Communication Protocol for Intelligent Sensors and Actuators of a Future Dialysis Machine

LTH School of Engineering at Campus Helsingborg
Computer Science and Engineering

Bachelor thesis:
Oi Kwan Ku
Diar Asivand

LTH School of Engineering
Lund University
Box 882
SE-251 08 Helsingborg
Sweden

LTH Ingenjörshögskolan vid Campus Helsingborg
Lunds universitet
Box 882
251 08 Helsingborg

Printed in Sweden
Media-Tryck
Biblioteksdirektionen
Lunds universitet
Lund 2011

Abstract

The thesis is done for the company Gambro which manufactures dialysis products worldwide. The idea of this thesis is to investigate weaknesses of the communication protocol which is used in one of the Gambro dialysis machines, Prismaflex and thereby investigate opportunities to improve the system by using other protocols.

The thesis was divided into three major parts. They were the study of Prismaflex, the study of several existing protocols and the recommendation of testing environment, so called migration steps. All studies and recommendation in this thesis were done on theoretical perspective. The conclusion of the study is that FlexRay is the most interesting protocol to study and would be most appropriate for the future dialysis machine. The thesis also resulted in a number of proposed analysis and test inclusive components and system structure analysis, unit test, system test, performance test and cost analysis which provides possible instructions of tests for the next study of a future dialysis machine.

Keywords: CAN/LIN, Communication protocol, Dialysis machine, FlexRay, I^{2}C, MOST

Sammanfattning

Projektet är gjort för företaget Gambro som tillverkar dialysprodukter över hela världen. Tanken med uppsatsen är att undersöka brister i det kommunikationsprotokoll som används i en av de Gambro dialysmaskiner, Prismaflex och därigenom undersöka möjligheter att förbättra systemet genom att använda andra protokoll.

Examensarbetet var indelad i tre större delar. De var studiet av Prismaflex, studiet av flera befintliga protokoll och rekommendationen av testing miljö, så kallad migrationssteg. Alla studier och rekommendation i detta exjobb var gjorda ur teoretiska perspektiv. Slutsatsen av studien är att FlexRay är det mest intressanta protokollet att studera och skulle vara det lämpligaste protokollet för en framtid dialysmaskin. Examensarbetet resulterade också i ett antal förslag till analys och tester inklusive komponenter och systemstruktur analys, enhetstest, systemtest, prestandatest och kostnadsanalys vilket ger möjliga instruktioner av tester för nästa studie av en framtid dialysmaskinen.

Nyckelord: CAN/LIN, Kommunikationsprotokoll, Dialysmaskin, FlexRay, I^2C, MOST

Foreword

Many thanks to Gambro for giving us the opportunity to do this thesis. We would like to thank our advisor at Gambro, Ulf Bergsten, Tommi Larsson and Johan Johansson for their timely supports and giving constructive suggestions to our work.

We would also like to express thanks to our advisor at LTH, Christian Nyberg for his constant support and patience. He not only taught us the knowledge in data communication but also gave us clear guidance so that our thesis could keep moving forwards.

Last but not least, we would like to convey our sincere thanks to our examiner Mats Lilja for his support and encouragement to achieve our goals.

We would like to thank all the people we met over the period of our thesis for their invaluable assistance and guidance. Without you, this thesis would not have been possible.

List of contents

1 Introduction

This thesis describes communications protocols for intelligent sensors and actuators of a future dialysis machine for Gambro. The following gives a basic picture about the background and the purpose of this thesis. Afterwards methods that were used to conduct this thesis and issues are briefly described. Abbreviations and words which are explained are in italics the first time they are named.

1.1 Background

Gambro is a global medical technology company which produces and develops products for dialysis treatment [16]. The dialysis machine that this thesis focuses on is Prismaflex. Prismaflex is used in emergency care in the blood purification treatment [9]. There are a variety of scales in the machine to measure pressure, flow rate, temperature etc. Based on these measurements actuators react to accommodate to the optimized state.

The electronic communication between different units in dialysis machines has an important role to play in reliability, functionality and flexibility. The communication protocol that Prismaflex uses now is Inter-Integrated Circuit (I^2C) which is a serial bus interface developed by Philips [8]. It is a relatively simple protocol that uses only two signal wires. However, it has disadvantages such as interference problems. Therefore, a development that Gambro is interested in is to evaluate new communication protocols for dialysis machines to achieve increased reliability of products, also in a cost effective manner. A short description of I^2C can be found in chapter 2.3. This thesis is the first step of the project and the study will be possibly continued by other students afterwards.

1.2 Purpose

The purpose of this thesis is to evaluate and compare different communication protocols for dialysis machines in Gambro with a view to be applied to future dialysis machines. The result of this thesis can serve as a theoretical foundation for which protocol/technology to use in a next generation dialysis machine.

1.3 Issues

In order to achieve the goal, the most concerned questions are raised and are listed as follows.

- How does the dialysis machine, Prismaflex work?
- What kind of properties do the different communication protocols have?
- Which of those protocols would fit the development of dialysis machines?
- How can the selected protocols be tested?

1.4 Methodology

Before the thesis was started, a weekly planning was constructed. The whole period was divided into four phases (Figure 1.1).

Phase	Week	Tasks	Time needed (hours)
1	9	Investigate CAN/LIN, FlexRay, MOST, USB, TCP/IP and deselect those protocols that do not fit	240
	10		
	11		
	12		
2	13	Investigate protocol used in Gambro	320
	14		
	15		
	16		
3	17	Establish criteria and according to the criteria evaluate CAN / LIN, FlexRay and MOST	320
	18		
	19		
	20		
4	21	Recommend testing environment	160
	22		
	23	Write final report and prepare presentation	

Figure 1.1 Time plan

1.4.1 Interviews with Gambro staff

Gambro staff have much experiences on the machines and are knowledgeable in Prismaflex system therefore interviews are necessary. The interviews were partly aimed at understanding the design of Prismaflex partly to articulate the requirements for next generation dialysis system. The interviews focused points including the current design of Prismaflex, data access, bandwidth and error handling.

1.4.2 Study of existing network protocols

Protocols that were studied were CAN, LIN, FlexRay, MOST, USB and TCP/IP. These studies were conducted with the help of different literatures, documents and internet. All these references can be found in chapter 11. The reason for studying these protocols is that they are widely used in different fields of industries and may have better ability in the use of dialysis system. The studies would first of all focus on overall understanding of the important features of the protocols. Then a first assessment would be made so as to sieve out unsuitable protocols. After that, the remaining protocols would be studied in detail according to different evaluation factors.

1.5 Limitations

There are several limitations in this thesis. First of all, only one dialysis machine (Prismaflex) was studied which didn’t provide comparisons between different dialysis machines. Secondly, there are many other communication protocols on the market but this thesis just focuses on a number of selected protocols. The more protocols which are studied the more objective the result would be. Thirdly, the thesis was only done theoretically because of lack of time and at the same time Gambro was not able to provide a test environment for this thesis.

2 Embedded system

This chapter describes the concept of protocol and serial bus system associated with I^2C which is presently used in Prismaflex as the communication protocol.

2.1 Protocol

A protocol is an agreed format for communications between two or more computing elements. It is essential because it establishes rules that ensure transfer of data. The rules include for instance the data format, transfer speed and control methods [4].

2.2 Serial Bus System

The transmission of data is usually considered in two transmission modes: Parallel and serial. This thesis focuses only on the serial bus system.

In a serial bus system, different electronic components share one common transmission and communication medium. All information, for example, control, address and data are sent on the same serial bus. Information is transmitted serially bitwise and can be accepted in principle by all participants in the bus nodes. This reduces costs, the need of space and weight compared to traditional point-to-point connections using cables and parallel transmission. A protocol should, however, have features that avoid any possibility of collision, data loss or blocking of information. Fast devices must also be able to communicate with slow devices.

Serial communications can be done with different methods, for example, synchronous or asynchronous. In synchronous communication, there are two or more periodic processes which have exactly the same period. Synchronism between the sending and receiving end must be constantly maintained. With asynchronous communication, the information is transmitted character by character and no synchronized clock signal is required. Synchronization of each character is done with a start bit. Afterwards, a number of data bits are transmitted followed by parity and stop bits. This method is used most commonly at low transmission speeds [2].

When using the serial bus system, the following aspects should be considered:

- Framing
- Addressing
- Data traffic
- Bus access

- Error detection and error handling
- Synchronization

2.3 I^{2}C

I^{2}C is used in Prismaflex as communication protocol. It is a multi-master serial bus protocol which is simple, inexpensive and has been widely used for more than twenty years. The rationale for developing I^{2}C was to add features to microcontroller (MCU)/designs using few pins on the MCU. This leads to a low-bandwidth, short distance protocol for board communication. Some important features of I^{2}C are as follows [11].

Each component on the bus has a unique predefined address and all components are associated over only two bidirectional wires, one for data, Serial Data Line (SDA) and one for clock, Serial Clock Line (SCL) which is used to synchronize all data that are transferred over the bus (Figure 2.1). When the bus is free, both lines are high. It is also designed so that components can be safely inserted or removed from the bus without affecting other circuits on the bus.

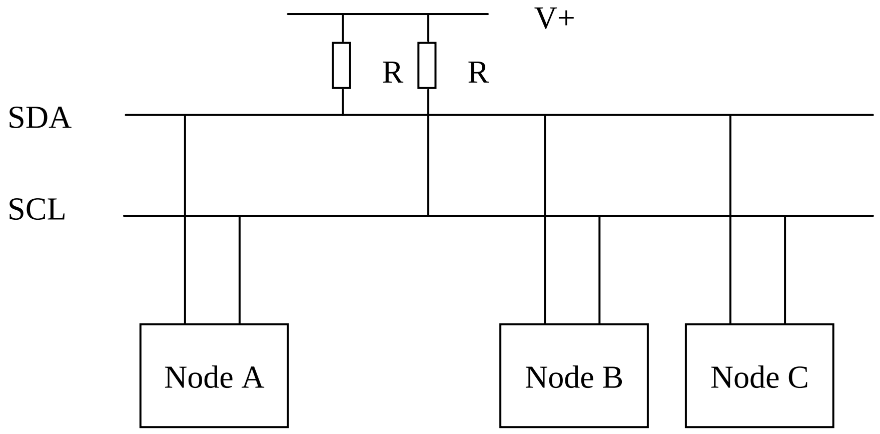


Figure 2.1 I^{2}C is a two-wire serial bus

The master is always the entity that drives the SCL and initiates a transfer over the I^{2}C bus. It is possible to have multiple masters. When a master wants to talk or stop talking to a slave, the master issues a start sequence that marks the beginning and a stop sequence that marks the end of a transfer. The two sequences are permitted to be applied only when SCL is high, as shown in figure 2.2.

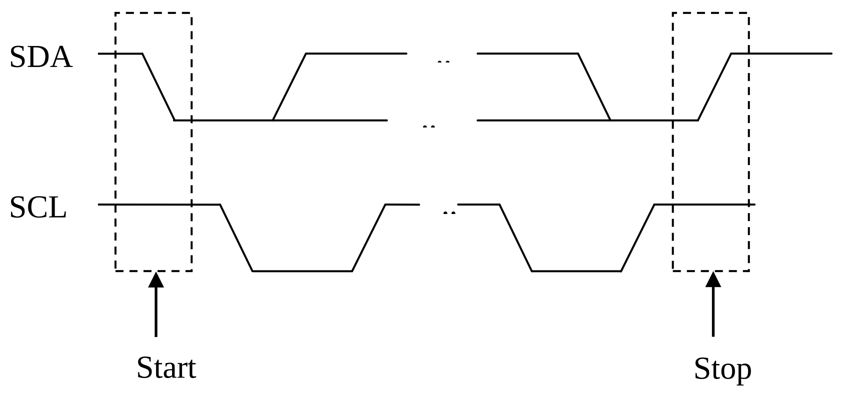


Figure 2.2 Conditions for the start and stop sequences

The version of I^{2}C that Prismaflex uses has 7 address bits. The eighth bit that is added to the 7-bit address is used to inform the slave if the master writes or reads from it. “0” indicates write signal while “1” indicates read signal. The message format is shown in figure 2.3:

Start	Slave address	Read/ Write	*ACK*	Data	ACK	Data	ACK	Stop
1 bit	7 bits	1 bit	1 bit	8 bits	1 bit	8 bits	1 bit	1 bit

Figure 2.3 I^{2}C message packet

3 Study of Prismaflex

To be able to evaluate new protocols it is important to study the dialysis machine, Prismaflex. The main function of Prismaflex is to pump blood from the patient and interchange fluids between the blood, effluent, dialysate and so on to provide acute blood purification [9].

In this chapter, Prismaflex is described from the highest level (how treatments are done by different modules) to lower levels (which modules and signals are needed in a special action and how signals are transmitted between different modules).

3.1 Background

There are three main therapies in Prismaflex [9], Continuous Renal Replacement Therapy (CRRT), Therapeutic Plasma Exchange Therapy (TPE) and Hemperfusion (HP). CRRT is a treatment for acute renal failure and/or fluid overload. TPE is used for patients who need to remove plasma components. HP is for those patients who need to remove toxic substances from the body. Figure 3.1 describes one of the therapies in CRRT, Continuous Veno-venous Hemodiafiltration (CVVHDF). CVVHDF is taken as an example here because the therapy uses all possible fluids which help us to understand the overall functions of all used devices and how they work together in Prismaflex.

