Bachelor Project

Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Measurement

LIN Slave Simulator

Aleksandra Pereverzeva

Supervisor: doc. Ing. Jiří Novák, Ph.D.
Field of study: Open Informatics
Subfield: Computer Systems
May 2018

Acknowledgements

I would like to thank my supervisor doc. Ing. Jiří Novák, Ph.D. for guidance, advices and passed knowledge.

Declaration

Prohlašuji, že jsem předloženou práci vypracovala samostatně pod dohledem vedoucího, a že jsem uvedla veškerou použitou literaturu.

V Praze, 25. May 2018

Abstract

This bachelor's thesis describes the process of designing and realization of a hardware module, which will be used mostly in automotive industry. The function of the module is simulation of several LIN slaves depending on orders and information received through CAN bus. Then the process of implementation of software follows. The module consists of a processor, LIN, CAN and SWD interfaces. The goal of this thesis is implementation of a working module, which will be able to send data to LIN bus and then change this data on request from CAN bus. The first part of the document focuses on the theoretical knowledge needed to implement and understand the module, such as used technologies. The second part deals with the description of the module itself, used components, connection of the circuits, PCB layers, the explanation of the code parts and its algorithm. The outcome of the work is the design, the module and the program for configuration of the module.

Keywords: LIN, CAN, module, PCB

Supervisor: doc. Ing. Jiří Novák, Ph.D.
Technická 2,
166 27
Praha 6

Abstrakt

Tato bakalářská práce popisuje návrh a proces realizace hardwarového modulu, který se bude používat převážně v automobilovém průmyslu. Hlavní funkcí modulu je simulace několika LIN slavů na základě přijatých požadavků od sběrnice CAN. Dále je popsán postup implementace softwaru. Modul se skládá z procesoru, rozhraní LIN, CAN a SWD. Cílem této práce je implementace funkčního modulu, který by byl schopný odesílat data na sběrnici LIN a pak tato data měnit podle požadavku ze sběrnice CAN. První část této práce se zabývá teoretickými znalostmi, které jsou potřeba vědět při implementaci modulu a pro pochopení jeho fungování, jako například použité technologie. Druhá část se soustředí na popis samotného modulu, použitých součástek, zapojení obvodů, vrstev desky plošných spojů a vysvětlení částí kódu a algoritmů. Výsledkem práce je návrh, samotný modul a programové vybavení pro jeho konfiguraci.

Klíčová slova: LIN, CAN, modul, DPS

Překlad názvu: Simulátor LIN Slave

Contents

Chapter 1

Introduction

Nowadays we can observe, that engineers of automotive industry replace mechanical elements with much more intelligent mechatronic components and systems, which need a way to communicate with each other. This is where networks that are relying on wires come in handy. Usually, they are relatively cheap and fast and sometimes even more reliable than mechanical types of data transferring which makes them even more useful. But what if these wires with data, so-called buses, of different types should send some orders to each other? It seems that additional device would be necessary, a device that can access both networks in some way and influence communication on one bus with orders received from the other bus.

This project is focused on the design of such a device and then its further configuration. The main requirements are the provision of a possibility of the configuration of communication frames on LIN bus via CAN bus, support of transmission of Event-driven frames. It might be vital in case of testing the networks or some of the nodes of the LIN bus break down but their contribution into communication on the LIN bus is needed. As I have already mentioned, this device can be used in automotive industry, such as Škoda Auto for example. Similar devices exist, such as [Emu], but their prices start at $355 (7277 CZK) according to [Phy], so there is also a secondary goal to lower this price.

1.1 Used Abbreviations

- CAN – Controller Area Network
- LIN – Local Interconnect Network
- U(S)ART – Universal (Synchronous) Asynchronous Receiver and Transmitter
- MCU – Microcontroller Unit
- JTAG - Joint Test Action Group
- ASICS – Application-specific integrated circuit
- CRC - Cyclic redundancy check
- PCB – Printed Circuit Board
- L – Electric Induction
- I – Electric Current
- V – Electric Voltage
- SMT – Surface Mount Technology
- TVS - Transient-voltage-suppression diode
- SWD - Serial Wire Debug
- IDE - Integrated Development Environment

Part I

Theoretical part

Chapter 2

Introduction

This part is dedicated to the description of technologies used for fulfilling this project.

The first chapter will focus on the network technologies such as CAN and LIN. This part will contain communication examples of these networks and also general information about UART technology.

The second chapter consists of more abstract conception about power supply, components and LED connections.

This part tempts to achieve the following goal: giving the reader all the needed theoretical knowledge for understanding of the structure of the module, its necessary components and its behavior.

Chapter 3

Used Technologies: Networks

3.1 Media Access Control

Media Access Control (MAC) is a sublayer of a Data Link Layer of the OSI Reference Model. Its protocols also describe the process of controlling the way devices can access the shared network. The nature of MAC can be deterministic and non-deterministic [sCN] [Med].

1. Deterministic Access

   A convention that garantees all stations the right to communicate within a certain time frame according to the priorities assigned by the administrator of the system. In this type of MAC, the collisions are not possible, because all of the communication is centralized.

   a. Master-Slave

      A special node called Master inquires, so called, Slave nodes. Slave nodes can only send responses, they are not allowed to initiate the communication. This type of communication is easy to implement and price for this is a limited number of devices and high dependency on Master node.

   b. TDMA (Time Division Multiple Access) - strictly given time slots, during which devices are allowed to send information to the network.

2. Non-deterministic Access

a. ALOHA - one of the oldest protocol. If a station has something to sends, it starts transmitting; if while transmitting it received another message, a collision has occurred and all stations have to resend the data.

b. CSMA (Carrier Sense Multiple Access) - Equivalent nodes have to wait for media to be free for starting the transmission. Collisions can occur, but are not necessarily detected.

   (i) CSMA/CD (Collision Detection) - all collisions are detected. After the collision nodes wait a random period of time $T_a \cdot d$ to start transmission again, where T_a is a constant and d is a random number.

   (ii) CSMA/CA (Collision Avoidance) - collision may not be detected. After a detection of a free media, a station waits for a random period of time and after that starts transmission. This type is used in wireless networks.

   (iii) CSMA/CR (Collision Resolution) - if a collision was detected, a frame with the highest priority is sent, all other stations have to repeat the transmission. This type of media access is used for CAN bus.

3.2 CAN

3.2.1 General information

CAN is a type of a bus that is usually used in automotive industry for communication between microcontrollers and devices inside a vehicle without a host computer. It is a message based protocol designed for automotive applications but nowadays it is used in several industry branches such as industrial automation and medical equipment. It is so widely spread thanks to independence to media and extremely high reliability even in worst environmental conditions.

3.2.2 History

The bus was invented by the company Robert Bosch GmbH and was presented in 1986; first CAN controller was introduced one year later by Intel and Philips. In 1988 CAN bus was used for the first time (in BMW automobiles of series 8) [CAN].

3.2.3 Communication protocols

The bus has two wires CANH (high) and CANL (low). Theoretical transmission speed is 1.5 Mb/s; however, two physical layer variants provide the maximum speed of 1Mb/s and 125Mb/s respectively. While the bus is not active it exists in the idle or recessive state. In the recessive state of higher speed signals are 2.5 V for both CANH and CANL; for the dominant state of higher speed, CANH has the voltage of 3.5 V, while CANL has 1.5 V according to ISO 11898 [ISOb]. Thanks to this the differential pair of 2 V is created.

Another ISO 11519 [ISOa] shows that for lower speed CANH conducts 1.75 V in recessive and 4 V in the dominant state, for CANL these values are 3.25 V and 4 V respectively. Unlike high speed, Low-speed standard does not require the resistors of 120 Ohm at the end of the linear bus because at this speed signal reflections are not originated. CAN bus can be taken as a compilation of two standards: Standard CAN (version 2.0A), which uses 11-bit identification, and Extended CAN (version 2.0B), which uses 29-bit identification [Zhu10].

There is another CAN protocol called CAN FD (Flex Data-Rate). CAN FD has a different frame structure, which allows larger data flow and a possibility to turn higher bit rate. CAN FD is created in such a way that it can co-exist on the bus with controllers of older standards [Bos12].

3.2.4 Implementation

The implementation requires 120Ohm resistors on both ends of the bus to prevent signal reflection, in other words for impedance matching (can be seen on [3.1]). The value of 120 Ohm corresponds to the line impedance of usually used twisted pair cables. In an ideal situation the number of connected terminals is not limited, but in order to save useful dynamic and static properties, it is recommended to put these limits, for example on 30 terminals [HMS02].

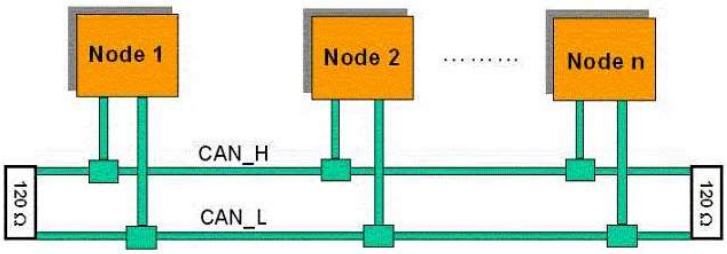


Figure 3.1: Diagram of the CAN bus network [HMS02].

3.2.5 Communication on the bus

In CAN 2.0 messages can be divided into 4 types, the first couple of them is for transmitting and receiving the messages and the second couple is for management of the network. The messages are received by every node that is connected to the bus and is in receive mode. Every message contains an identifier, which shows the content of the message and then is used by the terminals to decide whether the message is determined for this node and should be evaluated or not and should be destroyed. This mechanism is called filtration.

CAN bus uses protocol CSMA/CR for media access. This means that in case a collision takes place when two or more messages are sent at the same time, their priorities are evaluated and only the message with higher priority is sent first.

