University of Tartu

Faculty of Science and Technology

Institute of Technology

Computer Engineering Curriculum

Erik Amor

Design and implementation of prototype firmware for ESTCube-2 primary communication subsystem

Bachelor's thesis (12 ECTP)

Supervisors:

Janis Dalbinš, MEng

Erik Ilbis, MSc

Tartu 2018

Abstract / Resümee

Software development for ESTCube-2 primary communication subsystem

This thesis is focused on software development for the primary communications system of ESTCube-2 nanosatellite. The goals are to describe and document the architecture of the systems software, develop primary prototype software and test it in different scenarios. The system must be compatible with previously agreed standards, developed communication system hardware and other subsystems of the satellite.

As part of this bachelor's thesis, top and low level functionality of ESTCube-2 communication system was stated and implemented, including data handling, data transmission, communication with other subsystems, error handling and logging.

Keywords: ESTCube, nanosatellite, radio communication

CERCS: T320 Space technology, T180 Telecommunication engineering

Kuupsatelliidi ESTCube-2 peasidesüsteemi tarkvara arendus

Käesoleva lõputöö teemaks on tarkvara arendus ESTCube-2 nanosatelliidi sidesüsteemi jaoks. Töö eesmärk on kirjeldada ja dokumenteerida kasutatav tarkvara ülesehitus, implementeerida selle tarkvara prototüüplahendus olemasoleva riistvara peal ja samuti testida valminud süsteemi eri stsenaariumite korral. Loodud süsteem peab vastama ESTCube-2 eelneva arenduse raames loodud standarditele, peab töötama väljavalitud ja väljatöötatud sidesüsteemi riistvara peal ja peab olema liidestatav teiste satelliidi alamsüsteemidaga.

Töö raames kirjeldati ja implementeeriti erinevat kõrgema ja madalama taseme funktsionaalsust ESTCube-2 sidesüsteemile, sealhulgas andmetöötlus, andmete hoiustamine ja edastamine raadioside teel, suhtlus teiste alamsüsteemidega, veatöötlus ja logimine.

Märksõnad: ESTCube, nanosatelliit, raadioside

CERCS: T320 Kosmosetehnoloogia, T180 Telekommunikatsioonitehnoloogia

Table of Contents

Abstract / Resümee 2

List of tables and figures 5

Abbreviations and terminology 6

1. Introduction 7

2. Overview of EC-2 and the COM subsystem 8

2.1. Overview of other similar missions 8

2.1.1. Communication subsystems 9

2.2. EC-2 communication subsystem hardware 9

3. Primary COM software architecture description 11

3.1. COM purpose in satellite and architectural requirements 11

3.2. Architecture overview for ESTCube-2 primary COM 12

3.3. Functional blocks in software architecture 14

3.3.1. Processor status and regular interrupts block 14

3.3.2. ICP control block 15

3.3.3. AX.25 packet control block 17

3.3.4. Transceiver IC control block 18

3.3.5. External FRAM memory IC control block 20

3.3.6. External DAC IC control block 20

3.3.7. SPI control block 21

3.3.8. Housekeeping data collection block 21

4. Software solution 21

4.1. Software overview 21

4.2. Main functional blocks 22

4.3. Developed functionality 22

5. Functional testing results 23

5.1. Testing of individual blocks 23
5.1.1. Testing data transmission 23
5.1.2. Testing ICP 24
5.1.3. Testing packet handlers 25
5.2. Discovered problems and their solutions 26
5.2.1. ICP multiple nodes 26
5.2.2. AX.25 not full bytes for transceiver FIFO 26
5.2.3. Challenges with common HAL 27
5.3. Possible firmware enhancements for future 27
5.4. Overall functionality rating 28
6. Summary 29
Ülevaade tööst eesti keeles 30
Acknowledgements 31
References 32
Appendix 34
I – Used hardware 34
II – Transmission testing 35
Non-exclusive license to reproduce thesis and make thesis public 36

List of tables and figures

Figure 1 - EC-2 COM hardware schematics 10
Figure 2 - Overview of main system blocks of primary COM software architecture and their interconnections 13
Figure 3 - Detail description of the ICP block of EC-2 PCOM software architecture description 16
Figure 4 - Detail description of the AX.25 block of EC-2 PCOM software architecture description 18
Figure 5 - Detail description of transceiver control block of the PCOM software architecture 19
Figure 6 - Detail description of FRAM buffering block of the PCOM software architecture . 20
Figure 7 - Hardware, used in the development and testing process 34
Figure 8 - Transmitting short AX.25 packet initiated over serial communication with COM board 35
Figure 9 - The packet sent out by COM board detected by HDSDR software, shown in waterfall 35
Figure 10 - The packet sent out by COM and demodulated by HDSDR software, shown in Audacity program 35
Figure 11 - several packets sent by the COM and demodulated by HDSDR, successfully received and decoded by the High-Speed SoundModem software 35

Table 1 - requirements for EC-2 PCOM subsystem 12

Abbreviations and terminology

ESTCube-1 / EC-1 – first satellite of Estonian Student Satellite program, orbiting the Earth

ESTCube-2 / EC-2 – second satellite of Estonian Student Satellite program, planned to orbit the Earth

ESTCube-3 / EC-3 – third satellite of Estonian Student Satellite program, planned to orbit the Moon

COM – communication subsystem of ESTCube-2

PCOM – Primary COM – primary communication subsystem

KCOM – Kill-COM – secondary communication system, capable of muting primary COM

OBCS - on board computer system

EPS – electrical power system

HAL – hardware abstraction layer

MCU – microcontroller unit

PCB – printed circuit board

UART - universal asynchronous receiver-transmitter

FRAM – ferroelectric random-access memory

FIFO – first-in-first-out (type of data organization and handling in memory buffer)