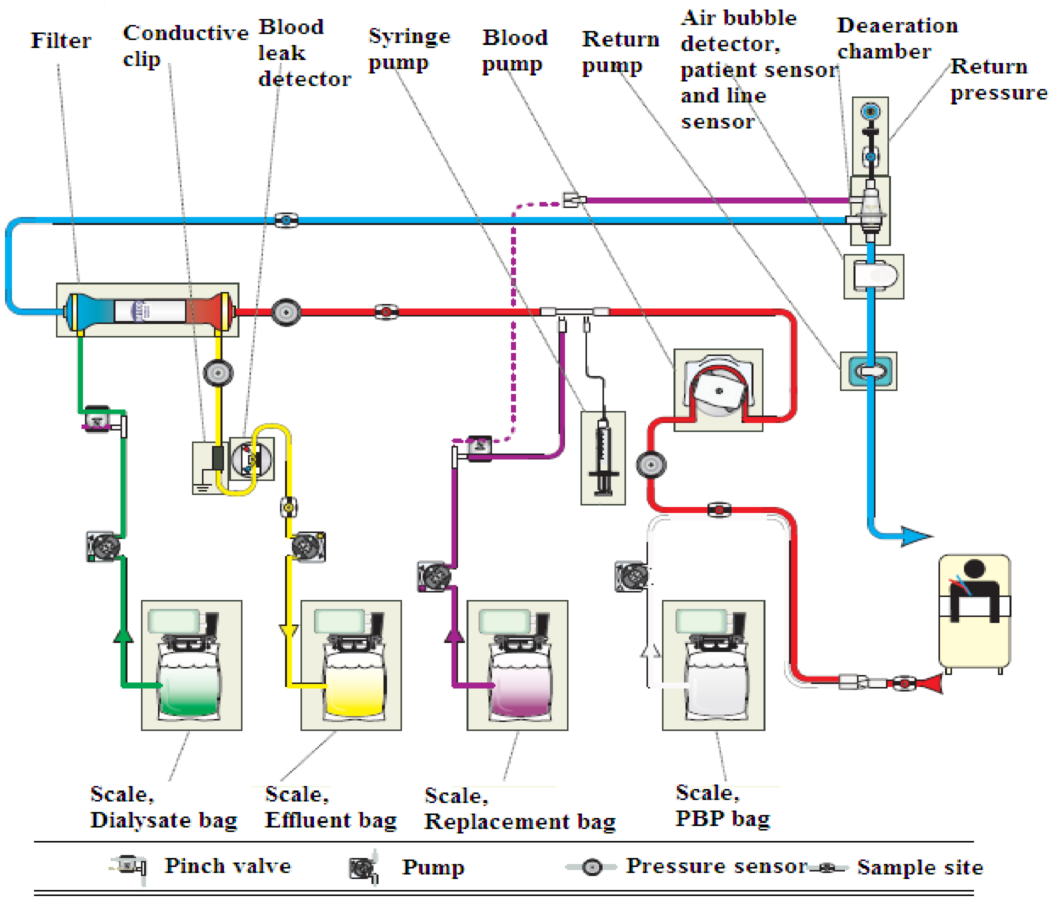


Figure 3.1 CVVHDF flow from service manual for Prismaflex, adopted from [9]

In order to drive the treatment 16 modules are. They communicate on the I^2C bus and among them there are 2 masters, the control and the protective and 14 of them are slaves which are:

1. Auto Repositioning System (APRS)
2. Peripheral Interface Board (PIB)
3. Pre-blood pump
4. Dialysate pump
5. Replacement pump
6. Effluent pump
7. Blood pump
8. Syringe pump
9. Loader
10. Dialysate scale

11. Effluent scale
12. Replacement scale
13. Pre-blood scale
14. Power Supervision Board (PSB)

Appendix A shows the overall connection diagram. The following are descriptions of the main purposes of these modules.

- **Control CPU** is responsible for the whole treatment. It supervises almost all the behavior of the machine and controls slaves like Auto Repositioning System (APRS), Peripheral Interface Board (PIB), fluid pumps, blood pump, syringe pump, scales and loader. In addition, it manages requests from the protective.
- **Protective** is a type of security that monitors basically everything. It talks partly to all the slaves as the control CPU does, partly with the control CPU to provide double checking. It has moreover electrical inputs that listen to specific pumps. Because the protective manages all processes, it generates the most data on the bus.
- **ARPS** monitors pressure values from different pressure sensors and controls pressure sensor valves.
- **PIB** is used to handle different peripheral interfaces including air bubble detector, blood detector, blood leak detector, venous clamp and pinch valves.
- Fluid pumps include dialysate pump, effluent pump, replacement pump and pre-blood pump. They are all peristaltic and have common commands to regulate the pump rotation direction and speed. Which pumps will be used depends on which therapy the patient needs. In CVVHDF, all of the fluid pumps are used.
  Pre-blood pump: Pumps a supplemental solution for hemodilution or anticoagulation.
  Dialysate pump: Pumps dialysate solution into the fluid. Dialysate solution helps to purify the blood through filter into effluent.
  Replacement pump: Pumps replacement fluid into the blood flow path. The purpose of using replacement fluid is that it can be added to the blood either pre and/or post-filter in order to add back water removed to help maintain a balanced fluid level.
  Effluent pump: In CRRT, it is used for pumping ultrafiltrate/dialysate. Based on the operator-set patient fluid removal rate, PBP solution rate, dialysate and replacement solution rate, it controls the ultrafiltration rate.
- **Blood pump** is a peristaltic pump like other fluid pumps and is responsible to pump blood through the flow path of the Prismaflex set.

- **Syringe pump** delivers anticoagulant to the blood flow. Syringe pump differs from other fluid pumps by means of using different mechanisms to add the fluid needed. Here it moves its actuator to push the piston of a syringe through the use of force sensor. It can even detect end-of-stroke, overload and the presence of a syringe.
- **Loader** pulls an attached line-set in proper position so that the pumps can route the pump segments into right runways.
- **Dialysate scale**, **effluent scale**, **replacement scale** and **pre-blood scale** can be considered in one group, scales. They read weight of different fluid bags to the control and protective through different channels. They also report to the protective whether the bag holder is properly inserted into the scale or not. The scales ensure that the fluid flow is accordance with the speed.
- **PSB** is a type of alarm which alerts if the power supply is interrupted suddenly.

3.2 Applications in Prismaflex

All data is transmitted in digital forms which means that all analog signals like pressure, weight must be transferred into digital signals first so that the sensors can interpret them as their nearest understandable values and afterwards further convert the data into corresponding values. Although there is derivation from the exact values, the derivation is small and does not affect the normal treatment. The following are some of the applications used in Prismaflex.

3.2.1 Flow rate regulation

One of the main applications in Prismaflex is flow rate regulation [9]. All liquids pass through the scales. Pumps pull in or give fluids by reading out from the scales. The regulation is handled by the control, i.e. Control CPU. The medical personnel first specifies a desired flow rate before starting the treatment. Once the treatment has started, the control collects data from scales continuously by sending command, “Weight Request” [8]. It comprises three bytes of data to store the weight value of a fluid bag, as shown in figure 3.2.

V[2] (8 bits)	V[1] (8 bits)	V[0] (8 bits)

Figure 3.2 Parameters in “Weight Request”

When the fluid is moving by means of pumps, it changes the weight of the fluid bags. The change in weight of the fluid bag helps the control to calculate the actual flow rate. If this actual flow rate differs from the desired value, the control adjusts the flow rate by changing the speed of the pump with the command, “rotation speed setting” [8]. This command comprises 4 parameters to describe how the rotation speed and direction can be set (Figure 3.3). The actual rotation speed is calculated through $p \cdot 10^{r \cdot q}$ rpm.

Rotation speed resolution, p (16 bits)	Rotation direction (1 bit) Clockwise: 0 Anti-clockwise: 1	Rotation speed scale, q (6 bits)	Scale sign, r (1 bit) Positive: 1 Negative: 0

Figure 3.3 Parameters in “Rotation Speed Setting”

3.2.2 Pressure management

Another example is the pressure management [9]. Again, the pressure values are predefined in advance. Once the treatment begins, the three pressure pods which are located in various places of the machine begin to measure pressures. Each pod contains a diaphragm which separates a fluid compartment and an air compartment. During a treatment, the fluid compartment is filled with the flowing fluid and the fluid pressure causes the diaphragm to expand or compress the air compartment. The pressure sensor which is located behind it receives this fluctuation and converts it to electrical signal. The control unit can then interpret that as a pressure value. To maintain the required pressure within an acceptable tolerance, the command “Set Pressure” is used [8] (Figure 3.4). It sets a specific pressure set point for a specific pressure sensor by keeping the motor active until the pressure read on the specific sensor agrees with the pressure set point.

Step motor counting (1 bit) Not required: 0 Start counting: 1	Sensor that the pressure is set on (3 bits)	Pressure set point (10 bits)	Tolerance on pressure to set (8 bits)

Figure 3.4 Parameters in “Set Pressure”

After the command execution is finished, the ARPS module will set a flag in the status byte. By calling “Status Request” (Figure 3.5), the master module knows whether the pressure has been set or not by checking actual ARPS action.

Motor phases power enable (1 bit)	Module enabling (1 bit)	Pressure sensor testing output (1 bit)	Valves command (5 bits)	ARPS motor actuation (1 bit)	Actual ARPS action (1 bit)	Valves output driver status (5 bits)

Figure 3.5 Parameters in “Set Pressure”

In contrast to flow rate control, pressure management runs in a more simple way. For pressure management, the command “Set Pressure” automatically performs several actions to run a specific task through only one command while for flow rate control, the control should send “Weight Request” and “Rotation Speed Setting” continuously to achieve an optimal flow rate.

3.3 I^{2}C in Prismaflex

In the case of Prismaflex, there are two protocol levels, namely line and application respectively (Figure 3.6).

Line:

Address (Receiver)	Address (Sender)	*LEN*	Counter	Data	Checksum

Application:

Address (Receiver)	Address (Sender)	LEN	Counter	Message ID	Count /ACK	Data	Checksum

$\longleftrightarrow$ Data of line level

Figure 3.6 I^{2}C packet of Prismaflex
* As in general, I^{2}C, there is one bit ACK for each byte of data

The line level is the lowest protocol level and is common to all units. The line level ensures right recipient, right length, and right checksum. If they are not correct it would send back a message to request retransmission. But if it appears that everything is correct, data would be transported to the unit that takes care of the application level where it will look at the message ID and the counter to verify if these values are reasonable. For example, a message ID expects a certain amount of data and if it is not true then it would insert a marker and send the message back again (Figure 3.7).

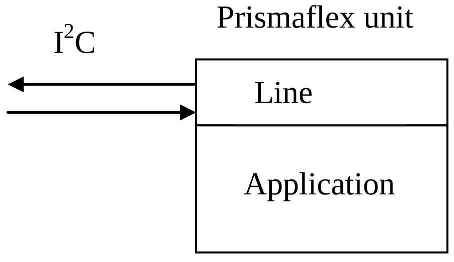


Figure 3.7 Communication layers in Prismaflex

4 Study of the existing protocols

In this chapter the other existing protocols are studied to get a clearer understanding of their basic structures.

4.1 Overview of protocols: CAN/LIN, FlexRay, MOST, USB, TCP/IP

In the modern applications a variety of serial bus systems are used. In this thesis, Controller Area Network/Local Interconnect Network (CAN/LIN), FlexRay, Media Oriented Systems Transport (MOST), Universal Serial Bus (USB) and Transmission Control Protocol/Internet Protocol (TCP/IP) are studied and compared in order to find which of them suits the dialysis machines best. The following is an overview of the different protocols with short descriptions.

CAN has been standardized internationally in 1994 and is undoubtedly the most widely used serial bus system, which can be used in many potential areas such as automotive, healthcare, retail and agriculture [15].

LIN is usually used as sub-bus for CAN because it provides cost-effective and easy data transfer between sensors and actuators. The data rate is limited to 20kbit/s but this is quite sufficient for transferring non-critical signals [15].

FlexRay is designed to be faster and more reliable than CAN. It is stable, deterministic and fault-tolerant but is also more expensive. It has a fast maximum data rate of 20Mb/s and uses two communication channels which allow redundant data transmission [21].

MOST provides a relatively high bandwidth and is used mostly for multimedia infotainment networks where devices such as navigation, radio and telephone requires a relatively high bandwidth so that signals like video and audio can be transmitted continuously [20].

USB was originally released for the personal computer for example to establish communication between devices such as a printer, mouse and personal computer. But now it has expanded to the use of embedded systems [15].

TCP/IP is a set of protocols and is designed for large interconnected networks of computers [2].

4.2 The CAN/LIN Protocol

4.2.1 CAN Basic Structure

CAN is a serial bus system, which originally is developed for automobiles in the early 1980's by Bosch. It has spread rapidly in many other industries. The reasons for its wide use are its reliability, stability and simplicity. Figure 4.1 shows the CAN's topology [6].