Existing frames: Data frame, Remote frame (request for data), Error frame, Overload frame. An example of a base frame can be seen on [3.2]

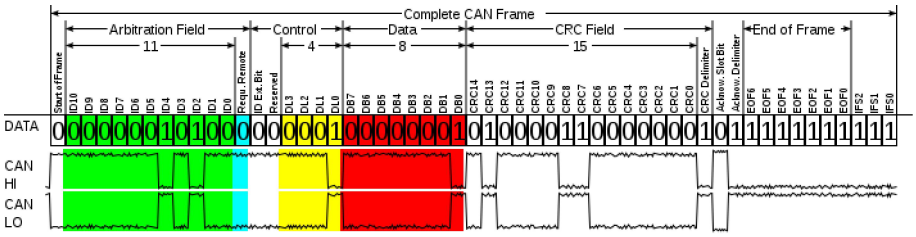


Figure 3.2: Structure of a CAN bus frame [htt].

3.3 Error detection

For error detection it is possible to use one or several of the following most common methods:

1. Monitoring
2. Bit stuffing
3. CRC

Monitoring is a mechanism of an error detection in which the transmitter always compares the data sent and the data on the bus.

Bit stuffing is a mechanism that inserts into the message in the transmitter node an additional bit after every 5 same bits in a row. When the receiver finds out that this bit is missing, it generates an error message.

Another controlling mechanism is the generation of **CRC code**. CRC code is a sequence of values which is generated in a certain way, then added to the message. The receiver after that generates the CRC code of his own and compares it to the one in the message if they differ, an error message is generated[Whab].

3.4 LIN

3.4.1 General information

LIN is a serial network protocol that is used for intercommunication among several components in a vehicle. Appeared as a cheaper but slower substitution of the CAN bus. LIN over DC powerline is standardized as ISO/AWI 17987-8 [ISOe]. LIN supports remote application within a car's network. This bus is an asynchronous bus that uses unbalanced communication among the nodes. The main features are:

- Single master with up to 16 slave nodes.
- The latest LIN specification 2.2 allows a maximum speed of 20 kb/s.
- Operating voltage of 12 V.
- Guaranteed latency time.
- Configuration flexibility.
- Can enable hierarchical networks.
- Detects defective nodes
- Low cost implementation based on standard UART
- Operates with checksum and error detection

The LIN bus specification was designed to use very cheap hardware-nodes within a network. It is a low-cost, single-wire network based on ISO 9141 [ISOd].

LIN main advantages:

- Easy to use
- Cheaper than any other communication buses (is generally 4 times cheaper than CAN bus), in industrial scales it can lower the budget drastically
- Fewer wires than CAN
- Easy to implement extensions
- Protocol license is free

LIN cannot compete or fully replace the CAN bus, only supplement when low costs are vital and bandwidth or speed are not that important. That is why it is mainly used in subsystems that are not critical to vehicle performance or safety. So, in easier words, LIN is generally used instead of the CAN in cases when operations that are needing to be controlled are too simple for the CAN, for example: opening the windows, turning on and off the lights, calibration of the rear-view mirrors, seat positioning, air conditioning and many others [3.3].

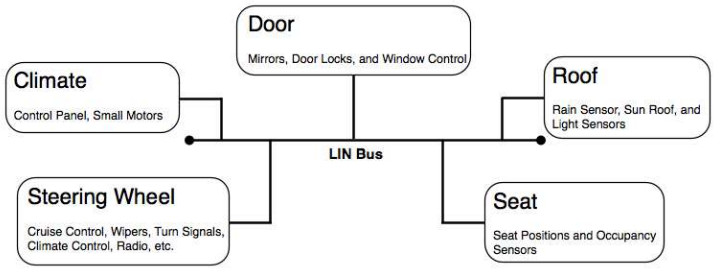


Figure 3.3: LIN functions [Leo].

The example of networks interconnections in a vehicle is shown on [3.4].

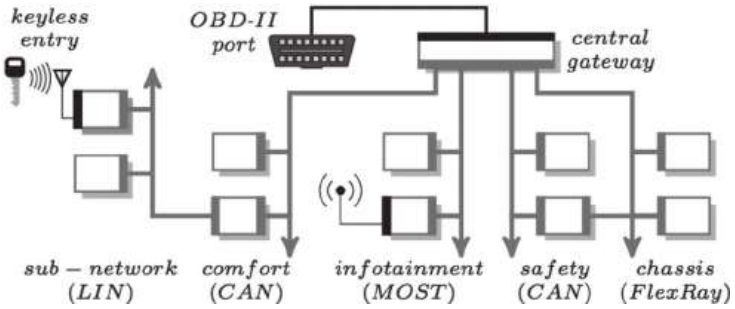


Figure 3.4: Networks in a vehicle [Sch].

Widely used LIN is usually combined with simple sensors for creation of local networks; then they are very often connected by so-called back-bone-networks, such as CAN in vehicles.

3.4.2 History

The first fully implemented version of the LIN specification was LIN 1.3 and was published in 2002. In 2003 version 2.0 was introduced to widen the spectrum of capabilities and provide additional diagnostic features. The latest LIN specification 2.2A is now a standard ISO 17987 [ISOc].

3.4.3 Communication on the bus

The bus has three modes: slow, medium and fast. The realization of the LIN node is relatively simple; besides it does not require a crystal oscillator, an RC oscillator is enough, that can reduce the price [DEK+01].

All communication is initiated by the master. Due to this fact, collision detection mechanism is not necessary. Only one slave can reply to the given message because at the beginning of the frame there is always an identifier of the message, which eventually determines to which slave node this message is intended. Master can reply to its own messages. For the baud-rate stability within one frame there is a special SYNC field in the header of the frame [12].

The role of slaves and a master are normally performed by microcontrollers with UART capability. But in some cases, when low cost, little space, and low power are extremely important, it can be implemented in specialized hardware or ASICs [Mos08].

3.4.4 LIN frames

Content of the frames is the following:

1. Synchronization break
2. Synchronization byte
3. Identifier byte
4. Data bytes
5. Checksum byte

Types of frames:

1. **Unconditional frames** [identifier 0 – 59 (0x00 to 0x3b)] carry signals to all the subscribers. If no errors are found, they can be made available to the application. Example of usage of this frame can be found on [3.5].

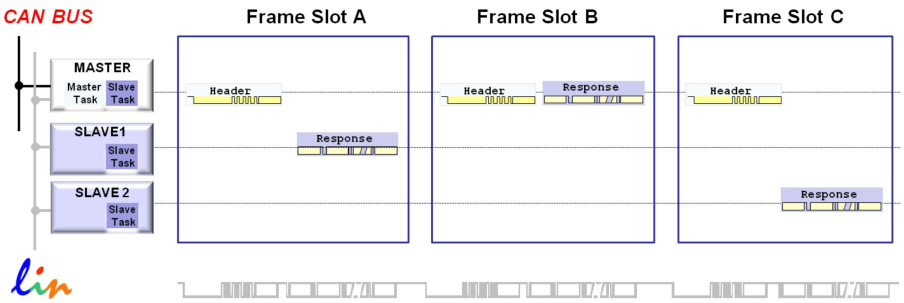


Figure 3.5: Unconditional frame example [Mic].

2. **Event-triggered frames** [identifier 0 – 59 (0x00 to 0x3b)] its purpose or usage is not part of the LIN specification. Can carry any kind of information. Example of usage this frame can be found on [3.6]. This example shows the case when two slaves have something to report, collision happens and master has to resolve this issue.

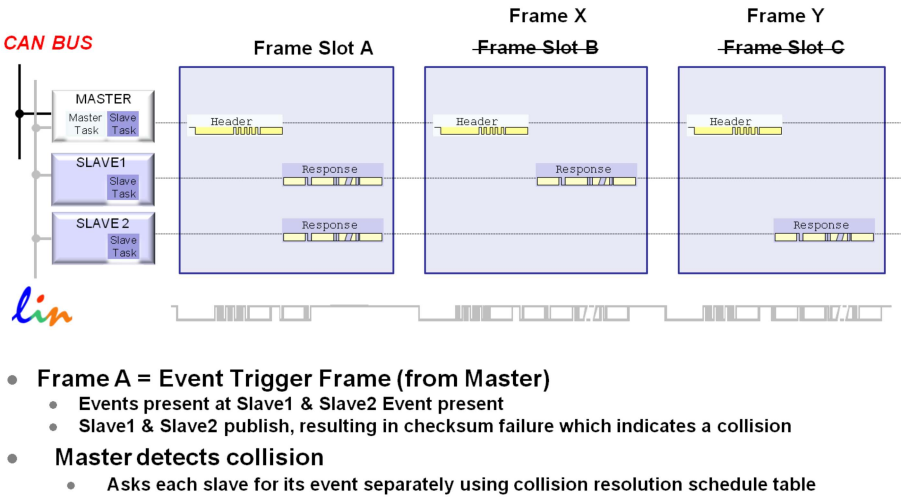


Figure 3.6: Event-triggered frame example [Mic].

3. **Sporadic frame** [identifier 0 – 59 (0x00 to 0x3b)] their purpose is to make the behavior of network deterministic. Example of usage of this frame can be found on [3.7].

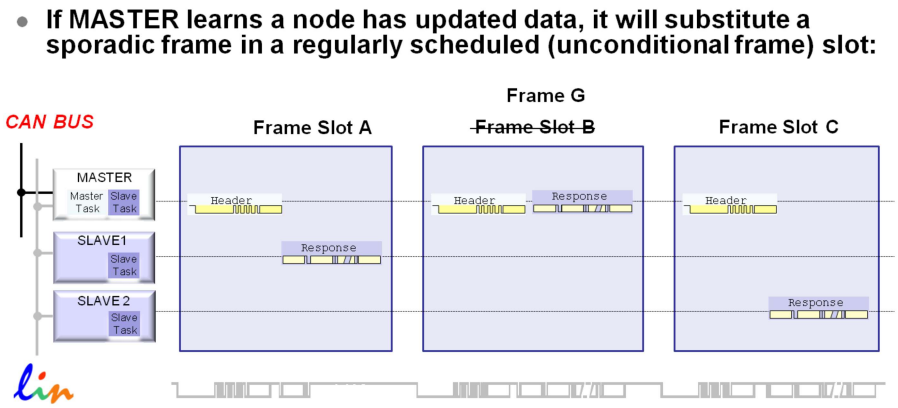


Figure 3.7: Sporadic frame example [Mic].