CRC – cyclic redundancy check

AX.25 - amateur X.25 (radio communication protocol)

AFSK - audio frequency-shift keying (modulation technique)

NRZI - non-return-to-zero inverted (data encoding type)

RSSI – received signal strength indicator

IC – integrated circuit

DAC – digital-to-analog converter

ICP – internal communication protocol

SPI – Serial Peripheral Interface

TX – transmission

RX – reception

TCVCXO – temperature compensated, voltage-controlled crystal oscillator

PID – protocol identifier

1. Introduction

From the beginning of the 21st century the space is more and more open for smaller industries and enthusiast groups, who don't need big and expensive satellites for their studies and experiments [1]. With the help of standardized nanosatellites and their modular launching system, it is possible to send small satellites to space with considerably smaller costs [2]. This brings more enthusiasts from different fields to the topic and helps to spread the knowledge about space technology [3]. However, as satellites are getting smaller and more compact, it challenges developers to miniaturize all systems of the spacecraft. For that, modern knowledge and engineering skill is needed. This makes developing small, standardized satellites popular in universities around the world as it provides create learning platform for wide range of fields.

University of Tartu is not an exception and in 2008 Estonian Student Satellite program was created [4]. The first mission, ESTCube-1 (EC-1) was launched in 2013 and turned out successful, after which ESTCube-2 (EC-2) development started [5] [6]. Ten years into the program and five years after first launch, the EC-2 development is in a state where hardware functionality for all satellite systems has been finalized and is in active development and testing process. The communication subsystem is in the same state.

The communication subsystem can, by some criteria, be considered the most critical system of a satellite as it mediates all the communication between ground and the satellite, making all other systems dependent on it. The responsibility of the subsystem is to provide uplink and downlink capabilities for command and data transfer but at the same time be configurable to account for environmental changes and errors, and work as efficiently as possible.

In EC-2 the communication subsystem (COM) is divided into two parts – primary COM (PCOM) and Kill-COM (KCOM), which are developed somewhat separately. This thesis is focused on the software development of the primary communication subsystem.

The main parts of the thesis are:

- PCOM software architecture description,
- Software development for PCOM,
- Testing of PCOM software,
- Providing description and reasoning for all previous parts as thesis document.

2. Overview of EC-2 and the COM subsystem

Estonian Student Satellite program is a program, started in 2008 in University of Tartu. The goal of the program is to support space research and the development of space technology in Estonia. It focuses on giving students an opportunity to learn new technology with hands-on experience [7]. The first mission of the program, EC-1, launched in 2013 and currently EC-2 is in development. Satellites of both missions are mainly developed and tested by students from University of Tartu with help from some scientists and companies.

EC-1, being the first satellite of the program, had a general mission of proving and showing Estonia as a space country. But apart from that it was also used to test E-sail technology [6]. Although the E-sail test wire failed to reel out, the mission was still successful, the satellite took and sent many pictures taken from space and also initiated the development of a successor – EC-2. This, second satellite of the program, is three times bigger, with dimensions of 10x10x30 cm or 3 U (units) by CubeSat standard, compared to 1 U or 10x10x10 cm of EC-1. It also means that EC-2 can fit a lot more payloads, have more sun panels and thus more powerful systems. However, one of the main goals of EC-2 is to test all the systems on board for reliability, as it is predecessor for EC-3 (ESTCube-3) which will be a Moon mission. As Moon missions are a lot more expensive, it is important to test the concept before. That is the reason, why EC-2 has several systems or system expansions on board that are not relevant for the mission itself but are there for future testing purposes.

2.1. Overview of other similar missions

There are several programs like Estonian Student Satellite Program from different universities around the world. A lot of them have sent more than one CubeSat type satellite into space. Some of the most long-term programs include AAUSAT missions from Aalborg University in Denmark and CP PolySat missions from Californian Polytechnic State University, which is also the place where the CubeSat standard started from [1]. The AAUSAT project was started already on 2001 and since then 5 CubeSats have been launched [8]. Currently the next generation of their improved satellites is under development [9]. Just like ESTCube, the project is mainly for educational purpose as all interested engineering students can join the development team. The CP PolySat satellite program was started already in 1999 by engineering students and has launched 12 satellites. They have improved from motivated students team all the way up to more advanced research missions supported and sponsored by

big science foundations and operated together with NASA (National Aeronautics and Space Administration) [10] [11].

2.1.1. Communication subsystems

Both, the AAUSAT and CP PolySat program have also had several problems with the communication subsystems of their satellites. Those problems have stated some must-haves of a simple satellite and some concepts that can be improved. For example, the very first satellite of Aalborg University suffered serious communication issues that never allowed fully functional communication with the satellite. With weak signal, not allowing to receive decoded data, they only received simple beacon data that was not enough to find the reason of the problem [12]. This is a good indicator that both the encoded beacon and simple Morse beacon should be used to send enough housekeeping data, because it might help to receive critical information to resolve on-board problems. Also, their connection-oriented AX.25 (Amateur X.25) communication turned out to have too much overhead [12], which concludes that communication with satellite should be kept as simple as possible while providing necessary functionality and performance. At least it should be possible to reset the communication to simplified format if a more complex version does not work. As another example, on the CP4 satellite by PolySat program, the intersatellite communication failed, that was implemented with well-known I2C (two-wire interface) protocol. The failure was most caused by an intermediate device failure [13]. This incident shows why a custom protocol and a backup communication bus can be useful.