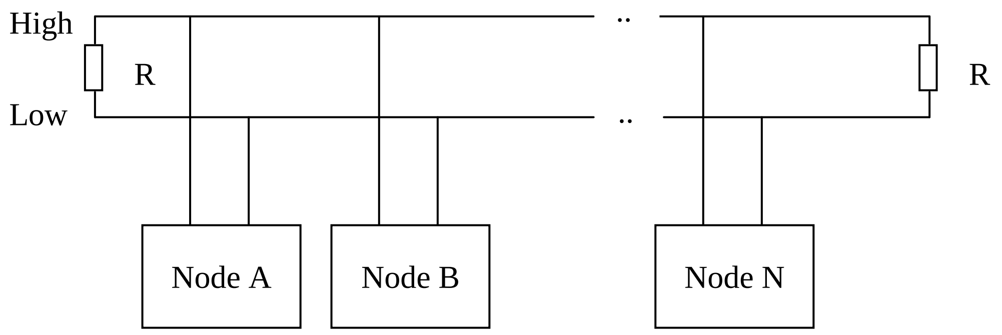


Figure 4.1 CAN's topology

Bus access is event-driven where the communication is priority-driven.
CAN has four frame types. A time interval (interframe) distinguishes these frames [6]. The four frame types are:

- Data frame: Contains data for transmission
- Remote frame: Request transfer of a specific identity
- Error frame: Transmitted by a node once the node detects an error
- Overload frame: Requests a time interval between the preceding and the following data or remote frame.

There are two message formats in the data frame: base frame format (with 11-bit ID, CAN 2.0A) and extended frame (with 29-bit ID, CAN 2.0B). Note that the identifier of CAN is message identifier but not node address. The idea is that the message must be sent to all nodes on the bus. The nodes can then filter the incoming messages and will only accept a certain type of messages that have been registered in advance. This increases flexibility by the fact that each node can be programmed to determine a certain type of message it will accept. Message format of the data frame using the base frame format is shown in figure 4.2.

SOF 1 bit	Message ID 11 bit	*RTR* 1 bit	Control 6 bits	Data 0-8 bytes	*CRC* 16 bits	ACK 2 bit	*EOF* 7 bits

Figure 4.2 CAN 2.0A message data frame

*Message ID: unique identifier for data sent with the message priority.
*Control: indicates the number of bytes of data that will be sent.

Data transfers occur on a binary model with dominant and recessive bits. Dominant represents always a logical 0 and recessive a logical 1. AND-operation can then be used in the sense that if a node sends a dominant bit and another sends a recessive bit, the dominant part wins. A dominant bit sent by creating a voltage difference, i.e. CAN HIGH is higher than CAN LOW. If both lines are at the same voltage, it means that the signal is for a recessive bit [6].

4.2.2 LIN Basic Structure

The original purpose of LIN is to provide a sub-bus for the CAN to provide an economical solution where performance (higher bit rate and network bandwidth), robustness and versatility of CAN are not needed (Figure 4.3). In practical, LIN combines with other protocols such as CAN to achieve both performance and economic efficiency [15].

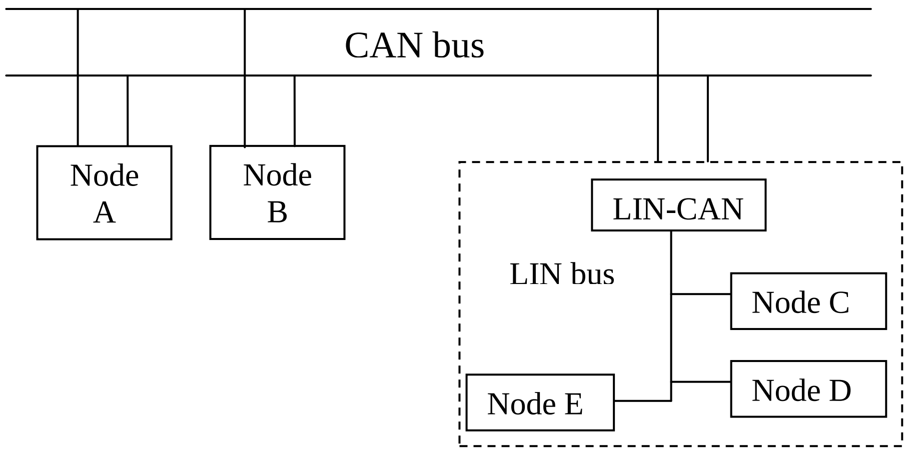


Figure 4.3 LIN-bus works with CAN bus

The LIN protocol is constructed for the two lowest layers in the ISO/OSI reference model: physical layer and data link layer [12]. It is even simpler than I^2C in such a way that it uses Master-Slave structure, that is to say it contains only one master. The master gives both master task and slave task. Master task is used for controlling the communication over the LIN bus which is fully controlled by the master and each slave has one slave task at a time to perform specified work (Figure 4.4).

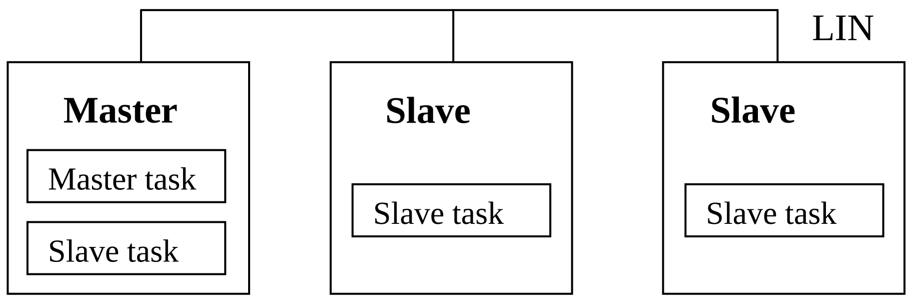


Figure 4.4 LIN's topology

The basic unit for transmission on the LIN bus is in the LIN message frame which consists of a header and a response element (Figure 4.5). The header has a fixed length and is divided into three parts: *SYNC*-break, the SYNC field and identify field while the response comprises 0-8 bytes of data and 1 byte of checksum [12].

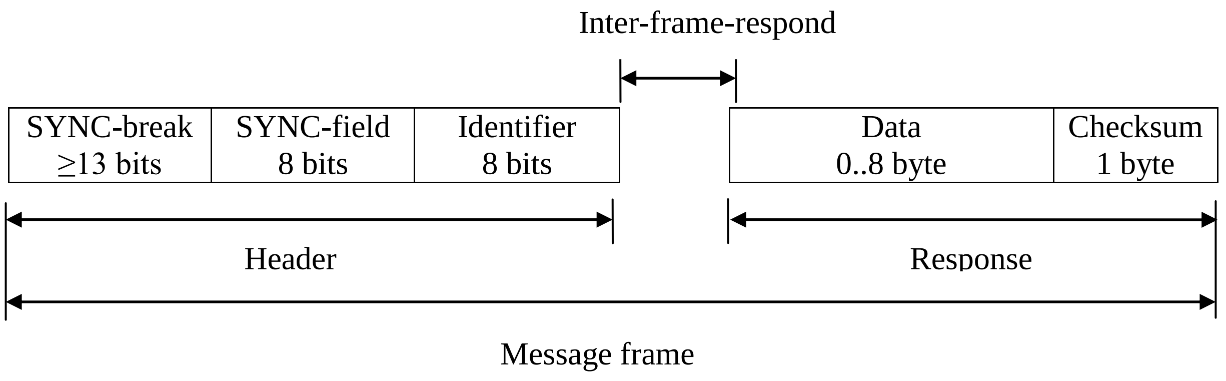


Figure 4.5 LIN message data frame

*SYNC-break: Works as a start-of-frame message to all nodes on the bus. It is required for slaves to discover that a message is sent on the bus.
*SYNC field: Allows a slave to measure the time of the baud rate and adjust its internal baud rate to synchronize with the bus.
*Identifier: Includes 6-bit ID and 2-bit parity and used to determine which nodes should receive or respond to any transfer.
*Inter-frame-respond: The time it takes for a slave to respond to a request from the master. It can vary between nodes in the network depending on hardware and software that are used in each node.

4.3 The FlexRay Protocol

FlexRay is a new protocol which is more expensive than CAN and LIN but has a higher performance. It is developed by a consortium of companies including Volvo, Motorola, Audi, BMW, DaimlerChrysler and Volkswagen. It is designed to develop faster and safer data transmission between sensors and actuators on an automobile [7].

FlexRay uses two communication channels, each of which has a data rate up to 10Mbit/s. Most FlexRay networks often use only one channel to reduce the cost but because the complexity is increasing all the time in applications, both channels are intended to be used in the future. The second channel can also act as a reliable redundant channel through same messages can be sent over both channels. Even if one channel does not work the data will still be sent to the destination via the second channel [7].

In addition to the larger bandwidth, FlexRay provides both time-triggered and event-triggered architectures. FlexRay also supports a wide range of network topologies, from a simple bus structure to hybrid topology. Because of these characteristics, many applications are addressable with FlexRay.

Figure 4.6 shows the frame for the message which consists of three parts, the header segment, the payload segment and the trailer segment. The header controls information such as sync frame flag, frame ID, frame length, 9-bit CRC and cycle counter. The payload section contains up to 254 bytes of data. The trailer includes a 24 bit-CRC that protects the privacy of the frame [7].

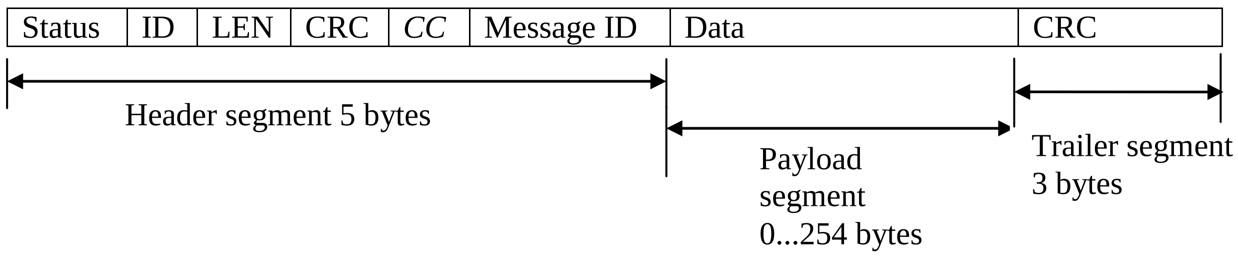


Figure 4.6 FlexRay Communication Package

4.4 The MOST Protocol

MOST is a fiber-optic networking technology optimized for multimedia and infotainment networking in the automotive industry but it can also be applied in many other industries. It is designed to be used with optical fiber, but it is also possible to implement via the electrical interface. Up to 64 units can be arranged in ring or star topology. It can transmit multiple streams of data such as control, package and real-time information simultaneously [1].

The description in this thesis is based on MOST25 [10]. Data is transmitted in blocks with a bit rate of 44.1 kHz (sampling rate of an audio CD). Each block consists of 16 packets and each packet is in 64 bytes, where 2 bytes are used for the transport control, and 60 bytes for the data and the last two bytes for the packet control (Figure 4.7).

The transport control part includes the preamble which synchronizes the timing slaves to the bit stream and boundary descriptor which determines the proportion between the synchronous area and the asynchronous area. Data part contains both the synchronous data and partly asynchronous Data. The control section contains for example arbitration, sender and recipient address and CRC [1].

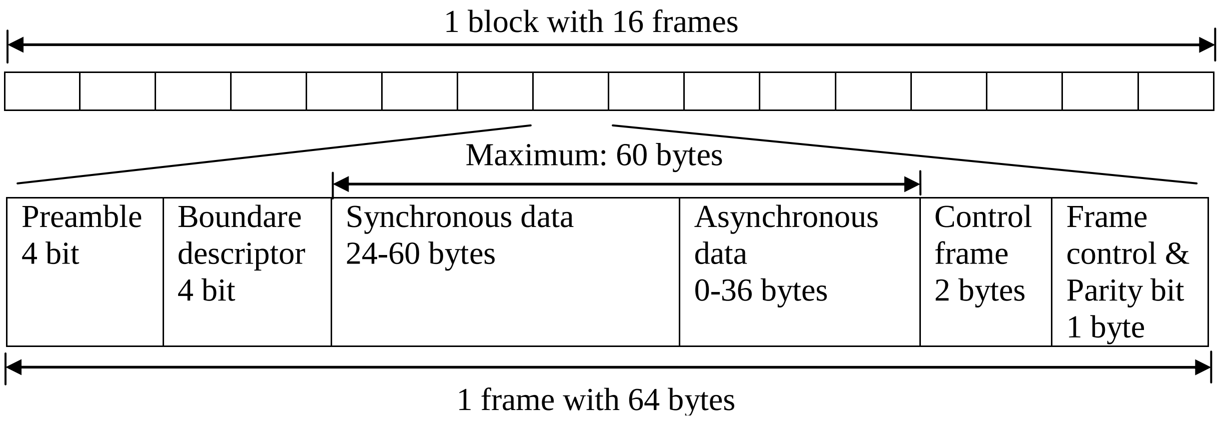

1 block with 16 frames
Maximum: 60 bytes
Preamble 4 bit
Boundare descriptor 4 bit
Synchronous data 24-60 bytes
Asynchronous data 0-36 bytes
Control frame 2 bytes
Frame control & Parity bit 1 byte
1 frame with 64 bytes

Figure 4.7 MOST communication package

4.5 The USB Protocol

USB was originally released for personal computers but now it has expanded to embedded systems and replaced old interface to increase ease of use, expandability and performance [13].

USB systems usually use star topology which is based on only one host that acts as a master (Figure 4.8). The host is connected with the hubs and other devices that work like slaves. A hub allows connection of multiple devices to a host or hub. In this way the number of units connected to a host increases. A host initiates all bus transfers and up to 127 USB devices can be either directly or through a hub connected to the host.