4. **Diagnostic frame** [master request is 60 (0x3C); slave response is 0x3D] carry diagnostic and configuration data. These frames are 8 bytes long.

5. **User-defined frame** [identifier 62 (0x3e)] can carry any kind of information.

6. **Reserved frame** [identifier 63 (0x3f)] shall not be used

The predefined scheduling tables that are used by the master node to start transmitting and receiving the messages on the LIN bus. LIN frame consists of the two parts: header which is always sent by master and the response sent usually by slave. Data is transmitted as eight-bit data bytes with one start bit, one stop bit and no parity. Bit rates are not strictly given and can vary depending on version, mode and configuration between 1 and 20 Kbit/s. Logical high is recessive and logical low is dominant [Ruf02].

Two bus states are used: sleep-mode and active mode. For normal operation all nodes must be active, but after a specified time the nodes go back into Sleep-mode. If any node still needs communication trough the bus, it sends a WAKEUP frame. Only if all nodes are awake, Master-node can proceed with scheduling the next identifier. The example of functioning of the WAKEUP frame is shown on [3.8].

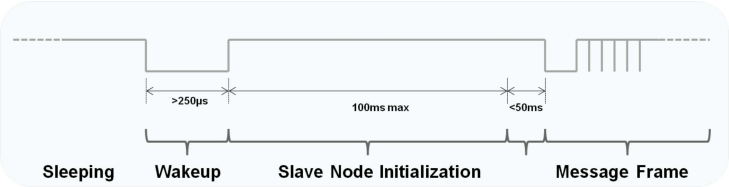


Figure 3.8: WAKEUP frame [Mic].

3.4.5 Header of the frame

Header of the LIN frame, that is sent by Master-node consists of five parts [Lup04]:

1. **BREAK** (one start bit, several dominant bits; min. 11-bit long, usually 13-bit long) prepares nodes to listen to the content of the following header.
2. **SYNC** (0x55) used by nodes for recalculation of the internal baud rate.
3. **INTER BYTE SPACE** (is optional, but if used, all the slaves must know about it) is used to adjust for jitter in the bus.
4. **IDENTIFIER** determines one action which should be performed by one or more nodes. The faultless functionality is ensured by the network designer.
5. **RESPONSE SPACE** – time between identifier and a response.

3.4.6 Response

Response sent by Slave-node or Master itself [Lup04]:

1. DATA - 0-8 bytes of data.
2. CHECKSUM – can be of two types:
   a. Checksum calculation includes data bytes only (in LIN 1.3)
   b. Checksum calculation includes both data bytes and the identifier byte (in LIN 2.0 and newer)

3.5 UART

3.5.1 General information

UART is a hard-integrated circuit for serial communication through the serial port. There are UART devices as standalone IC or as a part of microcontrollers. MCU modules, unlike standalone modules, can control the output of UART. Conversion to the physical signal output can be generated differently on the transmitter side depending on the standard. There are three of these recommended standards: RS-232, RS-422, RS-485. All of them define different methods for physical signal generation on the output. They can be different from MCU's TTL class of digital circuit.

For communication between MCU's UART module and a PC peripheral device, an additional circuit is needed. It is necessary because of differences in transmission methods. An example can be useful: In TTL logic voltages on the UART output can vary from 0 V (logical 0) to Vcc (logical 1). However, in case of RS-232 logical 0 is represented by a voltage in a range of 3-25 V and logical 1 is in a range between -3 and -25. But from the software point of view, MCU's UART module and PC peripheral device are the same.

3.5.2 Types of UART

There are two forms of UART: UART and USART.

1. The asynchronous form of communication:

   The transmitter generates the data clock internally. It is dependent on the MCU clock cycles. Transmitter and receiver (two modules) should work on the same baud rate (have the same clock cycle length). The data is usually transmitted in a byte.

2. The synchronous form of communication:

   The transmitter generates a clock used to let the receiver side to recover data from the stream without transmitter's baud rate. When the clock signal is separated from data and sent on another line, very high transfer rates can be achieved. Maximum possible transfer speed is 4 000 000 bits per second, which is much greater than usual UART speeds. The

data is normally transferred in blocks. Thanks to this, USART module can generate data in a form comprehensive to many standard protocols including LIN.

3.5.3 Communication

Communication happens on two independent lines TX (transmitter) and RX (receiver) in a form of simplex, half duplex or full duplex. The lines must be pulled up, so each side recognizes the existence of connection on the other side. Data frame's lengths can vary depending on configuration.

A frame consists of four parts:

1. Start bit (logic 0)
2. Data bits (1-14 bits)
3. The parity bit (optional), can be even or odd
4. Stop bits (1 or 2, logic 1)

Baud rate is a vital value, which specifies how fast the data is sent over the bus in bits per second. The most used value is a standard of 9600. The complete standard is called 9600 8N1, which means that the bus uses 9600 baud rate, 8 bits of data, No parity and 1 stop bit.

3.6 JTAG

1149.1 IEEE [IEEa] standard provides testing of integrated circuits. JTAG consists of several instruction modes. The most commonly used one is USERCODE instruction, which provides upload of the code into the integrated circuit. Also, is commonly used for so-called Boundary Scan, which is a test that checks the correct functioning of all the pins on the chip. Another usage of JTAG is debugging of the controller.

JTAG has four pins: TCK (test clock), TMS (test mode select), TDI (test data input), TDO (test data output). Communication takes place on the pins TDI and TDO.

However, another version of this standard exists. IEEE 1149.7 [IEEb] standard can be controlled using only two pins - TCK as a clock signal and TMS as a data line.

Nowadays it is a vital interface for microcontrollers.

3.7 SWD

Another debugging tool designed and created specifically for ARM core processors. Its main advantage is the fact that it requires only two wires: SWCLK (clock signal) and SWDIO (input and output). Besides SWD has higher speed of communication comparing to JTAG.

Chapter 4

Used Technologies: Circuits, Components and Devices

4.1 Power supply

Electronic devices can be powered in different ways. One way is using a battery. Another one is auxiliary circuit, which can transform one form of energy (solar, wind etc.) into electricity. Or transformations from mains electricity can be used.

Types of power supplies:

1. Linear regulators;
2. Switched-mode power supply.

4.1.1 Linear regulators

Linear regulators are relatively cheap and easy to implement. They provide high stabilization of output voltage and wide range of input voltage. The disadvantages of such a method of power supply is a low efficiency.

4.1.2 Switched-mode power supply

Such power supply has up to 95-percent efficiency. It is an electronic circuit that converts power by turning on and off special devices at high frequencies. There can be step-up(Boost) and step-down(Buck) converters. Proceeding part shows only step-down converter since only it is used in the project.

4.1.3 Step-down converters

The behavior of the circuit can be divided into two parts. First it is switched on by the S switch. The current steps up on the inductor according to the formula [4.1].

$$V_L = L \cdot \frac{dI}{dt} \Longrightarrow (V_i - V_o) \cdot \frac{t_1}{L} = dI. \tag{4.1}$$

The output capacitor is charged with i_c current. The scheme can be seen on the picture [4.1].

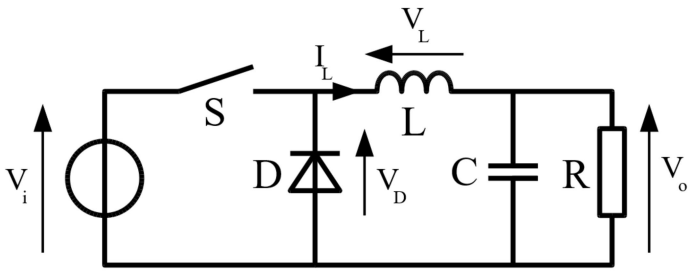


Figure 4.1: Step-down converter [com].

After S is switched off the inductor is trying to save the size and the direction of the current.The currents at the beginning and now are the same. Now the current is flowing through the diode D:

$$dI_Lon = dI_Loff. \tag{4.2}$$

Then the current on the inductor goes down:

$$-V_o \cdot \frac{t_2}{L} = -dI_Loff. \tag{4.3}$$

After combining all the three preceding equations the following equations can be obtained:

$$V_o \cdot t_2 = (V_i - V_o) \cdot t_1 = V_i \left[\frac{t_1}{t_1 + t_2}\right] = V_i \left[\frac{t_1}{T}\right]. \tag{4.4}$$

This gives the output current [KRE97]:

$$I_o = I_i \cdot \frac{V_i}{V_o}. \tag{4.5}$$

4.2 Types of components and packages

The modern PCBs commonly use SMT components due to their better properties and the fact that they do not require drilling of the PCB.

4.2.1 Quad Flat Package

In this type of the case the pins are located on all four sides of the component. The number of pins vary from 32 to 304. This type is used for medium efficient processors, for example STM32. It is possible to solder this case to the PCB in domestic conditions. The most commonly used cases of this type are TQFP, QFN, PLCC.

4.2.2 Dual in-line package

The pins are located on the two sides of the case. Are the best solution for simpler micro-controllers, operational amplifiers. The common space between the pins is 50 mils (1.27 mm). The most frequently used are SOIC, SOT, TSOP.