These two programs have also tested some concepts that will also be used on EC-2. For example, both programs use AX.25 standard for radio communication and amateur radio frequency range in their satellites [14] [15]. Also, the CP8 satellite by PolySat program had a digipeater mode, meaning that it allowed to relay data through the satellite, which provides amateur radio enthusiasts a way of testing their systems [13]. This is also planned to be used on EC-2.

2.2. EC-2 communication subsystem hardware

The system described in this thesis is designed for a communication subsystem hardware solution that has been researched, designed and developed by the EC-2 team earlier in the development process. In this thesis this is taken as a working base for the firmware design and development.

The COM PCB (printed circuit board) has two subsystems on it (see Figure 1). Those subsystems are primary COM for both up- and downlink and Kill-COM for only uplink. They each have their own MCU (Microcontroller unit). The subsystems and their peripherals get power from the EPS (Electrical power system).

PCOM, that offers main communication functionality, has external transceiver chip, while KCOM has the same type of transceiver built into the MCU. Additionally, the primary COM has external FRAM (ferroelectric random-access memory) IC for non-volatile data storage and external DAC (digital to analog converter) IC. The name Kill-COM comes from the fact that the KCOM MCU has a hardware-based capability to turn off the power of PCOM amplifiers, in order to stop transmitting if the PCOM does not respond.

The primary COM can communicate with the rest of the satellite using GPIO (general purpose input-output) and RS-485 based ICP (internal communication protocol) lines. The communication between the PCOM and KCOM is done using pure UART (universal asynchronous receiver-transmitter) and the prototype hardware also has a RS-232 debugging connection with PC.

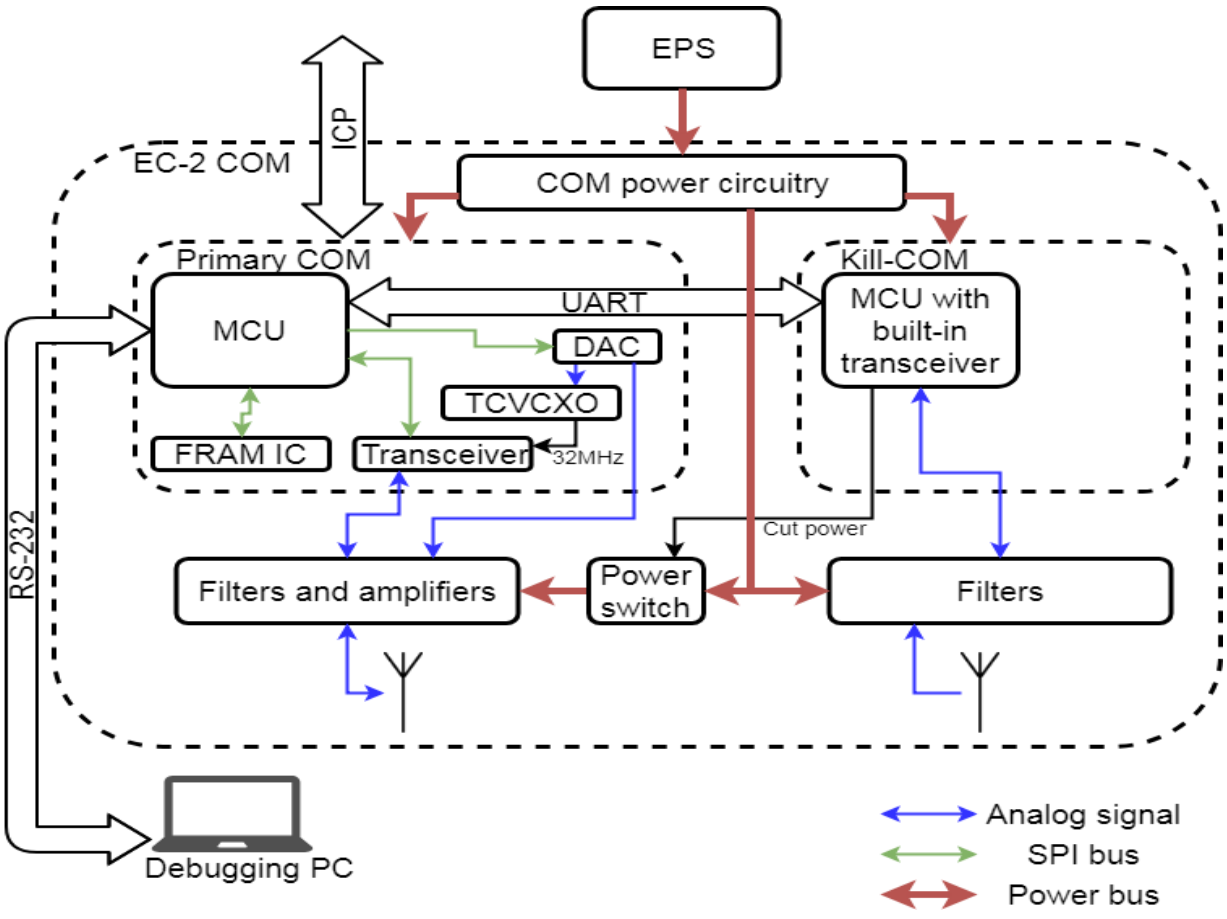

Figure 1 - EC-2 COM hardware schematics

3. Primary COM software architecture description

3.1. COM purpose in satellite and architectural requirements

The purpose of the communication subsystem in a satellite is to provide a link between the ground station and all the satellite systems that must communicate with the ground station. What makes COM (communication subsystem) one of the most critical systems is that it allows to control other subsystems from ground and make the information created or gathered by those other subsystems useful by transmitting it to the mission control center on Earth. Without a correctly functioning communications subsystem the satellite would become inaccessible space trash. To provide fast and reliable downlink and uplink, several architectural aspects must be learned and considered. As a part of this thesis the requirements for PCOM software were collected and a software architecture description for the subsystem in EC-2 was developed.