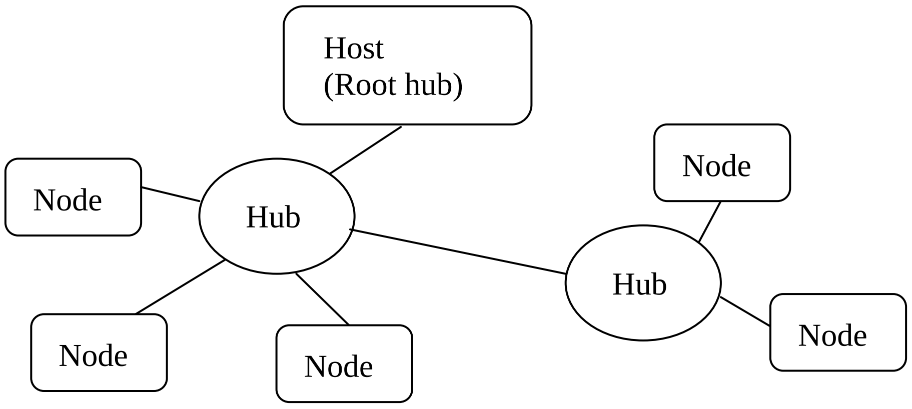


Figure 4.8 USB's star topology

Information is organized in the form of packets and frames in which a standard head part and an end part of each packet indicates the share of data between them. USB packages come in four basic types, all with different formats (Figure 4.9).

A transfer is started when the host sends a token packet (token packet) with a device address and direction of data transfer. The device decodes its address from the token and receives the package. If the direction field in the token indicates that the host asks for data, the unit will reply with the data packet otherwise, the host follows up with the data packet. After the data has been received by the host, the device sends a handshake packet to confirm receipt of the package. Moreover, there is also the start-of-frame packet that periodically synchronizes isochronous data streams. Isochronous data transfer is similar to synchronous data transfer, but with a fixed gap between two data streams which provides steady bit flow.

- Token packet

SYNC 8 bits	*PID* 8 bits	Address 7 bits	Endpoint 4 bits	CRC 5 bits	*EOP* 3 bits

- Data packet

SYNC 8 bits	PID 8 bits	Data 0-1023 bytes	CRC 16 bits	EOP 3 bits

- Handshake packet

SYNC 8 bits	PID 8 bits	EOP 3 bits

- Start-of-Frame packet

SYNC	PID	Frame number	CRC	EOP
8 bits	8 bits	11 bits	5 bits	3 bits

Figure 4.9 USB’s message packets

4.6 The TCP/IP Protocol

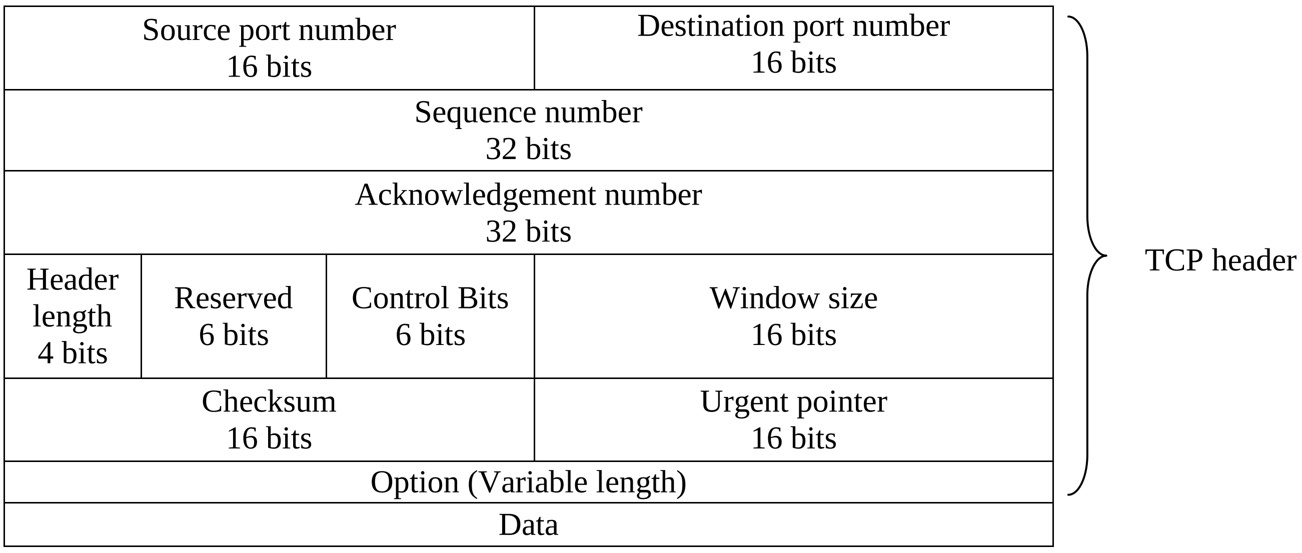
TCP/IP has been used more than 40 years and is designed for large networks. The protocol is suitable for connections that are relatively free of defects and require high transfer rate [2].

TCP/IP is divided into different layers, in many ways similar to the OSI model. The OSI model contains seven layers and TCP/IP is a model that contains four or five layers. TCP/IP consists of the following layers: application layer, transport layer, network layer, data link layer and physical layer. The four-layer model is the data link layer and physical layer merged into one and is called for the link layer (Figure 4.10) [2]. There is a set of protocols in each layer, namely TCP/IP is actually a set of communication protocols. TCP and IP are the most important protocols in it which define the transport respective internet layers. Besides them, Ethernet is the most dominant protocol used today in the local area network (LAN) market.

OSI Model	TCP/IP(DOD Model)	TCP/IP(Internet Protocol Suite)
Application	**Application**	**Telnet, SMTP, POP3, FTP, NNTP, HTTP, SNMP, DNS, SSH...**
Presentation		
Session		
Transport	**Transport**	**TCP, UDP**
Network	**Internet**	**IP, ICMP, ARP, DHCP**
Data Link	**Network Access**	**Ethernet, PPP, ADSL**
Physical		

Figure 4.10 OSI model and TCP/IP model

A computer that uses TCP/IP has an IP address, which identifies the host computer (server) and client computers. An Internet Protocol Version 4 (IPv4) packet has an address field with four decimal numbers ranging from 0 to 255, separated by dots. Each of the four numbers representing eight bits of the address and the address therefore consists of 32 bits. Figure 4.11 shows a TCP message packet and figure 4.12 shows the data flow of TCP/IP [2].



Source port number 16 bits			Destination port number 16 bits	TCP header
Sequence number 32 bits				
Acknowledgement number 32 bits				
Header length 4 bits	Reserved 6 bits	Control Bits 6 bits	Window size 16 bits	
Checksum 16 bits			Urgent pointer 16 bits	
Option (Variable length)				
Data				

Figure 4.11 TCP message packet

TCP/IP provides reliability, ensuring that data will be presented in the order they are sent. Since the physical network can not transmit large packet, this stream of bytes are assigned into suitable pieces (segments). Each segment is sent in a separate IP packet. When the IP packet reaches the receiver, the receiver sends an ACK back to the sender in order to show that the data have arrived. If the sender does not receive such an ACK segment within a certain time, the package will be retransmitted. A sequence number is also supplied to detect errors to protect integrity. For example, if a computer transmits 3000 bytes numbered from 0 to 2999 to a different computer, these 3000 bytes will be divided into three segments and each 1000 bytes. The first segment will then have sequence number 0, the second 1000 and the last 2000.

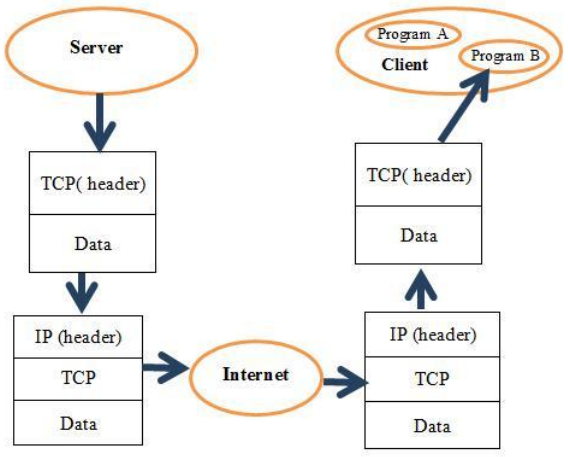


Figure 4.12 Data Flow of TCP / IP

5 Choices in determining a communication system for Prismaflex

This chapter introduces the system requirements, design objectives and evaluations factors for the design of future Prismaflex.

Before going further to the system requirements, USB and TCP/IP can already be evaluated in this stage. They will not be suitable for the use of Prismaflex because of the following reasons.

USB: It is not appropriate because it is a top controlled system that does not fit well to a distributed control system. USB is very fast, up to 4800Mbps (USB3.0) but USB is intended for communication outside "the box" and the transfer of data between the systems. With regard to the implementation of communication between components, such as a microcontroller and a small amount of peripheral, there is no point using too complicated protocols.

TCP/IP: An important factor to use TCP/IP is that it is an open standard. It is not tied to any particular supplier, and anyone may use the specification without paying license fees. Therefore, many companies, organizations and individuals participated in the development of protocol collection. But on the other hand, TCP/IP is a too large protocol due to the large amount of overhead which is not appropriate for the use of dialysis machine where the amount of data to be transferred is relatively small.

5.1 System Requirements

One of the key issues in evaluating the communications protocols is to examine the system requirements for Prismaflex. These requirements are set based on the current design of the Prismaflex system. The protocols should of course satisfy these requirements as a foundation to further development.

- The protocol shall be immune to noise. Noise is a considerable disturbance on I^2C bus that is used in Prismaflex.
- The protocol shall support connections with at least 16 components. The Prismaflex system uses currently 16 components in order to perform treatments.
- Data rate of the protocol shall be at least 100kbps. The I^2C bus speed in Prismaflex is 100kbps standard mode.

- The protocol shall be able to apply on real-time system. The current Prismaflex system is a real-time system that requires tasks to be performed within desired time.
- The protocol should manage collisions for bus access to avoid transmission error or data loss.
- The protocol should support error control methods for reliable data communication.
- The proportion of overhead should be as little as possible compared with the actual data size in order to have higher data transmission efficiency.
- Event-driven protocol is preferable. The current Prismaflex system uses event-driven application. It would be a merit if the new system can even adapt to the current system.

5.2 Design Objectives

Beside the system requirements, it is also necessary to have a clear understanding of the design objectives for Prismaflex in which the protocol will operate. This can be broken down into four kinds of objectives which are described as follows.

- **Cost effectiveness**: This can be described and measured in various ways depending on how it is defined and how detailed calculations can be done, for example, cost to add a new application, cost per message and cost per month for developing a system. As it may involve many unknown factors the actual expenses are not going to be estimated. Instead, it can be briefly considered through investigating the cost for all hardware and software needed to build up the system and microcontroller suppliers in the market. A more detailed recommendation for cost analysis can be founded in chapter 8.2.5.

- **Reliability**: The ability to perform functions which are required for a specified time. But before that, the following question should be identified: Which kinds of applications shall be supported? The nature of the device and application determine a large part of functionality requirements. Reliability can also refer to the ability to resist failure of a system. To provide reliability, the future protocol shall have features that can find and correct errors caused by, for instance transmission error or interference.

- **Performance**: How well the system can perform the desired tasks, for example the transmission time (How long is the delay?), protocol efficiency (How much traffic capacity is used in overhead?), data rate and data capacity.

- **Flexibility**: Having a reliable and high performance system, the next step is to concern about the flexibility. It refers to the ease to adapt new requirements and the ability to cope with uncertainty. A high level of flexibility can highly reduce development time and maintenance effort.

5.3 Evaluations Factors

Based on the design objectives, a number of categories can be cited in the evaluations of different protocols. After a discussion with personnel at Gambro, the categories can be fallen into the following major evaluations factors which are listed with its importance in descending order.

- **Methods to achieve noise immunity on physical layer**: The physical construction for dealing with noise.
- **Methods of transmission security**: Different error detection, signaling and correction methods to achieve accurate communication.
- **Data rate**: It refers to the data bit rate through the system.
- **Data capacity**: The amount of data which can be carried per frame.
- **Arbitration**: Bus access method to avoid data collisions.
- **Bus behavior**: The determinism characteristic of a protocol.
- **Data transmission efficiency**: The proportion of the actual useful data in relative to the overhead of the frame.
- **Topology**: Design layout of the communication system.

6 Comparative evaluations of communication protocols

The following table gives an overview of the properties of protocols with respect to different evaluation factors:

Protocol	**I^2C**	**CAN**	**LIN**	**FlexRay**	**MOST**
Methods to achieve noise immunity on physical layer (see 6.1)	Schmitt-trigger logic, Lower pull-up resistor	Balanced lines	Single-wire medium with restricted data rate	Bus Guardian	Optic fiber
Methods of transmission security (see 6.2)	ACK	CRC field, ACK, Bit stuffing, Bit monitoring, Frame check, Error frame, State unit	Parity bits, Checksum, diagnostic message, Status-management	Two communicat-ion channels, CRC, Protocol Operation Control	Double ring, Parity bit, Control channel(CRC ACK),Error message
Data rate (see 6.3)	Up to 3.4Mbps	Up to 1.6Mbps	Up to 20kbps	Up to 20Mbps	Up to 25Mbps
Data capacity (see 6.4)	Unrestricted	0..8bytes	0..8bytes	0..254bytes	0..60bytes
Arbitration (see 6.5)	Priority of data on SDA line	CSMA/CA	Collision resolving schedule	TDMA, FTDMA	TDMA, Token, CSMA
Bus behavior (see 6.6)	Non-Deterministic	Non-Deterministic	Deterministic	Deterministic / Non-Deterministic	Deterministic / Non-Deterministic
Data transmission efficiency (see 6.7)	>40%	Up to 59%	Up to 63%	Up to 97%	Up to 94%
Topology (see 6.8)	Bus	Bus	Bus	Bus, Star, Hybrid	Ring, Star, Hybrid

Table 6.1 Comparison between different communication protocols

6.1 Methods to achieve noise immunity on physical layer

Noise is a main problem in Prismaflex. In order to avoid errors caused by noise, a properly designed physical layer is the first important step to achieve a high level of noise immunity.