4.2.3 Ball Grid Array

The package is used for more efficient processors with large numbers of pins organized into matrices on the bottom side of the case. The advantage of such a package is the fact that it lies directly on the PCB and sometimes it does not require soldering and thanks to that it does not overheat. It also has less induction (in comparison to QFP, for example) that is why it is used for faster circuits, for example DDR SDRAM. The main disadvantage is the fact that it requires machine mounting, which can be a problem for projects in domestic conditions.

4.3 LED connection

LEDs are connected to the processor via resistors. The value of resistors depend completely on the parameters of the LEDs.The first important value is the LED voltage V_{LED}, another one in current on the LED I_{LED}. Then according to the Ohm's law the wanted value of the resistor that is needed can be calculated according to the formula:

$$R = \frac{V_{cc} - V_{LED}}{I_{LED}}. \tag{4.6}$$

Part II

Practical part

Chapter 5

Design of the module

5.1 Introduction

The module consists mainly of the processor, which is the heart and brain of the whole module. There are two LIN transceiver components connected to the processor, a CAN transceiver, SWD.

All the schematics were made in OrCAD Capture, a PCB layout in Allegro by Cadence. This software is available in school laboratories. It is a very powerful, though not easy to get to know tool for designing schematics and PCBs.

5.1.1 Hardware Requirements

The requirements to the hardware were the following:

1. The module should have 2 LIN bus interfaces.
2. Configuration trough CAN bus.
3. Allowed power supply of 12 V.

4. Hardware opportunity to set the CAN node address for some value.
5. Small size of the device (recommended dimensions of 10 cm x 10 cm x 4 cm as length x with x hight).
6. Low prices of unit components and their accessibility.

5.1.2 Proposal of the solution

1. This can be solved by translating LIN signals into USART communication. This will require additional components, but requirements to the processor are reduced to two USART interfaces which are more common.
2. CAN interface on the processor is required.
3. The module will use step-down power converter so that the allowed working voltage can be 3.3 V as well as 12 V.
4. For CAN node address settings will be used a DIP switch which is reliable (saves the set value even after the power cut) and easy to use.

5.1.3 Bypass Capacitors

In all circuit parts bypass capacitors are used for reducing the effect noise of other circuit elements can have on the protected component. These capacitors remove the AC caused by ripple voltage that can be present on DC signal. The simple rule for choosing the bypass capacitor is that it should be at least 1/10 of the resistance to the flow of current than what the resistor can offer for the frequency signal that needs to be bypassed. It needs to be put as close as possible to the V_{cc} pin to lower as much as possible the inductance of the wire (information is from [Whaa].

5.1.4 Scheme and Components

The block scheme of the module is shown on the [5.1].

Bill Of Mat	Page1					
Item	Quantity	Reference	Part	FootPrint	Buy	Type
1	1	CON1v	ARKZ950h_02/5.08mm	zt_arkz950h_02_5o08mm-a		
2	1	CON1	ARK1550H2STL-E/3.5MM	zt_ark1550h_02_3o50mm		
3	1	CON5	ST-LINK	zt_psh02-04pg		
4	3	CON6,CON	CAN_9_Z_90	zt_can9z	gme:ZBO_801-041	
5	1	C1	220MM/50V	zs_ce_g_cy400x400z480_p240x80	EEEFK1H221P - panasonic z TME	
6	1	C4	AVX_47M/6V3.B	zs_ce_tantal_b	tme:AVX TAJB476K006R	
7	2	C9,C10	22p	zs_0805c		
8	8	C11,C12,C1	100n	zs_0805c		
9	2	C23,C24	4,7uF	zs_0805c		
10	1	C25	4n7	zs_0805c		
11	1	D2	SK24A	zs_di220re180x120_do214ac	fa:1299275	
12	1	D3	SMAJ36A-E3/61	zs_di220re180x120_do214ac	tme:SMAJ36A-E3/61	
13	2	FD1,FD2	FID_TYPE1	zs_fiducial_type1		
14	1	F1	SMD050F/TEconnectivity/	zs_290x214z125_p90x90	tme:bsmdp-0.5a	
15	3	IC1,IC2,IC3	Value	SOIC127P600X175-8N		
16	4	JP1,JP5,JP	SW_LINE_4l	zt_s1g02		
17	1	LE1	OSRP0603C1C	zs_0606d2x	tme:OSRP0603C1C	OSRP0603C1C
18	3	LE2a,LE3a,	YE150/3.6mA	zs_0603d	fa:2426217	kph-1608syck
19	3	LE2b,LE3b,	RE230/5.9mA	zs_0603d	fa:2146432	kph-1608surck
20	1	R1	5M1	zs_0805r		
21	14	R2,R3,R4,R	10k	zs_0805r		
22	1	R5	560R	zs_0805r		
23	4	R6,R8,R11,	270R	zs_0805r		
24	3	R7,R9,R12	390R	zs_0805r		
25	2	R21,R22	68R	zs_0805r		
26	1	S1	ADE0804	SW_ADE0804		
27	1	U1	TSR-1-2433	zt_dcdc-tsr-1	fa:1696320, tme:TSR 1-2450	
28	1	U3	STM32F105R8_LQFP64	zs_quad_0o50mm_64_wg12mm	fa:1737131	
29	1	Y1	8MHz	zs_hc49s	fa:2449400,tme:8.00M-SMDHC49R	

Figure 5.2: Bill of material.

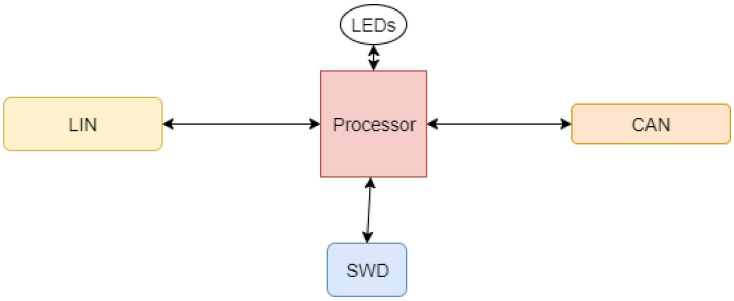


Figure 5.1: Block scheme of the module.

All the used components were thoroughly selected according to their characteristics and features. The whole list can be found in Bill of Material file (.BOM). On the picture [5.2] the list of components with description can be seen.

The connection of the module is based on the CAN GW II project by ČVUT FEL v Praze Katedra měření, 2016.

5.2 The Processor

The processor STM32F105xx [Con] was used thanks to its wide range of useful features, simplicity in testing, its compatibility with a variety of common peripheries, technical support in Prague. Besides that, this processor has an ARM based architecture with 32-bit Cortex-M3 core, which I was well acquainted with on the lectures in CTU. As I have already mentioned, this microcontroller is well balanced in means of functionality and simplicity. The block schematic of this processor is shown on the [1].

The main features of this microcontroller are:

1. 72 MHz maximum frequency;
2. 256 KB of 64-bit Flash memory; SRAM 64 Kbytes;
3. 14 communication interfaces:
   a. 2 CAN interfaces (2.0B Active);
   b. 3 SPI;
   c. USB 2.0;
   d. 10/100 Ethernet;
   e. 5 USARTS;
4. CRC calculation unit;
5. Debug mode;
6. Low power.

This of course led to meeting of all the requirements that were put on the controlling of the module.

There is a connection of the processor on [2]

There is an external oscillator for higher precision. Also, a connection of DIP switch can be observed. It will be used later for changing between 8 possible modes or signals on the LIN bus.

The connection of the Reset circuit is the following: [5.3].

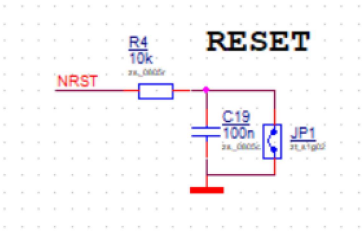


Figure 5.3: Reset.

To prevent spurious resets, it is necessary to connect the NRST pin to a capacitor.

5.3 Power converter

It was necessary for this module to work from 12 V, however the microcontroller works from 3.3 V, so it was necessary to add a step-down power supply. For this goal was used TSR 1-2433 converter, which can transform the range of 4.75 – 36 VDC into 3.3 VDC. To align the voltage was used Shottky Barrier Rectifier along with polymer fuse and TVS diode for protection from voltage spikes. The connection can be found on [5.4].

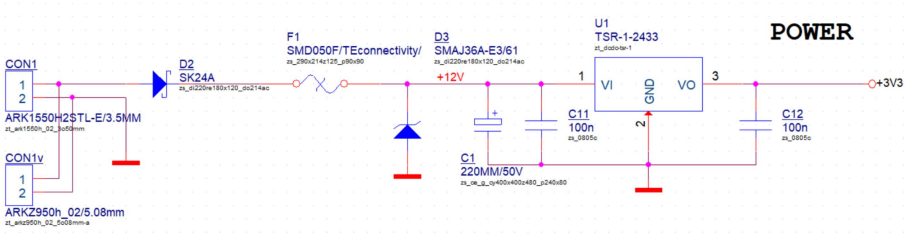


Figure 5.4: Power converter.

5.4 CAN transceiver

The device used for that is powered by 3.3 V.

On CAN circuit part there are used two termination resistors for filtering and stabilization of the common mode voltage since split termination “im-

proves the electromagnetic emissions behavior of the network by eliminating fluctuations in the bus common mode voltages" [SN6].

CAN power supply also has a bypass capacitor. The exact connection can be seen on the [5.5].