To achieve the main functionality of the communications subsystem – transmitting data between the ground station and rest of the satellite, the PCOM must be configurable but provide the service itself while being transparent. This means the service must be provided in a way that the subsystems have direct communication with the ground station and other way around just like the subsystems onboard communicate with each other. In other words, the COM should provide router functionality for packets.

Also, for the COM to be reliable, it is necessary that it provides information about itself and act upon any errors. This means that for better further control, the system must send critical information periodically to the ground station and log basic information on board for mission operators to know the status of the satellite and make decisions based on this information.

But the PCOM subsystem must also have some logic on board to handle cases when the uplink fails and there is no possibility to send recovery commands. In such scenario the subsystem should lower baud rate, go to safe mode with only main functionality and start sending critical beacons in simple encoding. If none of those actions help and the problem is in corrupted firmware, then the system should boot from backup firmware image.

All the requirements were gathered from and discussed with rest of the EC-2 team and specially COM team members and complete list of agreed requirements is stated in Table 1 alongside with the priority for each requirement from low to the highest priority level of critical.

ID	Requirement	Priority
1	Transmit data from the satellite to the ground station (Downlink functionality)	Critical
2	Receive data on the satellite, sent from the ground station (Uplink functionality)	Critical
3	Communicate with other subsystems over common protocol	Critical
4	Collect and transmit satellite and COM health and state information	Very High
5	Provide functionality to change data rate and modulation technique	High
6	Buffer data for bulk data transmission	High
7	Switch to safe mode in case of problems	High
8	Transmit health and state data as Morse beacon	High
9	Secure uplink by validating incoming commands	High
10	Provide functionality to retransmit not received bulk data packets	Medium
11	Provide digipeater functionality to retransmit frames received by KCOM	Medium
12	Automatic changing of data rate and modulation type for best speed	Medium
13	Provide functionality for clock synchronization for data timestamps	Low
14	Provide functionality to transmit FM (frequency modulated) soundtrack from satellite	Low

Table 1 - requirements for EC-2 PCOM subsystem

Considering all the previous requirements, a primary COM software architecture description was developed as part of this thesis. The architecture description was changed and improved during the development process to meet new requirements and keep the architecture description up to date with the actual system.

3.2. Architecture overview for ESTCube-2 primary COM

For better understanding the architecture of PCOM subsystem was developed with modular approach. The system is divided into blocks that each have their own role in the system (see Figure 2). It is also described how all these blocks interact with each other and what is their purpose in the whole system.

The main blocks of the system are:

- Processor status and regular interrupts block;
- Transceiver IC (Integrated circuit) control block, including two sub blocks:

    - Transceiver configuration block;
    - Data transmission block.
- External FRAM IC control block;
- External DAC IC control block;
- SPI (Serial peripheral interface) control block;
- Housekeeping data collection block;
- ICP control block, including two sub blocks:
    - ICP core and bus control;
    - ICP packet handler and creator.
- AX.25 packet control block.

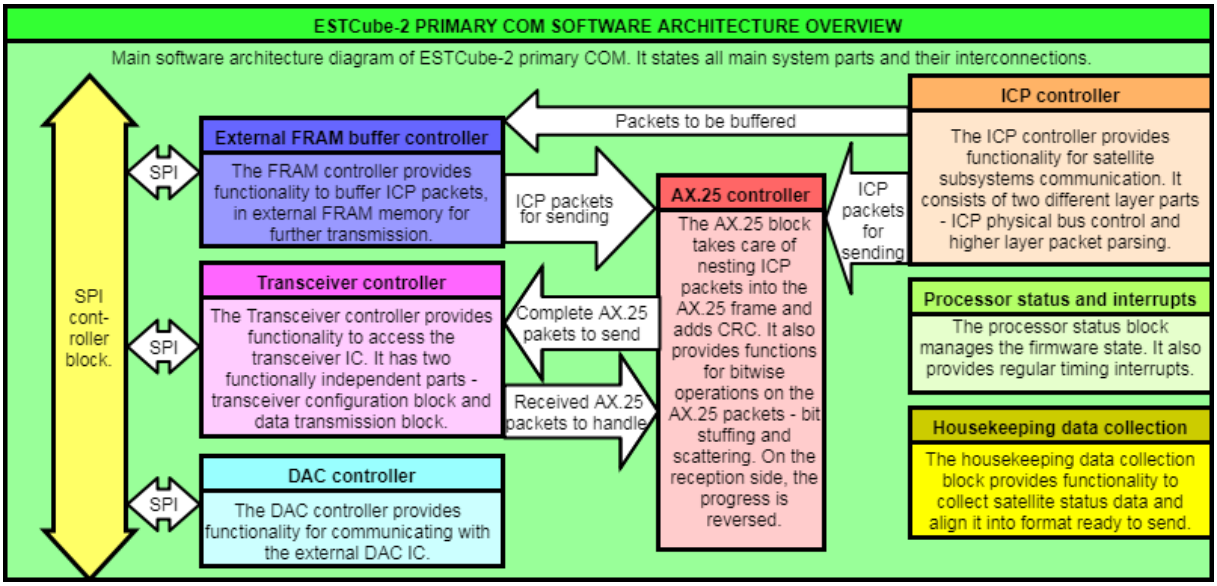

Figure 2 - Overview of main system blocks of primary COM software architecture and their interconnections

Each of these blocks has its own task in the final software. The processor status and regular interrupts block keeps track of the current state of the program and any state changes. It initiates any regular activities and acts upon any errors. Next main part of the PCOM subsystem software is the transceiver block. It provides both transceiver configuration options and data transmission functionality. External FRAM is used to buffer bulk data, like files and images, that come as multiple packets, prior to sending. DAC block is used to provide analog voltage to temperature compensated voltage-controlled oscillator (TCVCXO), which allows to fine tune the reference frequency for transceiver. Another DAC channel is used to give gate voltage for MOSFET transistor to determine DC bias point for the transistor amplifier. The external FRAM, DAC and Transceiver are physically separate components and communication with those works over

SPI. To provide media control for those three SPI devices, a SPI control and configuration block is used.