I^2C:
I^2C is not designed for robustness but there are still several ways to improve the ability of noise reduction.

Schmitt-trigger logic: It is a decision-making circuit which gives two binary states. If the input voltage differs from the preset values, the circuit will draw back or pull up the input to the predefined state [19].
Lower pull-up value: Manufacturer always define recommended values for pull-up resistors. A lower value of pull-up resistor would help maintain the signal integrity [18].

CAN:
Balanced lines: In the physical layer, both CAN HIGH and CAN LOW lines are balanced. The two wires are routed in parallel so they are in the same condition for electromagnetic influence. The difference in voltage between the two lines does not vary, and because of this CAN is insensitive to electromagnetic interference.
Both wires have also the same impedance to ground. The voltage induced by noise is the same in both wires. As the receiver at the end operates only on the difference between the wires, if the two wires are affected by induced noise, the receiver is immune to the noise [22].

LIN:
Single-wire medium with restricted data rate: As LIN uses single-wire transmission medium, the maximum data rate is restricted to 20kbps in order to reduce the effect of electromagnetic interference [3].

FlexRay:
Bus guardian: The bus guardian is an optional electronic component between the communication controller and the communication medium. It ensures safe data exchange by protecting channel from interference on physical layer. It stores its own communication schedule independently. When it detects communication that is not aligned with the schedule that means time failure exists. It will restrict transmission attempts from the communication controller to those times allowed by the schedule [7].

MOST:
Optic fiber: MOST supports the use of optical fiber which has little heat emission and provide protection against electromagnetic interference as the optical fiber is less sensitive than copper [2].

6.2 Methods of transmission security

Even though noise and interference can be reduced or avoided on physical layer, there are still probabilities that errors may occur. The protocol must not only have functions to find and correct individual errors but also the ability to identify and disable a faulty node which may continuously create errors and thus interference the bus. Having effective methods of error control is therefore essential for reliable data communication.

I^2C:
Error detection: Data is transmitted in a sequence of 8 bits and the bits are places on the SDA starting with *MSB*. The receiving party sends an acknowledgment for each 8-bit transmitted to indicate the transmitter that the transfer was correct. There are nine clock pulses in total on SCL in order to transfer each 8-bit data.
For data transfer from master to slave, when all data is successfully transferred, the master will generate a stop bit. If a slave is busy with other functions (for example internal interrupt) and cannot receive the data completely, it can hold the clock line low to force the master to wait. The data transfer continues again when the slave is ready again. If the slave for some reason is unable to receive data at all it will generate a not-acknowledge (NAK) indicating to the master that the transfer has failed and the master will then abort the transfer. The master should generate a stop or a repeated start condition.
A similar process happens for data transfer from a slave to the master. After the desired data is received, the master will not generate acknowledge which signals the end of data. The slave-transmitter will then release the SDA line to allow the master to generate a stop or a repeated start condition [11] (Figure 6.1).

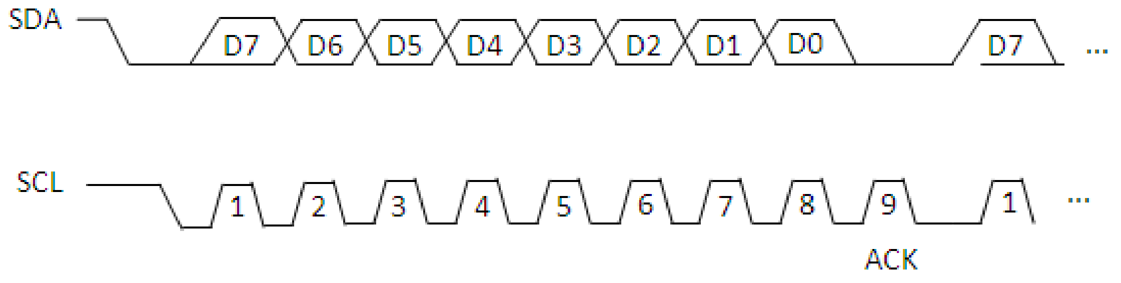


Figure 6.1 Data will continue to send when ACK is low

CAN:
The CAN protocol is designed for safe data transfer by providing error detection, error signaling with recovery time and fault confinement.
Error detection: CAN defines five ways for detecting errors [6].

1. CRC check: One of the main methods is the use of CRC field. It is placed after the data field and uses CRC-15, i.e. the generator polynomial is $x^{15}+x^{14}+ x^{10}+ x^{8}+ x^{7}+ x^{4}+ x^{3}+ x^{1}$ which has a degree of 15, will detect up to 5 single errors, all burst errors affecting any odd

number and will detect all burst errors with a length less than or equal to 15. The error rate for undetected corrupted messages is less than $4.7*10^{-11}$.

2. ACK check: Another method is the use of ACK. After the CRC check the receiver will check the consistency of the message. If the message is correctly received, an ACK bit will be sent. The transmitter will retransmit the message if no ACK bit is detected.
3. Bit stuffing: When five consecutive equal bits have been transmitted by a node, a sixth bit of the opposite level will be added to the outgoing bit stream by the transmitter. At the other end, the receiver will do the reverse operation and remove the extra bit. So if there are six consecutive bits of the same type (111111 or 000000) it will be signaled as a stuff error.
4. Bit monitoring: Each unit on the bus monitors the transmitted signal level. If the bit value that is monitored differs from the one transmitted, a bit error is detected.
5. Frame check: There is certain fixed form bit fields in CAN massages which are defined exactly when and how they occur. If one or more illegal bits occur on the bus, a frame error is signaled.

Error signaling with recovery time: All nodes that a message passes will try to detect errors in the frame. Whenever a fault is detected, a special frame called error frame will be sent by the discovering node. It contains an error flag which states the type of error followed by an error delimiter which provides space for other nodes to transmit their error flags. The other nodes detect the error flag and discard the faulty frame. The original frame will be automatically retransmitted. The recovery time, that is to say, the time interval between the moment an error is detected and the moment a new message starts should not more than 29 bits for CAN2.0A or 31 bits for CAN2.0B [6].
Fault confinement: There are a total of three states for each unit, error active, error passive and bus off. Error active unit means the unit functions properly and can take part in bus communication normally. When an error is detected, an error frame with active error flag will be sent. Error passive unit is suspected of having a certain faults. It can still take part in bus communication but its error signaling behavior is restricted through error passive flag. Bus off unit is considered as a faulty unit and it cannot make any influence on the bus. To define the state a unit has, two counts, transmit error count and receive error count are implemented in every bus unit. They are followed by a set of rules to calculate the condition of each unit [6], as shown in figure 6.2.

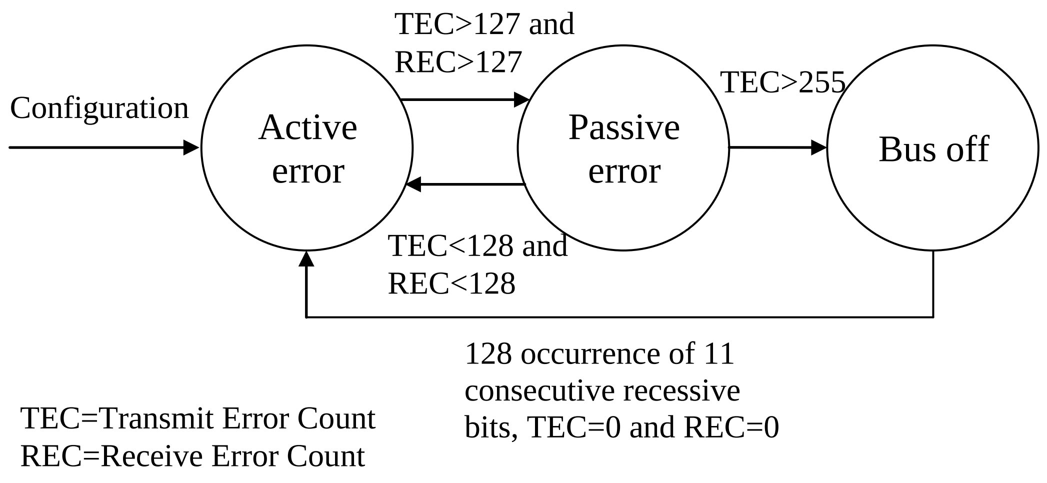


Figure 6.2 Node error states

LIN:
Error detection: Every transmitter must have a monitoring device which checks at any time that the sending message on the bus is consistent with the original message. Furthermore, two ID parity check bits are implemented in the identifier field for protection. It also uses a 1-byte checksum in the last field of a frame. LIN 1.x uses classic checksum (sum of data bytes) while LIN 2.x uses enhanced checksum (sum of data bytes and the protected identity) [12].
Error signaling: There is no error-signaling device but if an error is detected by a slave, the slave would save the information and send to the master in form of diagnostic messages. The master can then retransmit the message [12].
Status management: Master can fetch status reports from each node. Whenever an error is polled by the master, the LIN slave shall report to the master by a one bit signal, response_error in an unconditional frame. However, the LIN protocol specification does not standardize further detailed error information. The error handling is left for the user to define [12].

FlexRay:
Two communication channels: FlexRay allows single or dual-channel communication. To increase data security, both channels can be used to connect the nodes. So even if one channel does not work, the data will still be sent to the destination via the second channel [7].
Error detection: FlexRay has a rather strong CRC protection, consisting of 9 bits in header segment and 24 bits in trailer segment. The FlexRay should notify the microcontroller which governs if a message is received correctly [7].

Protocol Operation Control: When frames are not received within the expected time or clock synchronization error occurs the controller will degrade the sending node from active mode to passive mode. If errors persist, the protocol operation control will transmit the passive mode to halt mode. The conditions for degradation are configurable in the application level to give flexibility for design of different system [7].

MOST:
Double ring: All nodes can be connected through a double ring structure. If there is any error, the ring can be closed via the redundant segments. However, to realize this purpose, each node should have two optical receivers and two transmitters which is costly and also implies a high complexity in establishment [1].
Error detection: One parity bit is placed in the end of each frame for detecting bit errors in the frame.
A 4-byte CRC check is also used for protecting data in asynchronous frame area which enables the receiver to detect transmission errors.
In addition, there is a control channel which is distributed over the 16 frames and can be combined into one block. Each frame transports 2 bytes with a total of 32 bytes for one block. This channel is used for transporting commands, status and diagnostic information. Control messages carried by the channel are protected not only by CRC but also an acknowledgement mechanism. It is used for the receiver to inform the sender about the reception of message. There are three possible states: Message successfully received, cache blocked and CRC error [1].
Error signaling: If an error occurs during operation, the slave will return a message of the operation type Error (OPType: 0xF) to the controller. ErrorCodes table in MOST specification 3.0 defines all error messages with additional information parameter ErrorInfo. The error code, ErrorCode is sent in the first byte of the data field while the second byte contains error information, ErrorInfo [1].

6.3 Data rate

The data rate of a communication system determines the amount of data that can be sent in a specific unit of time. With a higher data rate, larger amount of data can be transmitted but at the same time higher cost and electromagnetic interference problems may also arise. Data rates comparison between different protocols is shown in figure 6.3.

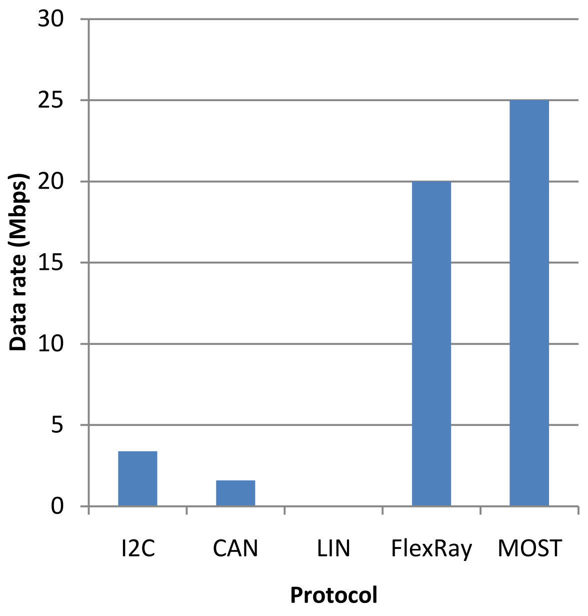


Figure 6.3 Data rates comparison

I^2C:

There are three modes for data transmission on the I^2C bus: Standard-mode: up to 100kbps, Fast-mode: up to 400kbps, High-speed mode: up to 3.4Mbps. Prismaflex uses standard-mode and currently uses approximately 40% of the maximum speed. Data traffic in Prismaflex is more or less continuous. Of course there are some peaks at times but there is not really a set of routine for how it looks like. As soon as the bus is free, data is sent [11].