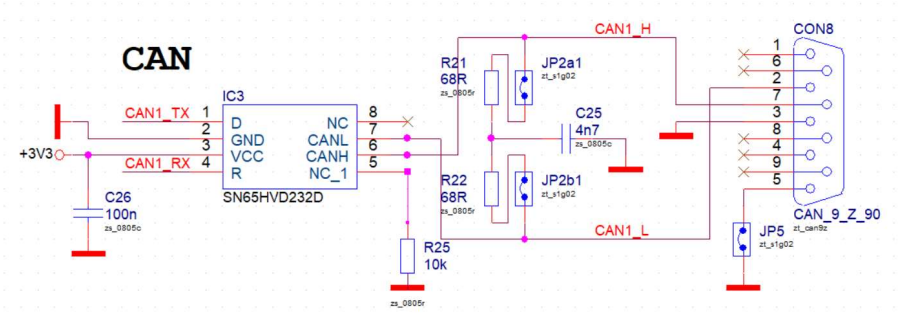


Figure 5.5: CAN transceiver.

5.5 LIN UART converter

For this conversion was used a device TPIC 1021 [TPI]. This circuit has power supply of +12 V, a bypass capacitor is also connected. As recommended in the data sheet, the receiver wire is pulled up on + 3.3 V to eliminate uncertain state on the wire. The closer look on schematic can be taken on [5.6].

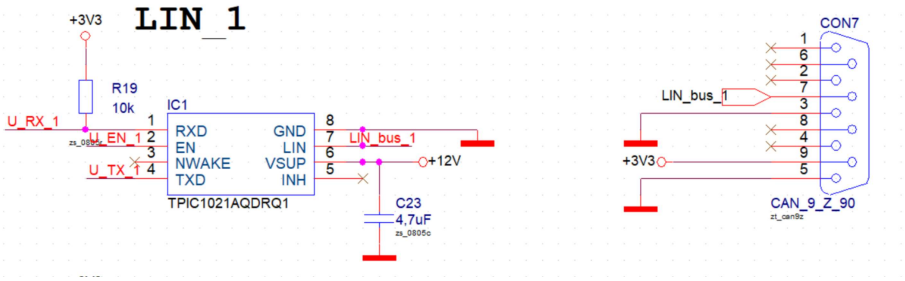


Figure 5.6: LIN UART converter.

5.6 SWD

The connection of the debugging unit can be found on the [5.7]

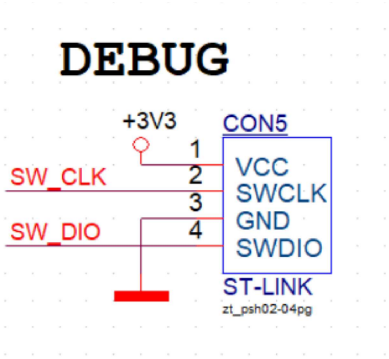


Figure 5.7: Debugger.

5.7 LEDs

The connection of the LEDs is the following: [5.8].

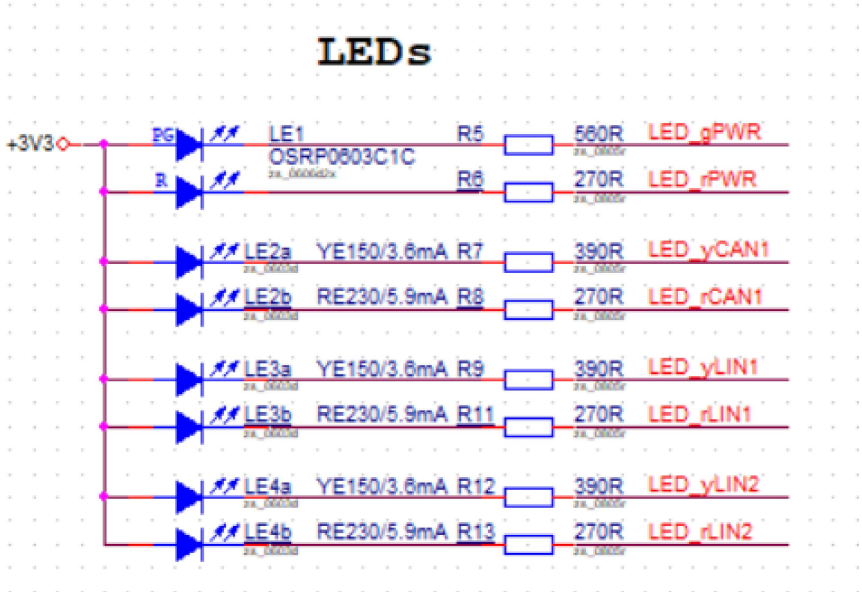


Figure 5.8: LEDs.

The values of the resistors were calculated according to the formula [(4.6)].

The PCB design was made in the Cadence Allegro software. The outut can be seen on the [3] and [4].

Chapter 6

Program

The whole program consists of one loop, which checks, if any of three peripheries has a received message pending. The intuitive flow of a program is shown on [6.1].

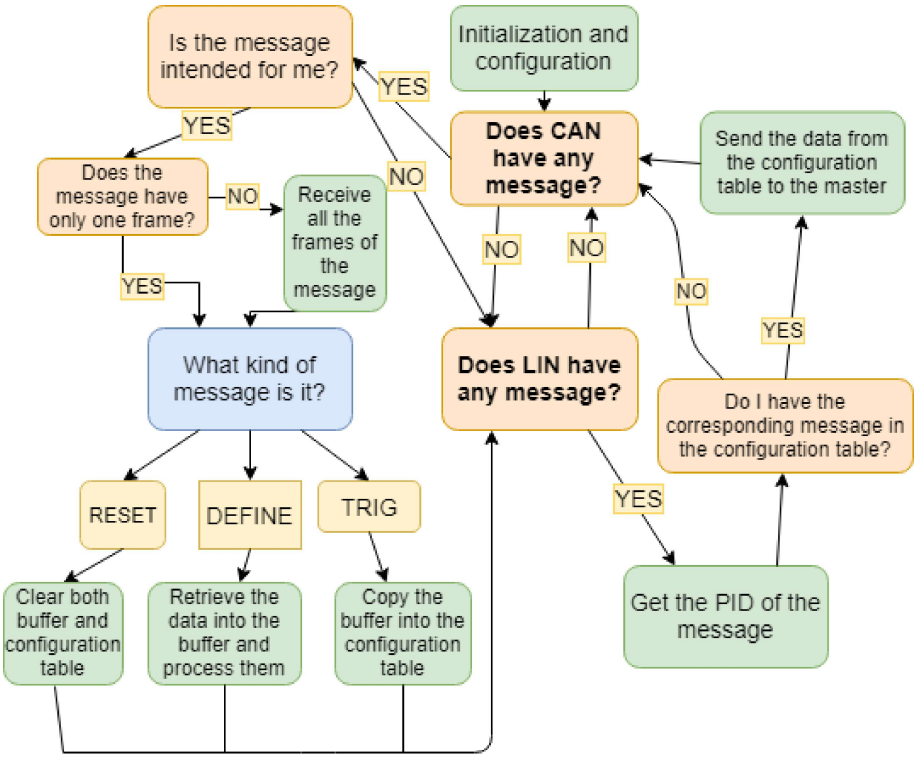


Figure 6.1: Flow of a program.

The program uses polling instead of interrupts due to the fact that its simple, well arranged and easy on the eye. The other advantage of polling is that it uses a state machine and eventually is faster than interrupt mode according to the [Arc].

According to the thesis specification the main requirements are:

- Configuration of incoming and outgoing messages;
- Possibility to select various IDs of the message, the length of the data in the message, checksum version and the content of the data including Error Response bit;
- Sending of the Event-triggered frames.

Pseudocode, which shows functionality and structure of the program can be seen.

```
while(1){
  CAN();
  LIN();
}
  CAN(){
    if(CAN has message){
      receive_data();
      CMD = retreive_command();
      process_can(CMD);
    }
  }

  process_can(CMD){
    switch(CMD){
      case RESET:
        annul_tables();
      case DEFINE_ETF:
        retrieve_data();
        if (last_frame){
          insert_item_into_buffer_table();
        }
      case DEFINE:
        retrieve_data();
        if (last_frame){
          insert_item_into_buffer_table();
        }
      case TRIG:
        swap_buffer_and_conf_table();
    }
  }

  LIN(){
```

```
33    switch(state){
34      case NO_MESSAGE :
35        if(lin_break){
36          state = 1;
37        }
38      case BREAK_RECEIVED :
39        if (LIN has message){
40          receive_message();
41          if(message = sync){
42            state = 2;
43          } else {state = 0}
44        }
45      case SYNC_RECEIVED :
46        if(LIN has message){
47          receive_message();
48          retreive_PID();
49          state = 3;
50        }
51      case DATA_FOUND :
52        if(find_PID_in_conf_table() == SUCCESS){
53          send_item();
54        }
55      case DATA_SEND:
56        send_data_checksum_byte_by_byte();
57    }
58  }
```

The code is loosely based on the CAN_BREAK project by Jiri Blecha, 2012.

The messages that come through CAN bus have the following content of the data bytes as shown on the [6.1].

CMD	$param\_1$	$param\_n$

Table 6.1: Format of the configuration messages.

The CMD byte has in the higher four bits the type of the command and in the last two four contain the descending serial number of the physical frame, because the physical frame of the CAN message always contains only 8 bytes of data. The commands can be seen on the [6.2].

After the last frame of the message was received, the modules sends a response: CMD(four higher bits of the command module responses, four lower bits are zeros) + PAR (parameter [0x00 - positive acknowledgement; 0x01 - negative acknoledgement (uknown command)]).

Command code	Command name	Command description
0x00	Reset	Resets configuration tables and returns the module into the initial state
0x01	Define	Defines the simulated LIN frame
0x02	ETF Define	Defines the simulated Event-Triggered frame
0x03	Trig	Request for activation of the configured functions

Table 6.2: Configuration commands from CAN.

RESET command has no parameters.

Define command has the following parameters:

- **ID** - [1B] - $IIIIII00$.

  Simulated 6-bit frame identifier, its 2 higher bits are zeros.

- **Type/Length** - [1B] - $0_b t000lll$.