Other set of blocks provide data manipulation and parsing functionality. ICP control block takes care of managing the data flow on the ICP bus and providing correct ICP packet format. In addition, it creates the packets that are sent out and decodes incoming packets. AX.25 block takes care of packets that are ready to be sent out to the ground station or are received over radio. It manages the packet error checks and AX.25 header data manipulation. The housekeeping data collection block gathers all the necessary data for stating the current physical and software condition of PCOM subsystem and critical parts of other subsystems.

All the blocks are described more in-depth on separate diagrams, stating the functional logic that they work upon. This helps in developing the actual firmware and additionally helps to understand the code of the final solution for other coworkers from EC-2 team.

3.3. Functional blocks in software architecture

3.3.1. Processor status and regular interrupts block

The processor status block takes care of managing all required states. These states include working mode, transceiver mode, error states, interrupt states and running firmware version.

The PCOM subsystem has two working modes – normal mode and safe mode. It is kept track of and logged working mode is currently active. The firmware should go into safe mode when the program does not respond, or the satellite has not received command from the ground station for longer than expected. In safe mode, it should be still possible to upload new, fixed software, but most importantly the satellite should send easily decodable and informative beacon. For example, long Morse beacon. This allows to human-decode signal even if it is quite weak or corrupted up to a point. This way it could be possible to figure out what is wrong with the satellite and a fix can be developed.

Transceiver mode keeps track of whether the transceiver is in transmitting or receiving state and when it should switch between the modes. Besides, it handles the baud rate of data transmission, allowing to lower it when higher data rates don’t work because of weather conditions or other factors and increase again when it is ordered. If in dynamic baud rate mode, any change in baud rate must still follow very strict logic as it is important to know on the ground station, what baud rate to use to communicate with the satellite.

Errors might occur in all parts of the system – in communication with other systems (ICP errors), in data Transmission, in data buffering, in power supply and in other functional blocks. Those all require different approach. Some ICP errors require switching to secondary bus, some data buffering errors require dumping buffered data or trying to find valid data. Transmission errors might require reconfiguring the transceiver or resending some packets and power supply errors might require communicating with electrical power subsystem or going to sleep and turning off transceiver for some time.

Additionally, it is important to know what is the firmware version that is currently running for several reasons. First, when nothing other helps, the satellite should boot from backup firmware and then it is required to know when a backup firmware is used and if it was loaded correctly. Also, it is important in case of uploading and switching to new updated firmware. Then it needs to be known if the new firmware was loaded properly and the PCOM subsystem is now successfully using it.

The block also keeps track of interrupt states. When an interrupt occurs, the fact it has happened is saved and therefore the program always has overview of unhandled interrupts. These saved interrupt notifications are constantly handled in order of priority within the main program loop. This approach ensures that even if several interrupts happen at the same time, the ones with higher priority are handled before. Having only a flag-switching statement within the interrupt also makes the initial handlers a lot less time-consuming. That means, the most critical interrupts, if handled straight in the interrupt can be served while the lower priority interrupt handling is in progress.

3.3.2. ICP control block

In ESTCube-2 system there is a custom protocol used for communication between all subsystems. This protocol has been designed and is called and referenced as ICP in ESTCube documents.

ICP architecture consists of two different layer parts – ICP core and configuration part and higher layer packet managing part (see Figure 3). The underlaying core part is common for all subsystems of the satellite, providing reliable base, as communication is interpreted the same way throughout the system. The higher, packet processing layer takes actions on actual packets, including creating the information in the ICP packet that will be sent out and decoding the data of an incoming ICP packet.