CAN:

The data rate of CAN may be different in different system. The maximum transfer speed can be up to 1.6Mbps. For a system like Prismaflex in which long cable length is not considered, a high data rate can be achieved [3].

LIN:

The maximum data rate is 20kbps [12].

FlexRay:

Maximum data rate of 10Mbps for each channel, giving a total data rate up to 20Mbps [7].

MOST:

MOST25 which has a sampling rate of 44.1 kHz results in a data rate of around 25Mbps at a frame of length of 512 bits. There are even other versions, MOST50 and MOST 150. MOST50 has the same sampling rate but each frame has a length of 1024 bits which results in twice the data rate, 50Mbps. In 2007, MOST150 is released to provide an even faster data rate of 150Mbps with sampling rate of 48 kHz [20].

6.4 Data capacity

Data capacity is defined in this report as the amount of data that can be carried per frame.

I^{2}C:
Data transmitted per frame is unrestricted but data should be divided into bytes and each byte is followed by an acknowledge bit. Data is transferred with the most significant bit first [11].

CAN:
Data field in CAN frame consists of 0 to 8 bytes. Transmission begins with the most significant bit [6].

LIN:
Like CAN, a frame carries from 0 to8 bytes. However, here the least significant bit in a byte is sent first and the most significant bit in the byte sent last [12].

FlexRay:
The payload field of FlexRay can carry up to 254 bytes data which can be divided into static and dynamic segments [1].

MOST:
In MOST25, synchronous and asynchronous data shares the 60-byte data field in a frame for transmitting stream respective packet data. The synchronous area is specified to have a width of between 24 and 60 bytes while the asynchronous area has a width of between 0 and 36 bytes [1].

6.5 Arbitration

Arbitration is needed because when more than one node is trying to transmit data at the same time, collisions are possible.

I^{2}C:
Priority of data on SDA line: The bus arbitration in I^{2}C is done by the physical setup. When a node is using the bus, the node pulls the SDA line to

low. So when other nodes want to send data they will first see if the bus stays high or low. If it stays low it means that the bus is occupied by another node. However, if two nodes want to send data simultaneously, the node which sends the first “0” rules the bus. The other node will draw back and wait until the next condition where the state of the bus changes to high again. This is a so-called bitwise arbitration mechanism where the priority is based on the identifier [11] (Figure 6.4).

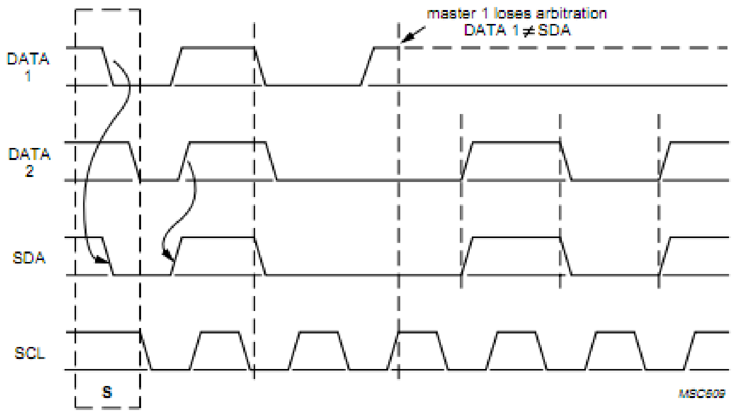


Figure 6.4 I^{2}C Arbitration process of two masters, adapted from [11].

CAN:
Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA): CAN frame transmission between nodes is event-driven which means that collisions are possible on the bus. Therefore it uses CSMA/CA to support arbitration which is the same type of principle as that of the I^{2}C bus. It is based on that each node can send and receive messages, which consists primarily of an ID and this ID represents the priority of the message. If two nodes are trying to send messages at the same time, the smaller value of ID would have a higher priority and be transmitted first, i.e. it ensures that messages with a high priority would be transmitted which means that CAN is very suitable for priority communications. However, this arbitration scheme does not allow deterministic scheduling of real-time events. The message with lower priority cannot be delivered predictably [6].

LIN:
Collision resolving schedule: There is no arbitration scheme for LIN because it has only one master. The master initiates and supervises all data transfer to avoid collisions. Only one master task is transferred at a time which will be responded by one slave task. But in case there are responses transmitted by more than one slave, error occurs. The slaves publish error signals and the

master will then resolve the collision by processing collision resolving schedule. Each frame has an associated collision resolving schedule table where the master queries responses from each slave by using unconditional frames [12].

FlexRay:
It uses a communication method in which a communication cycle comprises a static time-controlled segment and a dynamic event-driven segment [1].
Time Division Multiple Access (TDMA): Data that is time critical and real-time relevant often need synchronous transfers and is sent in the static segment. Bus access takes place under TDMA. The static window comprises equal slots assigned to nodes. Each node is synchronized to the clock each time when a transfer is started and each node is waiting for their turn to write on the bus. Since the clock is consistent in TDMA, FlexRay can guarantee determinism and avoid the data becoming corrupted when two nodes should write at the same time.
Flexible Time Division Multiple Access (FTDMA): For the data that has lower real time requirement, it would fit with asynchronous transfers and is sent in the dynamic segment. Unlike the static segment the length of message is flexible. Bus access here is influenced by FTDMA. In the dynamic window, slots are assigned to nodes according to message identifiers. Figure 6.5 constructs an example of a communication cycle. Node A and node C send data over both channels while the node B and node D are designed to send data over one single channel.

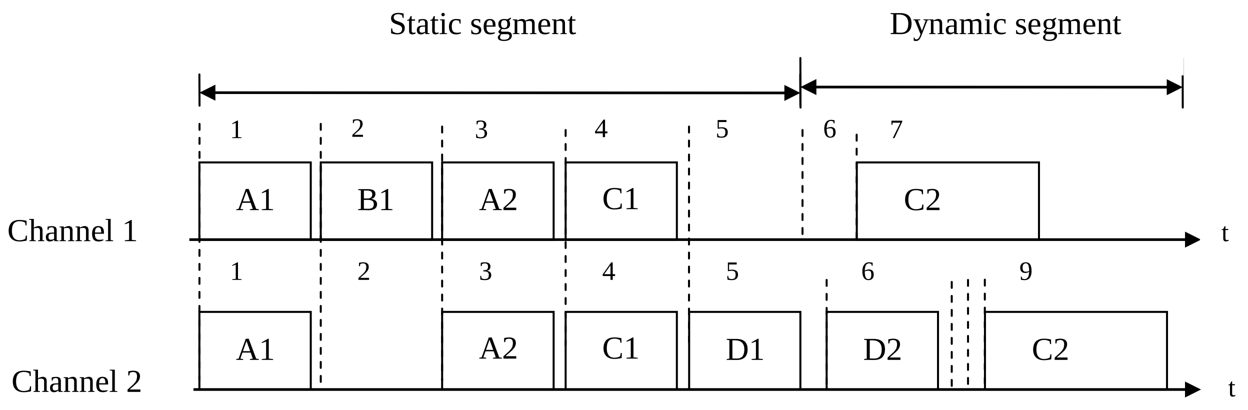


Figure 6.5 FlexRay Communications Cycle

MOST:
There are three different arbitration mechanisms for the three channels [1].

TDMA: In the synchronous channel, the arbitration according to TDMA runs where the data are cyclically sent at the same frame position in the same time slot.
Token: In the asynchronous channel, a token is provided with fair arbitration for each node. The token passes from one node to the next if the node does not wish to send data while if one node wishes to send data, it has to wait until the token comes and take the token as an authorization for data transfer.
CSMA: In the control channel, CSMA is used to guarantee a fair bus allocation.

6.6 Bus behavior

In general, all protocols which are studied have real-time properties, that is to say, all possible events are reacted in a predictable way but for systems that have strict real-time requirement, determinism of data is necessary.

A deterministic communication system guarantees defined communication rates of the signals under any condition [1]. As I^2C and CAN use a priority-based data transmission method, data with lower priority may become delayed depending on the actual bus load. It cannot be guaranteed that the receiver can get all types of messages in time thus they can be considered as non-deterministic.

LIN is however deterministic as there is a schedule table stored in the master determining the sequence and the time all messages should be sent. FlexRay is deterministic because the transmission time of each message in the static segment is known within specified limits while for data which have lower determinism requirement can be transferred through the dynamic segment. MOST combines multiple data access methods like FlexRay where CSMA is not deterministic while TDMA ensures determinism.

6.7 Data transmission efficiency

Overhead in the protocol aspect means all other information than the actual data used for routing and describing the context of the message. A large size of overhead allows usually for more error protections and other functions but this also expenses extra bus load. It may affect data transmission efficiency in such a way that each frame should take up a large amount of resources even if the size of the data source is small compared to the total protocol overheads. However, to evaluate the efficiency, data rate should also be taken into consideration because if the data rate is high enough the impact of overhead is low.

The following formula is a simple calculation of data transmission efficiency (E) without considering the data rate.

$$E = \frac{data\ size}{data\ size + protocol\ overheads}$$

	Data size(bits)	Protocol overhead (bits)	Efficiency
I^2C	8n, n≥1	12+n, n≥1	>38% *
CAN 2.0A	1..64	44	2%-59%.
LIN 2.1	1..64	37	3%-63%
FlexRay 2.1	1..2032	64	2%-97%.
MOST25	1..480	32	3%-94%

Table 6.2 Data transmission efficiency

*When n=1, E≈38% and when n increases, E will also increase.

6.8 Topology

Topology is about the interconnection of different nodes in a system. Commonly used topology includes bus, star, ring and hybrid. Each of them has its own advantage. In the case of Prismaflex, the implementation of nodes should not only achieve high reliability but also simplicity.
Before discussing the topology of each protocol, it is necessary to have basic concepts on different topologies [2].

- Bus: It contains a main run of line with a terminator at each end. All nodes are connected to the line. The installation is simple but if there is a break on the main line, the entire network shuts down and it is difficult to trace the problem.
- Star: Each node is connected directly to a central hub. All data flow pass through the hub to its destination. It is easy to install. Adding or removing devices causes no disruptions to the network. It is also easy to detect the faulty part which increases the system reliability. However, it requires more cable and is more expensive than bus topology. The failure of the central hub leads to the failure of the entire system.
- Ring: Nodes are connected to two other nodes so as to create a ring. Because data can be transmitted in two directions, higher speed of

transmission is possible. Changes or failure of a single node affect the entire network.

- Hybrid: It uses a combination of two or more standard topologies like bus, star, and ring. The purpose of hybrid is that it combines benefits of different topologies to provide high performance. However it is difficult to design and it requires often more equipment.

I^2C:
I^2C uses simple bus topology. All masters and slaves are connected to the two bi-directional lines, known as Serial Data Line and Serial Clock Line. The typical voltages used are usually +5 or +3.3V. Up to 112 slaves are allowed to connect on the bus while 10-bit addressing allows the use of up to 1024 slaves. But note that the number of interfaces connected to the bus is dependent on bus capacitance, which has a limit of 400 pF [11].

CAN:
The topology which CAN uses is bus where a large number of units can be connected. The number of nodes which the system can support is not specified and it depends on the electrical loads and propagation time, but up to 64 nodes is normal [3].

LIN:
LIN is a one-wire bus which can connect up to 15 slaves [12].

FlexRay:
Three topologies are supported including passive bus, active star and hybrid topologies that are a combination of active star and passive bus configurations. FlexRay also supports simultaneous transmission over a redundant bus or star [7].

MOST:
A MOST system can connect up to 64 nodes. The basic architecture is ring but can also be arranged in a star topology or any hybrid form of ring and star [10].

7 Selection of protocol

To provide a system which matches all desired properties is a very difficult problem because every choice is related to certain aspects and always has its pros and cons. An important task is then to take advantage of the benefits and minimize the impact of disadvantages. After studies of the various protocols, the evaluation results are concluded as follows:

7.1 Summary of the evaluation results

CAN/LIN:
The advantage of the CAN/LIN is the way they deal with disturbances. CAN and LIN provide a high level of error detection and handling by using error frames and retransmissions from CAN and diagnostic message from LIN. Another advantage is its flexibility. It supports multicast because CAN is message-oriented, a unique address for each node is not needed to be specified. More than one module can act on the message and at the same time they can also check for transmission errors. In addition, CAN is built into many microcontrollers that can be easily found in the market which can provide lower cost. However, since the CAN protocol is event-driven, it may have variable message latencies depending on the current bus load. In CAN bus, collisions are also normal and needed to be resolved by prioritized arbitration while on the other hand, it is very suitable for real time prioritized systems like Prismaflex.

Conclusion can be drawn that CAN offers high data security with relatively low cost. Although LIN has a limit data speed of 20kb/s which does not meet the system requirement, the limited speed can avoid problem with interference problems. So based on its cost-effective and simple construction, LIN is appropriate to be sub-buss for CAN.

FlexRay:
FlexRay supports deterministic data transmission because it is predictable, application development can be facilitated. It can be configured for fault-tolerance through its error detection method, redundant channel and bus guardian ability. A high flexibility is also an important property of FlexRay. Communications cycle is divided into static and dynamic part which suits different needs. Moreover, both bus architecture and star architecture can be constructed. On the other hand, a new protocol means that it has not been used very much and is less well-tested compared to other protocols that have since long existed in the market which may require high levels of research for future applications.

Overall, FlexRay is the best solution if the system needs to be fast, deterministic, secure and flexible. It fits very well to time and safety-critical and data intensive applications.