  The t bit defines if the frame uses the classic checksum (0 - for LIN specification 1.3) or the enhanced checksum (1 - for LIN specification 2.0 and newer).

  $llll$ bits define the requested length of the simulated LIN frame.

- **Data** - [$llll$B].

  Data of the simulated LIN frame number of bytes corresponds to the $llll$ bits.

- **ErrResp** - [1B] - $0 - 63; 128$.

  The position of the ErrorResponse bit in the data, 128 represents the absence of the ErrorResponse bit in the frame.

Define ETF command has the following parameters:

- **ID** - [1B] - $IIIIII00$.

  Simulated 6-bit frame identifier, its 2 higher bits are zeros.

- **Alias ID** - [1B] - $EEEEEE00$.

  Event-Triggered frame ID.

- **Type/Length** - [1B] - $0_b t000lll$.

  The t bit defines if the frame uses the classic checksum (0 - for LIN specification 1.3) or the enhanced checksum (1 - for LIN specification 2.0 and newer).

  $llll$ bits define the requested length of the simulated LIN frame.

- **Data** - [$llll$B].

  Data of the simulated LIN frame number of bytes corresponds to the $llll$ bits.

- **ErrResp** - [1B] - $0 - 63; 128$.

  The position of the ErrorResponse bit in the data, 128 represents the absence of the ErrorResponse bit in the frame.

More information on this can be found in document LIN Slave, ČVUT FEL v Praze Katedra měření, 2018.

Chapter 7

Conclusion

The main aim of this bachelor thesis was designing and realization of the hardware module and implementing the software for it. The requirements to the module were two LIN and one CAN connectors, the module shoud be able to simulate the behavior of LIN slaves according to the configuration messages from CAN bus.

To fulfill this, I suggested the connection of all the components (wiring diagram). After it was confirmed, I made the design of the PCB. After the PCB was printed out and all the components were soldered to it by my supervisor, the module was tested. During the testing of the module it was found out, that I made some minor mistakes in the wiring diagram, which were solved by my supervisor.

The output module has two layers PCB and equipped with the required peripheries (LIN, CAN, SWD). The size of the PCB is saved within desired dimensions and all the components are optimized according to their functionality, size and price. It uses the power of 12 V. This meets all the requirements introduced in the specification of the thesis.

The module can recognize the input data from CAN bus, process it and save to the configuration table. It also can react to the headers that come from the LIN bus by sending the corresponding items from the configuration table, both unconditional and Event-Triggered. The module responds to found errors in the headers by setting the Error Response bit. This allows monitor problems on the LIN bus. This allows to meet all the software requirements to module.

It is supposed, that in the future, the module will be able to simulate more than one LIN slave (at least two). This is possible due to additional connector on the module and the fact, that the code structure allows to use two USARTs with only minor changes in the main loop.

Working in the Cadence IDE made a great contribution to my skills, because I have never designed a PCB before. I also learned how to work with documentation, found new things for myself during writing of the code and had a deeper look on the hardware implementation.

Appendices

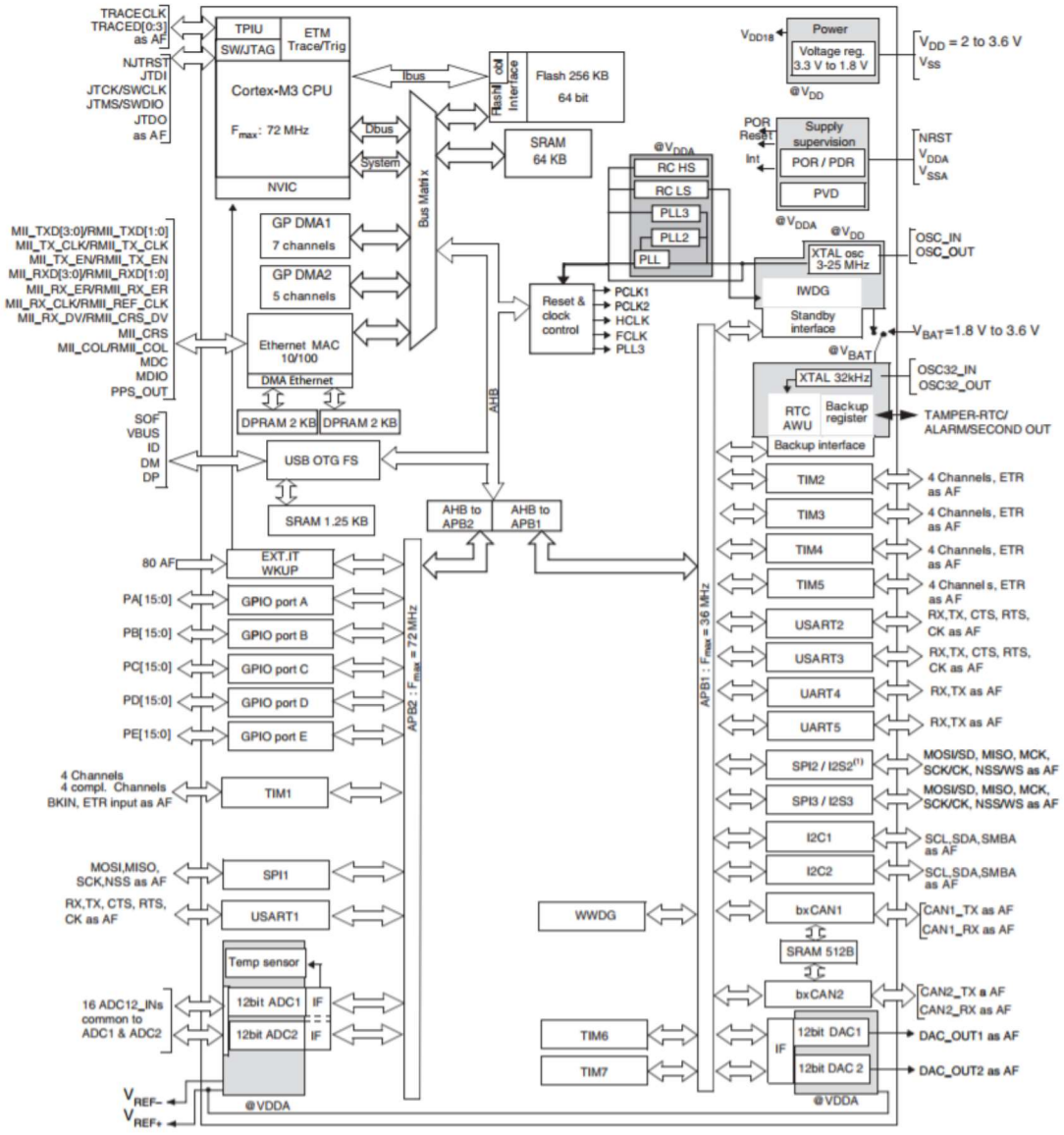


Figure 1: Block scheme of the processor [Con].

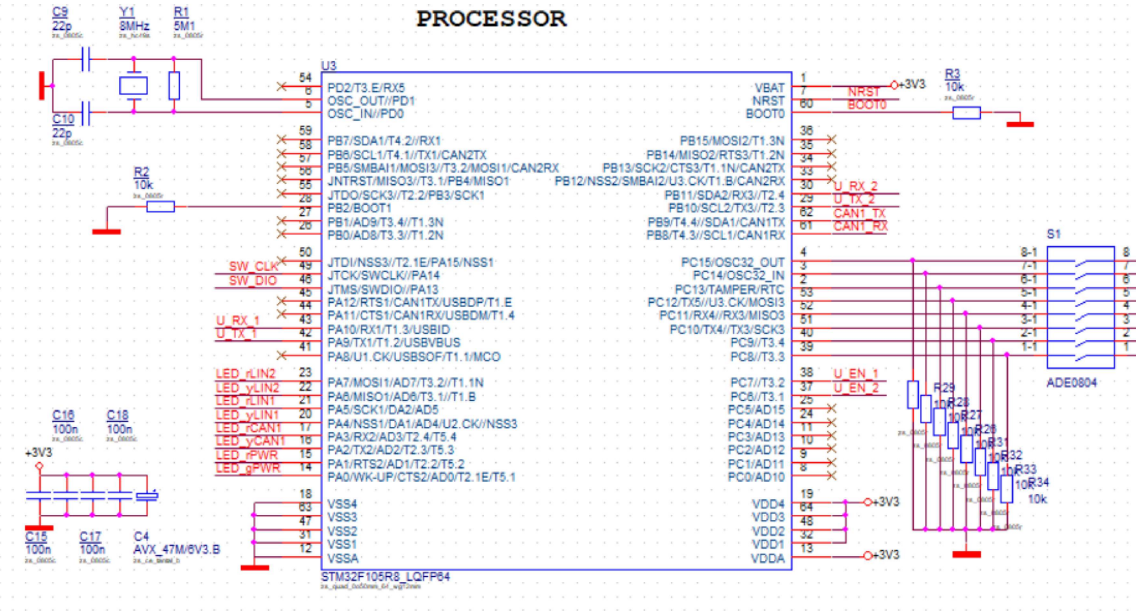


Figure 2: Connection of the processor.