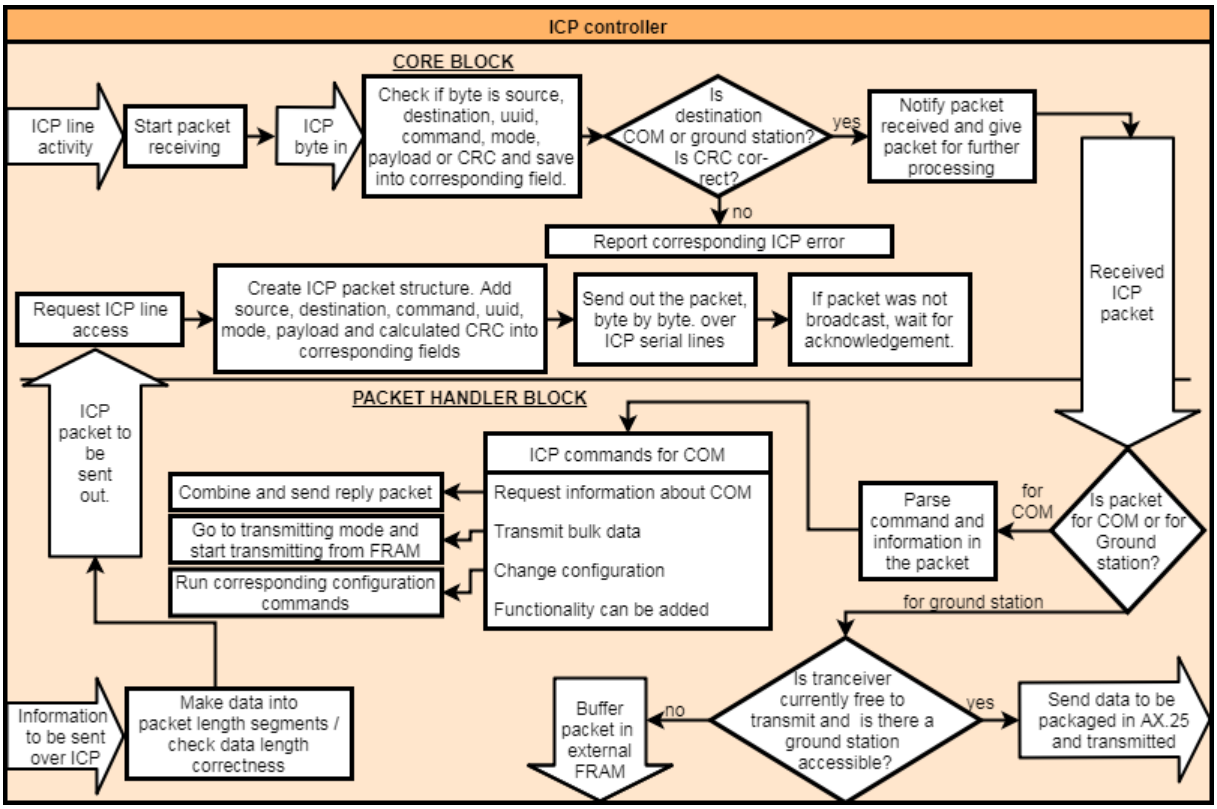

Figure 3 - Detail description of the ICP block of EC-2 PCOM software architecture description

The task of the core block is to manage the physical ICP lines according to the agreed ICP standard. It also does the filtering of packets according to protocol – in normal operation each subsystem should only accept packets destined for this certain subsystem or if the packet is broadcast. COM should only get packets destined for the COM or for the ground station. Systems can also listen to all packets if needed, but the acknowledgement scheme remains the same. The Core block also provides functionality to send and receive the fields of ICP packet in correct order and as continuous data flow.

The higher layer of packet handling allows to decode the command and information fields in the packet and initiate corresponding action. The ICP packet handler for COM should handle differently packets destined for the COM and packets destined for the ground station. Packets that are meant for the ground station should be buffered to be handled and transmitted later unless it is possible to transmit them right away. Packets destined for COM itself go into further parsing where dependent on the command field, certain functionality is invoked.

The packet handling layer is also responsible for creating fields for outgoing packets. This includes selecting corresponding destination and source nodes, command and information. The ICP controller of COM should be able to initiate communication itself, but also check and forward the packets that are received from the ground station and unpacked from AX.25 frame.

3.3.3. AX.25 packet control block

The AX.25 block is responsible for nesting data that is ready to be transmitted according to amateur radio protocol named AX.25. This standard provides rules for address field, control field, Protocol identifier (PID) field, data field and CRC field encoding and delimiter flags [16]. This metadata is added by the AX.25 block when transmitting and checked and removed when receiving. The standard also regulates how the delimiting flags are avoided inside the packet and how to encode the data, which is handled by this block by providing bit stuffing and NRZI (non-return-to-zero inverted) encoding. Bit stuffing for AX.25 states that there must not be any 6 consecutive 1-bits elsewhere in the packet than flags, so a 0 is stuffed (added) after every 5 consecutive ones [16]. NRZI encoding means that there is transition in the signal when a zero-bit occurs and no transition when one-bit occurs. Also, for data rates of 9.6 kBd and above, it is advised to use scrambling to avoid unwanted sequences of continuous ones or zeroes [17].

On EC-2 it is agreed that for compatibility between all subsystems and to avoid excessive data handling, all data is transmitted as an ICP packet. So, the maximum total length of ICP packet is made to be equal with the maximum length of AX.25 packet information field that is 237 bytes. The ICP packet however contains its own CRC (cyclic redundancy check) and as the CRC of AX.25 already covers this same data, it is inefficient to send both checksums. To use the space of the field more efficiently, the CRC of ICP packet is replaced with the packet sequence number on transmission that helps to group and recognize missing packets on the receiving side. The receiver can then send back bitmap image with information of which packets were not received, so these packets could be sent again.

When transmitting, the first operation is replacing the CRC of ICP packet with sequence number, then the CRC for AX.25 frame is calculated over modified data (ICP packet) and constant header fields – address, control and PID (protocol identifier) field. The whole packet is put together in corresponding order – first start flag of 0x7E, followed by Address, control and PID fields, then data, CRC and finally end flag, which is the same as the start flag. According to the AX.25 standard all other fields are transmitted low-order bit first, but CRC is transmitted bit 15 first [16], so to make further processing easier, the CRC is already put into the packet accounting this. After the packet is constructed the bitwise operations are done – first the packet is bit-stuffed, then scrambled and finally NRZI encoded.

When receiving, the whole progress is reversed, but some extra checks are made to identify whether the packet is correctly delivered and if it is meant for the satellite. This check is done

after the packet is decoded, descrambled and all stuffed bits are removed. If it appears that the packet is correct in all means, only then the AX.25 frame is removed, CRC of the payload packet is calculated and put to corresponding field and the received payload is sent for further processing. Otherwise the whole frame is dropped.