MOST:
Optical fiber that MOST uses not only provides high bandwidth but also protection against electromagnetic disturbances. However, the technology is relatively expensive compared to electrical interfaces. MOST supports high speed and large capacity. But the control signals that dialysis machine uses is usually much smaller load compare with media that MOST is primarily designed for and thus dialysis system cannot take special advantage of these benefits.

7.2 Protocol rating

The following table is estimation for respective protocol value. Value weight is the relative contribution of the factor scaled from 1 to 5 where 5 has the highest weight. Each protocol has an estimated grade (-1 or 0 or 1) for respective field. Protocol value is the overall grade of that protocol by calculating the sum of estimated grade of each factor which multiples with the respective value weight. Note that the cost effectiveness for each protocol has not been investigated in this thesis. It is only a rough comparison from [3 (Part B, Grading in terms of performance versus costs)].

	I^2C	**CAN/LIN**	**FlexRay**	**MOST**	**Value weight**
Cost effectiveness	1	1	0	-1	5
Noise immunity on physical layer	-1	1	1	1	4
Transmission Security	-1	0	1	1	4
Data rate	0	-1	1	1	3
Data capacity	1	-1	0	0	3
Arbitration	0	0	1	1	3
Bus behavior	-1	0	1	1	2
Data transmission efficiency	1	-1	1	0	1
Topology	0	0	1	1	1
Protocol value	-1	2	18	12	26

Table 7.1 Protocol rating

According to the above estimation, FlexRay would be the most promising protocol to be examined. But since this is only an evaluation upon the theoretical perspectives, other protocols can also be interesting alternatives to be examined to find out the practical performances of different protocols.

8 Migration steps

In this chapter, the nature of protocol testing is discussed followed by proposed tests for future research. The main step of migration testing will be to set up a test bed with the help of a computer model since testing of a system can be costly. The system may not work as expected if a prototype is developed and run directly for the first time. With the computer model, data can be easily obtained and problems can be diagnosed quickly and cheaply.

8.1 The nature of protocol testing

Having determined functional and performance objectives for the communication system, the next step is to identify the problems so as to decide which types of tests should be performed. The kinds of questions which must be asked are as follows.

- There are sensors and actuators already in existence which need to be connected. Does the protocol support these existing hardware and software? Must the existing interfaces be used or can new hardware and software be constructed to connect devices to the communication system?

- Are there any new types of applications which will connect to the system in the near future? Does the protocol support these functions?

- How often do errors occur? What are the main types of errors that can occur in the system? How does the system recover from different types of errors and failures?

- When establishing a testing environment, certain cost factors can also be considered. What are the cost for hardware and software needed? Is it easy to find supplier in the market? Does the system cost less than the system it replaces?

8.2 Proposed analysis and tests

8.2.1 Components and system structure analysis

The first step is to analyze all functionalities of the migrated system, i.e. the Prismaflex system should be studied and checked in order to gain the knowledge of how they work in the system.

8.2.2 Unit test

The second step is to connect a device with a computer. As the unpredictability of the system increases with the complexity of the system, the setup should be done individually for each device to test the feasibility of the tested protocol, so called unit test.

A bus adapter may be needed to provide compatibility between a protocol and a device, as shown in figure 8.1(TP is the shortening of Tested Protocol). Although a specialized device of the tested protocol can be used, it may not be the best choice. A bus adapter can in some cases provide lower costs and allow easier upgrades to different platforms and shorten the development time. As an aspect of economy, the existing system should be reused as much as possible so as to reduce the cost. But in the long run, it is better to have devices for specialized protocol because it can provide simplicity in connection and also reduce weight when hardware needed decrease (Figure 8.2). The main function in this step is therefore to investigate and discover different possibilities for the connections.

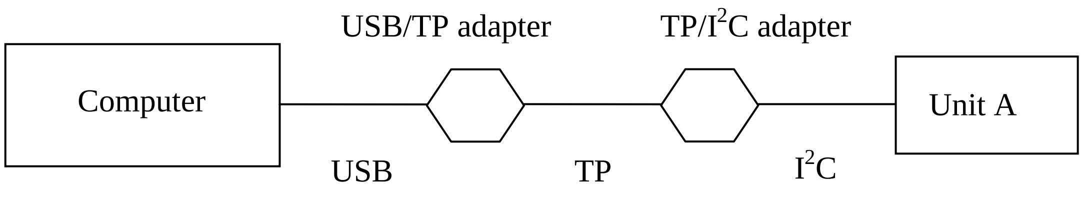


Figure 8.1 Connection between a computer and a unit with the use of a bus adapter

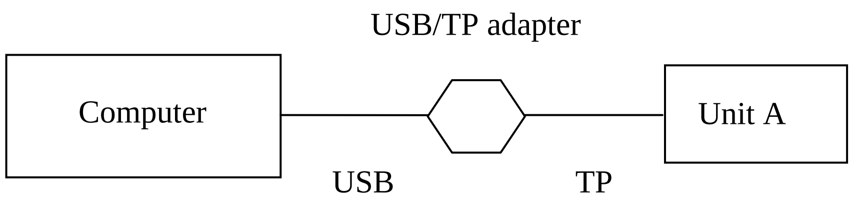
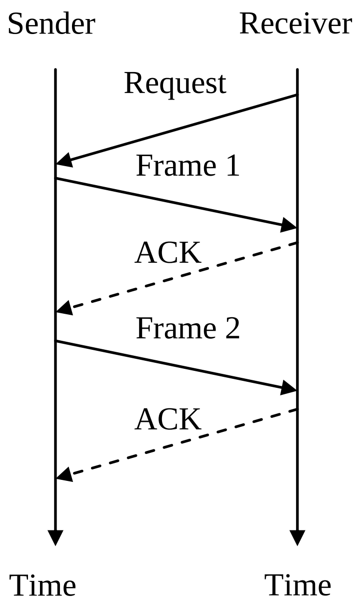


Figure 8.2 Connection between a computer and a unit for specialized protocol

A developing program should be implemented in the computer that can transmit or receive a message from a unit. The flow control can then be investigated. Examples can be found in figure 8.3 and figure 8.4. Afterwards, the message flow can also be tested by transmitting simple commands which are currently used in Prismaflex.



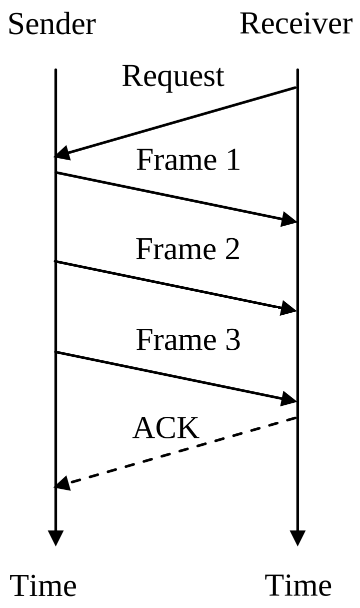


Figure 8.3 The receiver sends an ACK each time it receives a frame

Figure 8.4 The receiver waits and sends an ACK until all frames needed have been received

Error control is another mechanism that can be studied. Error control refers to both error detection and error correction. Frames can be lost or damaged in transmission therefore it is important to define how the protocol handle errors in different situations. Figure 8.5 shows an example of retransmission of a lost frame by using timer.

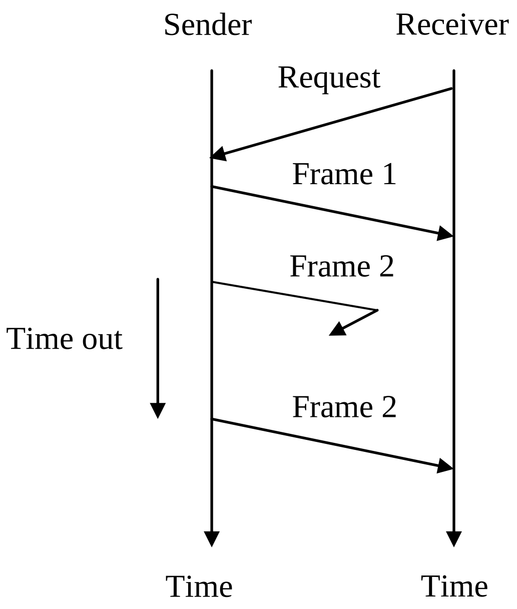


Figure 8.5 Retransmission of a lost frame

8.2.3 System test

In case all devices are proved to be adaptable with the protocol, the next important step would be the redesign of the system architecture. There are a variety ways to re-architect the system, depending on different needs, such as the simplicity, cost or flexibility. The first test can be conducted by using all nodes which are produced for the tested protocol under the condition that all existing nodes can be changed to the tested protocol nodes (Figure 8.6). The second test can be performed with a mix of I^2C nodes and the nodes of the tested protocol to see if all nodes can work together in a system (Figure 8.7). The third test is to investigate the feasibility to have a multiple-bus system where it combines different protocols and different types of nodes in one system. An example of it is shown in figure 8.8.

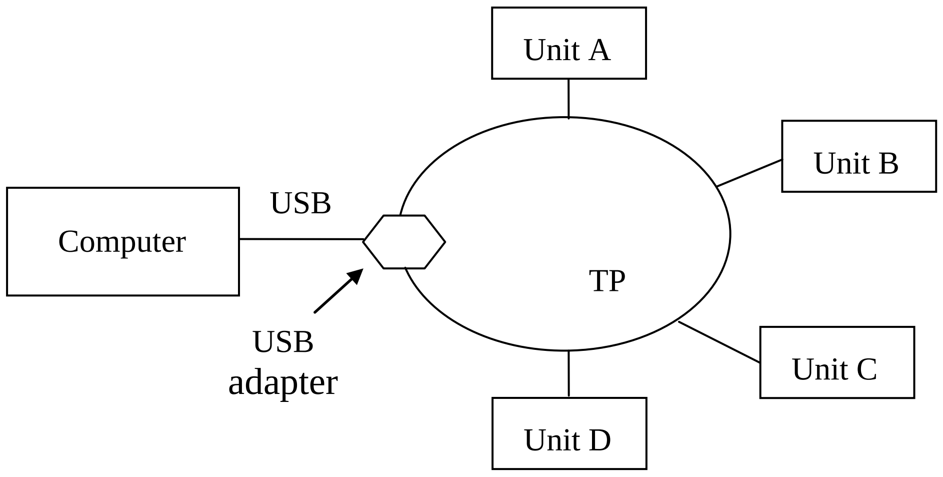


Figure 8.6 All nodes of the tested protocol are constructed to form a system

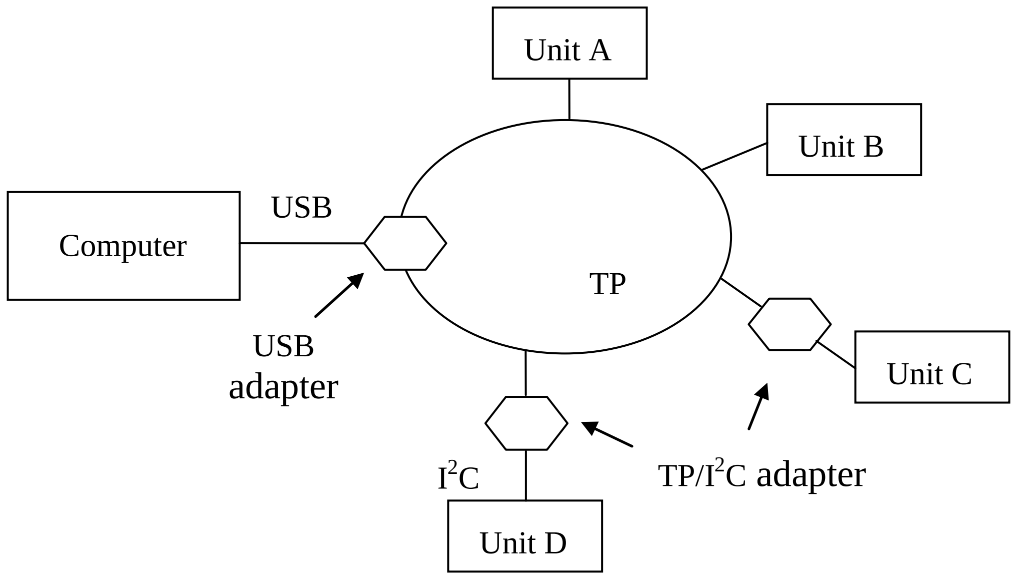


Figure 8.7 System with a combination of I^2C nodes and the tested protocol nodes

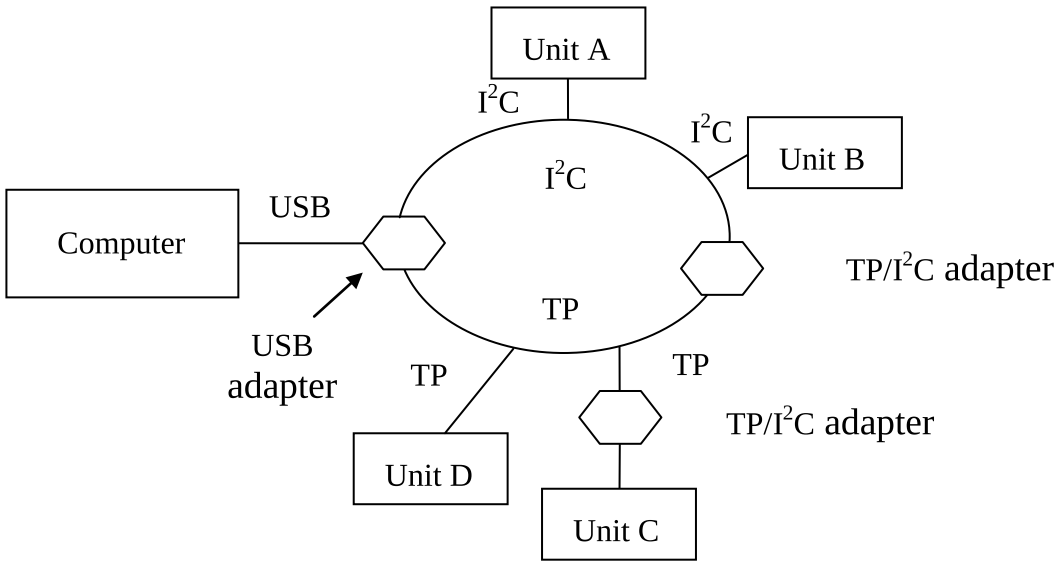


Figure 8.8 System with multiple communication protocols

8.2.4 Performance test

While establishing system with different alternatives, a set of performance tests can be carried out to investigate the practical performance of the system, including delay test, noise immunity test.