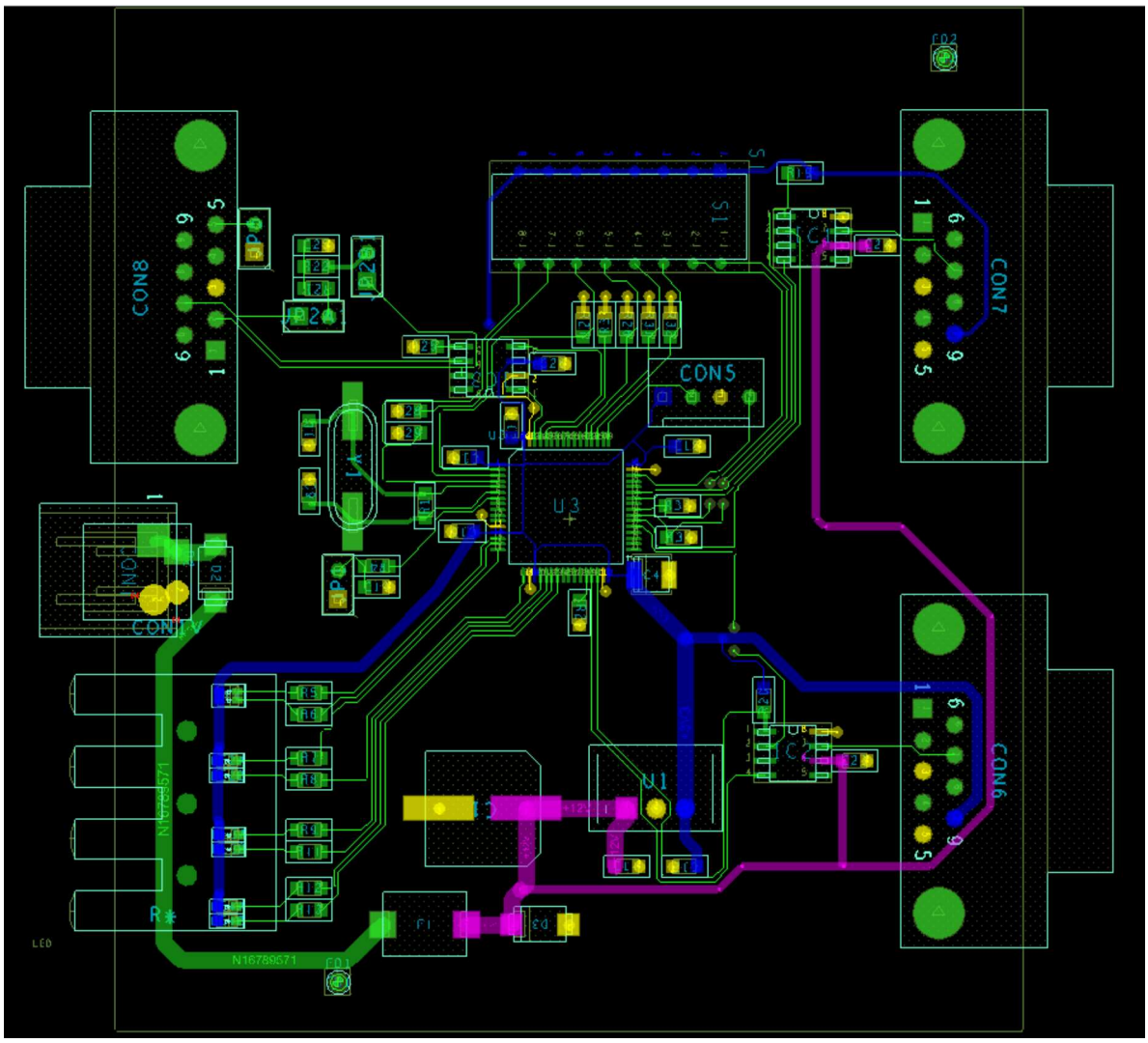


Figure 3: TOP layer of the PCB.

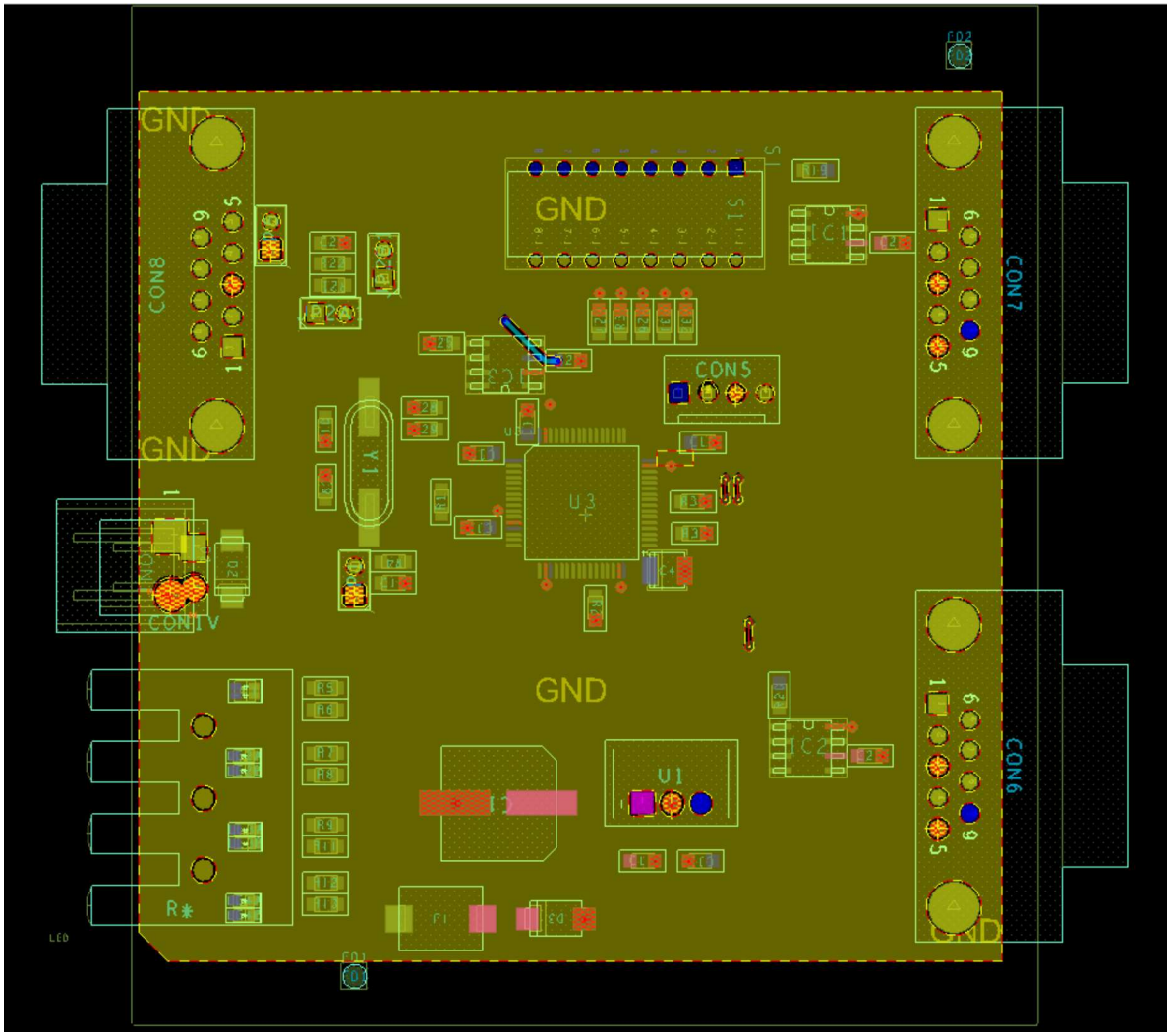
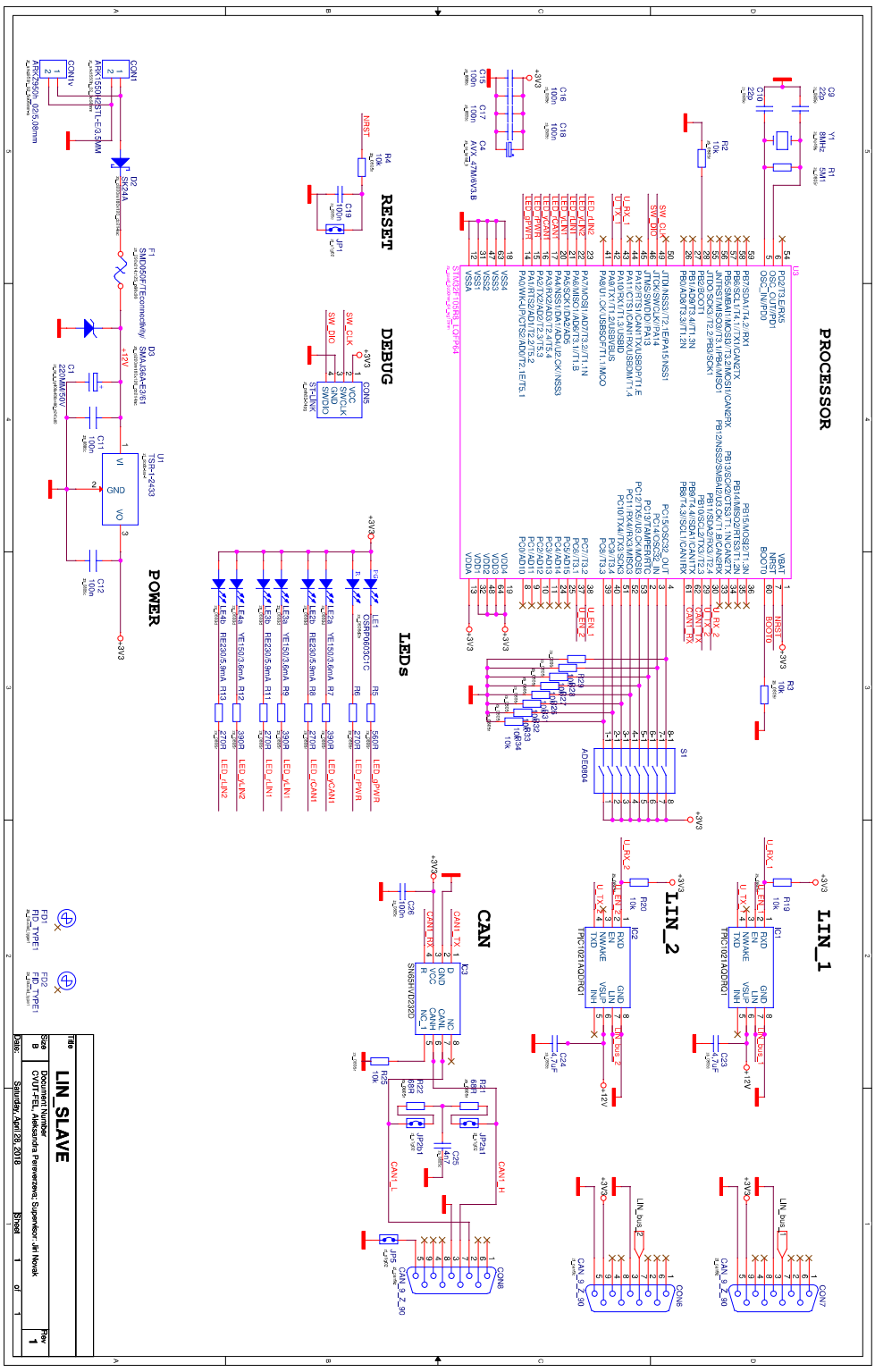


Figure 4: BOTTOM layer of the PCB.