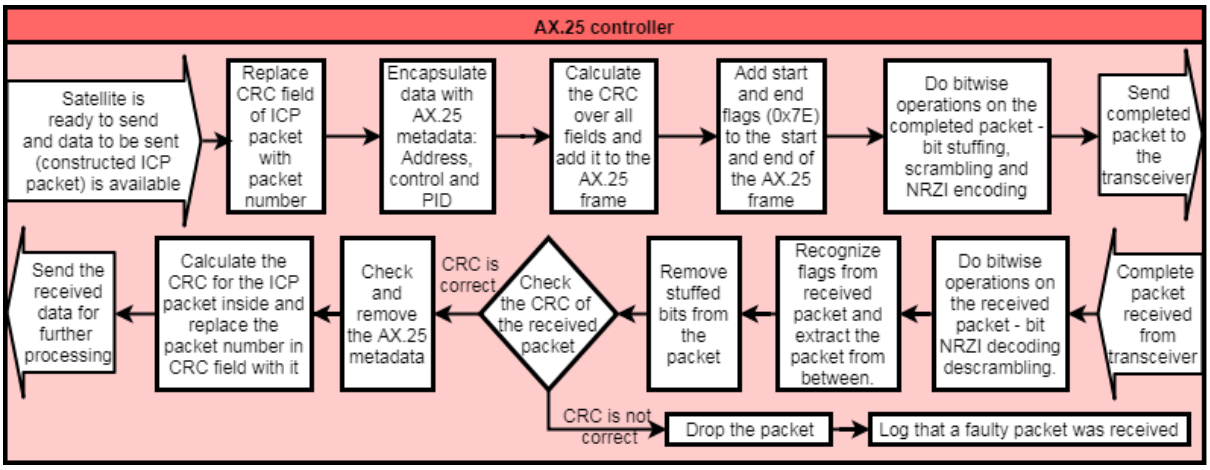

Figure 4 - Detail description of the AX.25 block of EC-2 PCOM software architecture description

3.3.4. Transceiver IC control block

The transceiver IC control block is used for communicating with transceiver and managing its state and properties. It provides functionalities for transmitting and receiving data, and for configuring, viewing different properties of the data transmission and the transceiver chip itself.

The transceiver properties can be configured all at one time, by sending full configuration array to transceiver IC or change only specific property by sending corresponding property command followed by property value. In the COM application the transceiver full initialization should be done on every reboot, but when changing some parameters during normal program operation, then the possibility of changing individual properties should be used. Overall in the system, only some properties have to be changeable, while others are the same on all program executions and in all conditions. Consequently, there is no need to provide functionality to change all possible parameters. Some of the changeable parameters include modulation scheme, baud rate, frequency deviation and internal amplifier gain meanwhile constant parameters are for example base(carrier) frequency, input crystal frequency, sync word and preamble, FIFO (first-in-first-out type of buffer) setup, incoming packet match conditions and all functionality that is not used – checksum calculation, channel selection and others.

Data that will be transmitted or has just been received is carried between transceiver IC and the processor over SPI. To make the process more resource-efficient, the transceivers internal data

buffers are used for both received and ready to transmit data. This allows the slow radio transmission to be handled mostly by transceiver and so freeing the processor from constant polling that also allows for longer sleep cycles or time for other processing. These transceiver buffers are 64 bytes each if used for reception and transmitting separately, but they can be used as one 128-byte buffer. However, a full-length AX-25 packet cannot fit into the transceiver buffer at once because the maximum length of the packet is over 256 bytes, meaning that processor must refill the FIFO during the transmitting process and read out bytes during reception. To account this problem of long packets, the transceiver IC has built in interrupts to notify that the internal FIFO buffer is getting almost empty while transmitting and that the reception FIFO is almost full while receiving. These interrupts could be used to let processor know that it is time to add more transmittable data to buffer or to read out received data, therefore ensure constant data transmission without gaps. The transceiver also has interrupts to let know that full packet has been transmitted or that a full packet has been received, letting know that the packet can be now handled.