Delay test: A too large delay has a negative effect on the system and may cause congestion. When a message is delayed and cannot be received by the receiver on time, the sander may retransmit the packet which may cause even more delay and congestion. So it is important to examine how the delay is related to different busloads.

To calculate the delay and the propagation time, the payload size should be finalized. It can be started by one byte and increased by one byte after each iteration to increase the busloads. Note the actual data rate and the propagation time when transmitting a message between two nodes. The arbitration behavior can meanwhile be studied by allowing two messages to be sent at the same time. Increase the number of nodes which transmit messages and the transmission frequency to study if this would cause delay and seriously slow down the propagation time.

Noise immunity test: In practical conditions, problems such as line impedance, transmitting and receiving equipment, data rate and external noise can distort waveform, causing the voltage to rise and fall exponentially. These extra pulses may produce errors in the receiver by interpreting them as bits that have been transmitted.

This test focuses on the protocol's bus immunity assessment and studies the effect of noises on the system. The main test equipments required include a spectrum analyzer for examining the spectral composition of the electrical waveform and a noise generator and amplifier for generating the noise condition (Figure 8.9). Analyze the waveforms of the data by using analyzer and investigate how the protocol deals with abnormal conditions that are caused by interferences and on the other hand which types of errors can be aroused in case the protocol cannot cope with the interferences. The noise emission shall be enhanced for each measurement.

As the speed of the data is increased, the wave can become even more distorted, the variation of data rate should also be taken into consideration to study the relationship between data rate and waveform distortion.

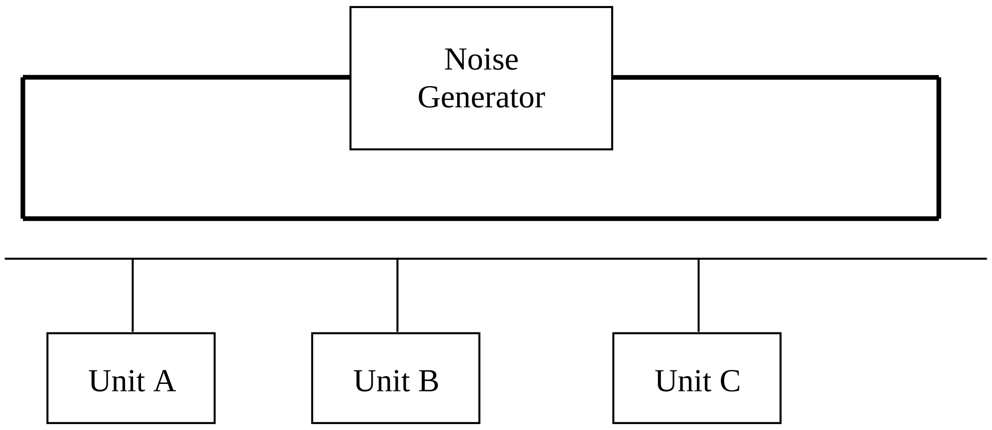


Figure 8.9 Noise immunity test

8.2.5 Cost analysis

The purpose is to discover the economic factors and calculate cost estimation for the migration process. The following points can be interesting to study:

- Materials: Summarize all components which are needed to build up the system. They can be hardware like microcontroller and other necessary peripheral equipments as well as software like development or design program.
- Suppliers: Find out whether the components needed are marketed by many suppliers or not and what the average cost is for each of the components. A protocol that is well-established in the market can effectively reduce the cost and even increase the flexibility.

There are absolutely a lot of other external factors which can be for example the cost of professional associations and equipment maintenance. However, these factors are out of the field of this thesis and therefore it is better to leave this part to other professional researcher.

9 Conclusion

The purpose of this thesis are achieved by answering several concerned questions (See chapter 1.3) which are the study of Prismaflex, the study of different protocols (CAN/LIN, FlexRay and MOST), evaluation of these protocols and recommendation of a testing environment.

Upon all studies and investigations that were done in this thesis (See chapter 6), the four protocols, CAN/LIN, FlexRay and MOST are considered to be appropriate to be the communication protocol for next generation dialysis machines. Of these four protocols, FlexRay got the highest grading and would be the most interesting alternative (See chapter 7.2). MOST got the next highest grading which also could be interesting to study.

Further research in the design of network and testing is absolutely needed in order to learn more about the protocols and obtain accurate performance results for different designs of communication systems. The potential tests that are proposed in this thesis may give some recommendations for future research (See chapter 8). It would be interesting to have the chance to investigate the protocols in a testing environment. Unfortunately, there was no possibility to do this due to lack of time and thus it has positioned itself outside the boundary of this thesis.

10 Terminology

Word	**Explanation**
ACK	Acknowledgement, commonly used in data networking to signify receipt
CC	Cycle count, incremental counter for communication cycles
CRC	Cyclic Redundancy Check, an error-detecting method that a number of check bits are appended to the transmitted message
EOF	End Of Frame, to indicate the end of a message
EOP	End Of Packet, to indicate the end of a packet
LEN	Payload length
MSB	Most Significant Bit, often refer to the left-most bit
PID	Packet identifier
RTR	Remote Transmit Request, to distinguish a data frame and the request of a data frame, always equal to “0” for the data frame (dominant) and “1” for remote frame (recessive).
SOF	Start Of Frame, to indicate the start of a message
SYNC	Synchronization

11 Bibliography / References

11.1 Literature

The literatures chosen are famous books in the data communication field.

[1]Andreas Grzemba (2008) MOST, the Automotive Multimedia Network: Franzis Verlag GmbH

[2]Behrouz A. Forouzan (2006) Data Communications and Networking: McGraw-Hill Companies, Inc.

[3]Dominique Paret (2007) Multiplexed Networks for Embedded Systems: CAN LIN, FlexRay, Safe-By-Wire.... England: John Wiley & Sons Ltd.

[4]John M. McQuillan and Vinto G. Cerf (1978) Tutorial: A Practical View of Computer Communications Protocols: IEEE

[5]Richard Zurawski (2009) Networked embedded systems: CRC Press, Taylor & Francis Group

11.2 Documents

The following documents are the specifications of different protocols and descriptions of Prismaflex machine. They are designed by Gambro and different manufacturers so that right and reliable information can be assured.

[6]BOSCH, CAN Specification version 2.0

[7]FlexRay Consortium, FlexRay Communications System Protocol Specification Version 2.1 Revision A

[8]Gambro, I2C Interface Design Description

[9]Gambro, Prismaflex Operator’s Manual For use with software version 5.xx

[10]MOST Cooperation, MOST Specification Rev 3.0

[11]Philips Semiconductors, The I^{2}C-bus specification Version 2.1

[12]LIN consortium, LIN Specification Package Revision 2.1

[13]USB Implementers Forum, Inc., USB 2.0 Specification

11.3 Internet

The following references from internet were selected because all of them are from reliable organizations.

[14]Dietmar Millinger Technology Consulting, FlexRay Technology, retrieved May 20, 2011
http://www.millinger-consulting.com/dmtc/index.php?option=com_content&view=article&id=3&Itemid=3

[15]EE Herald, Online course on Embedded Systems, retrieved May 20, 2011
http://www.eeherald.com/section/design-guide/

[16]Gambro, about Gambro, retrieved May 26, 2011
http://www.gambro.com/en/sweden/About-Gambro/Gambro-in-brief/

[17]Embedded Systems Academy, General Introduction to I^{2}C from NXP, retrieved May 20, 2011
http://www.esacademy.com/en/library/technical-articles-and-documents/miscellaneous/i2c-bus/

[18]I^{2}C Bus, I^{2}C properties, retrieved May 19, 2011
http://www.i2c-bus.org/

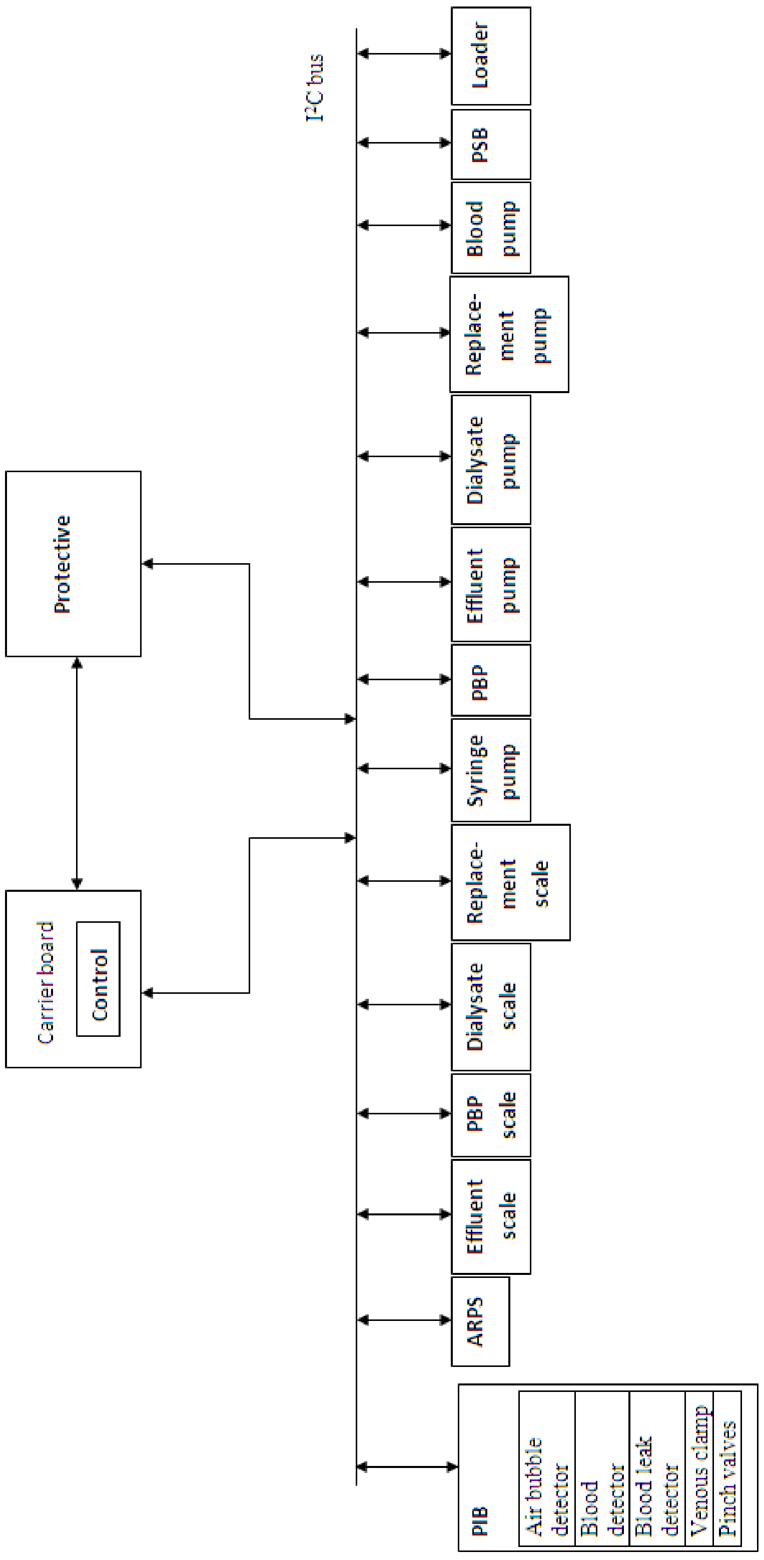
[19]Lammert Bies, Schmitt-trigger circuit tutorial, retrieved May 26,2011
http://www.lammertbies.nl/comm/info/Schmitt-trigger.html

[20]MOST Cooperation, General information about MOST, retrieved April 13, 2011 http://www.mostcooperation.com/home/index.html

[21]National instruments, FlexRay Automotive Communication Bus Overview, retrieved April 13, 2011
ftp://ftp.ni.com/pub/devzone/pdf/tut_3352.pdf

[22]Netrino, Introduction to Controller Area Network (CAN), retrieved May26, 2011
http://www.netrino.com/node/188

12 Appendix A – Prismaflex connection diagram


I^2C bus
Loader
PSB
Blood pump
Replace-ment pump
Dialysate pump
Effluent pump
PBP
Syringe pump
Replace-ment scale
Dialysate scale
PBP scale
Effluent scale
ARPS
PIB
Air bubble detector
Blood detector
Blood leak detector
Venous clamp
Pinch valves
Protective
Carrier board
Control