PROCESSOR
LIN_1
LIN_2
CAN
LEDs
DEBUG
RESET
POWER
LIN SLAVE
Document Number
CVUT-FEL Aleksandra Pereverzeva, Supervisor: Jiří Novák
Saturday, April 28, 2018
Rev
1

Appendix A

Bibliography

[12] , *Lin network for vehicle applications*, nov 2012.

[Arc] *Archived: Benchmarking single-point performance on national instruments real-time hardware - national instruments*, http://www.ni.com/white-paper/5423/en/, (Accessed on 05/21/2018).

[Bos12] Bosch, *Wayback machine*, https://web.archive.org/web/20151211125301/http://www.bosch-semiconductors.de/media/ubk_semiconductors/pdf_1/canliteratur/can_fd_spec.pdf, 04 2012, (Accessed on 04/29/2018).

[CAN] *Can in automation (cia): History of the can technology*, https://www.can-cia.org/can-knowledge/can/can-history/, (Accessed on 04/29/2018).

[com] CyrilB commonswiki, *File:buck conventions.svg - wikimedia commons*, https://commons.wikimedia.org/w/index.php?curid=617371, (Accessed on 04/30/2018).

[Con] *Connectivity line, arm®-based 32-bit mcu with 64/256 kb flash, usb otg, ethernet, 10 timers, 2 cans, 2 adcs, 14 communication interfaces*, http://www.st.com/resource/en/datasheet/cd00220364.pdf, (Accessed on 04/30/2018).

[DEK+01] John V. DeNuto, Stephen Ewbank, Francis Kleja, Christopher A. Lupini, and Robert A. Perisho, *Lin bus and its potential for use in distributed multiplex applications*, SAE 2001 World Congress, SAE International, mar 2001.

[Emu] *Emulin*, http://www.ihr.de/index.php/en/products/101-produkte-lin-english/390-emulin-english, (Accessed on 05/10/2018).

[HMS02] Motorola Hamed M. Sanogo, *Implementing the can protocol in 3g equipment designs | ee times*, https://www.eetimes.com/document.asp?doc_id=1206484, 12 2002, (Accessed on 04/29/2018).

[htt] *https://upload.wikimedia.org/wikipedia/commons/5/5e/can-bus-frame_in_base_format_without_stuffbits.svg*, https://upload.wikimedia.org/wikipedia/commons/5/5e/CAN-Bus-frame_in_base_format_without_stuffbits.svg, (Accessed on 04/29/2018).

[IEEa] *Ieee 1149.1-2013 - ieee standard for test access port and boundary-scan architecture*, https://standards.ieee.org/findstds/standard/1149.1-2013.html, (Accessed on 05/21/2018).

[IEEb] *Ieee 1149.7 - texas instruments wiki*, http://processors.wiki.ti.com/index.php/IEEE_1149.7, (Accessed on 05/21/2018).

[ISOa] *Iso 11519-1:1994 - road vehicles – low-speed serial data communication – part 1: General and definitions*, https://www.iso.org/standard/19469.html, (Accessed on 05/23/2018).

[ISOb] *Iso 11898-1:2015 - road vehicles – controller area network (can) – part 1: Data link layer and physical signalling*, https://www.iso.org/standard/63648.html, (Accessed on 05/21/2018).

[ISOc] *Iso 17987-1:2016 - road vehicles – local interconnect network (lin) – part 1: General information and use case definition*, https://www.iso.org/standard/61222.html, (Accessed on 05/21/2018).

[ISOd] *Iso 9141-2:1994(en), road vehicles — diagnostic systems — part 2: Carb requirements for interchange of digital information*, https://www.iso.org/obp/ui/#iso:std:iso:9141:-2:ed-1:v1:en, (Accessed on 05/21/2018).

[ISOe] *Iso/cd 17987-8 - road vehicles – local interconnect network (lin) – part 8: Electrical physical layer (epl) specification : Lin over dc powerline (dc-lin)*, https://www.iso.org/standard/71044.html, (Accessed on 05/21/2018).

[KRE97] A. KREJČIŘÍK, *Napájecí zdroje i. 1.vyd.*, Praha: BEN, ISBN 80-86056-02-3., 1997.

[Leo] *Leonard-lin-bus.png*, https://cecas.clemson.edu/cvel/auto/Buses/images/Leonard-LIN-BUS.png, (Accessed on 05/11/2018).

[Lup04] Christopher A. Lupini, *Vehicle multiplex communication*, SAE International, may 2004.

[Med] *Media access control (mac layer) - definition*, http://ecomputernotes.com/computernetworkingnotes/communication-networks/media-access-control, (Accessed on 05/14/2018).

[Mic] Microchip, *Lin frame types - developer help*, http://microchipdeveloper.com/lin:protocol-dll-frame-types, (Accessed on 04/30/2018).

[Mos08] Anthony Moschella, *Simulation of lin clusters for reducing in-vehicle network development and validation costs*, SAE World Congress & Exhibition, SAE International, apr 2008.

[Phy] Phytools, *Phytools - can-to-lin interface, can repeater, lin slave, industry-best support*, https://www.phytools.com/CAN_to_LIN_Interface_CAN_Repeater_LIN_Slave_s/1829.htm, (Accessed on 04/29/2018).

[Ruf02] Matthew Ruff, *Implementing local interconnect network (lin) slave nodes*, SAE 2002 World Congress & Exhibition, SAE International, mar 2002.

[Sch] *Schematic-of-a-typical-in-vehicle-network-architecture-of-a-modern-automobile-12.png*, https://www.researchgate.net/profile/Jonathan_Petit/publication/266780575/figure/fig1/AS:295746776649730@1447522934824/Schematic-of-a-typical-in-vehicle-network-architecture-of-a-modern-automobile.png, (Accessed on 05/11/2018).

[sCN] Distributed systems and FEL CVUT Computer Networks, *Testing the distributed systems for the harsh environment*, https://moodle.fel.cvut.cz/pluginfile.php/83967/mod_resource/content/3/6_Communication%20Basics_cz.pdf, (Accessed on 05/14/2018).

[SN6] *Sn65hvd23x 3.3-v can bus transceivers datasheet (rev. o)*, http://www.ti.com/lit/ds/symlink/sn65hvd231.pdf, (Accessed on 04/30/2018).

[TPI] *Tpic1021 lin physical interface datasheet (rev. d)*, http://www.ti.com/lit/ds/symlink/tpic1021.pdf, (Accessed on 04/30/2018).

[Whaa] *What is a bypass capacitor?*, http://www.learningaboutelectronics.com/Articles/What-is-a-bypass-capacitor.html, (Accessed on 04/30/2018).

[Whab] *What is cyclic redundancy check (crc)? - definition from techopedia*, `https://www.techopedia.com/definition/1793/cyclic-redundancy-check-crc`, (Accessed on 05/21/2018).

[Zhu10] Yu. Zhu, *Can and fpga communication engineering: Implementation of a can bus based measurement system on an fpga development kit.*, Hamburg: Diplomica Verlag., 2010.

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Pereverzeva** Jméno: **Aleksandra** Osobní číslo: **453454**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra měření**

Studijní program: **Otevřená informatika**

Studijní obor: **Počítačové systémy**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Simulátor LIN Slave

Název bakalářské práce anglicky:

LIN Slave Simulator

Pokyny pro vypracování:

Navrhněte a realizujte HW modul pro funkční simulaci uzlu typu Slave. Implementujte programové vybavení modulu, umožňující konfiguraci komunikačních rámců, a to jak vysílaných, tak přijímaných. Konfigurace jednotlivého rámce musí umožnit volbu identifikátoru, délky rámce, způsobu výpočtu kontrolního součtu rámce a datového obsahu rámce včetně definice ErrorResponse bitu. Modul musí podporovat vysílání rámců generovaných událostmi. Konfiguračním rozhraním modulu bude CAN.

Seznam doporučené literatury:

[1] Kocourek, P., Novák, J.: Přenos informace, Skripta ČVUT, Praha 2003
[2] Lupini Ch.: Vehicle Multiplex Communication, SAE International 2004
[3] ISO 17987 - LIN Standard

Jméno a pracoviště vedoucí(ho) bakalářské práce:

doc. Ing. Jiří Novák, Ph.D., K 13138 - katedra měření

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **09.01.2018** Termín odevzdání bakalářské práce: **25.05.2018**

Platnost zadání bakalářské práce:
do konce letního semestru 2018/2019

doc. Ing. Jiří Novák, Ph.D.	podpis vedoucí(ho) ústavu/katedry	prof. Ing. Pavel Ripka, CSc.
podpis vedoucí(ho) práce		podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Studentka bere na vědomí, že je povinna vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. | Datum převzetí zadání

Podpis studentky