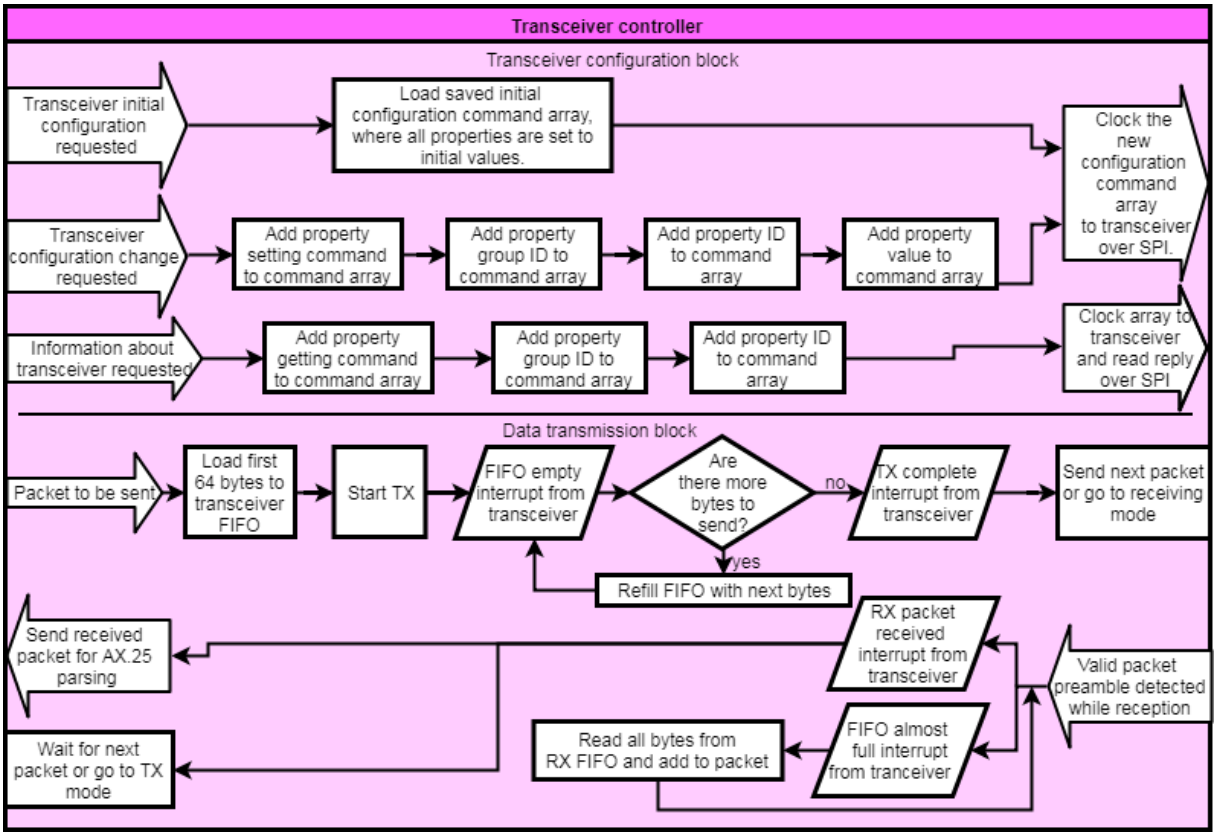

Figure 5 - Detail description of transceiver control block of the PCOM software architecture

3.3.5. External FRAM memory IC control block

The external FRAM memory is used to buffer outgoing packets. The buffer is used because bulk data like files and pictures come in through ICP faster than it is possible to transmit. Also, packets may be requested to transmit while there is reception in progress or there is currently no connection with the ground station. In all those cases, packets need to be buffered until they can be transmitted. For this a 4 Mbit external FRAM IC is used. The idea is to be able to buffer as many packets of information there without having to waste extra time. For this the packets are buffered as ICP packets. Storing a full packet also stores the packet length information, which allows the packet to take only as much space as it needs and still knowing where the packet ends. As the maximum agreed length of an ICP packet can be 237 bytes, the 4 Mbit or 524 KB FRAM can hold about 2200 full-length packets. The block provides basic functions for buffering packets and receiving buffered packets as well as buffer initialization, as seen on Figure 6.

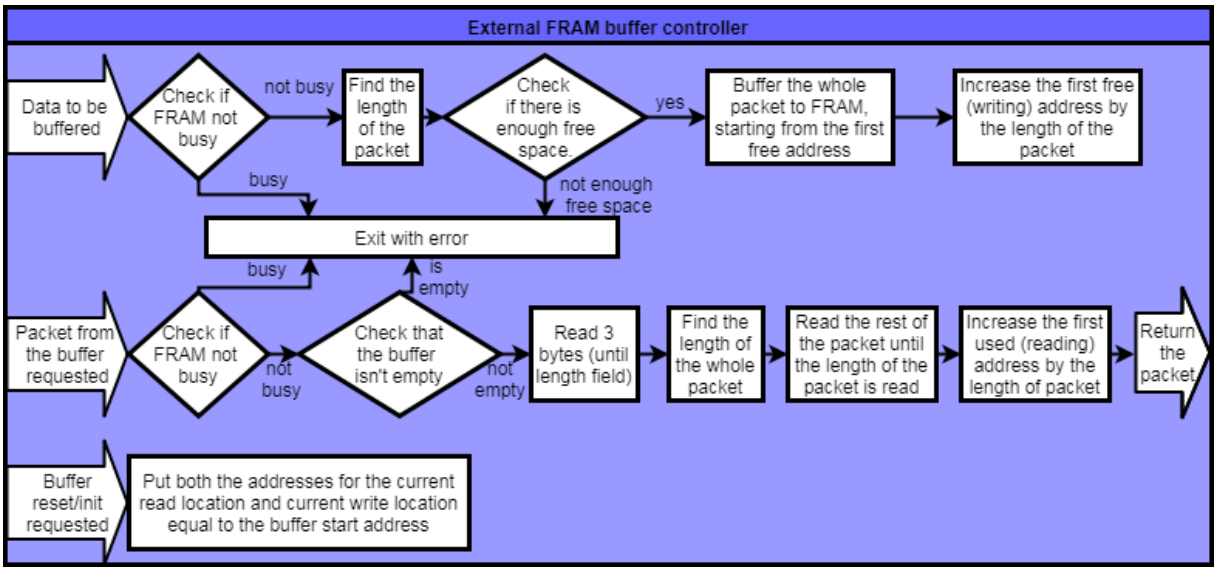

Figure 6 - Detail description of FRAM buffering block of the PCOM software architecture

3.3.6. External DAC IC control block

The DAC control block converts the desired voltage levels to be given to analog devices to corresponding SPI commands, where the channel and new value are specified. It sends them to the dual DAC IC, which then uses its internal parser to read the SPI input, loads the new value to the internal 10-bit DAC and outputs corresponding analog voltage. The voltage is kept in output, until new value update command is received.

3.3.7. SPI control block

The function of the SPI control block is to provide uniform access to all SPI slave devices and manage slave selection. The devices that are connected over SPI in current hardware version are the transceiver IC, external DAC IC and external FRAM. Both, the Transceiver and FRAM, use 2-way communication, where the slave replies to commands sent. The SPI block is responsible for managing this data flow and returning the replied information. The communication with DAC however is unidirectional and commands are just clocked in without waiting for reply.

3.3.8. Housekeeping data collection block

Housekeeping block takes care of collecting primary health data from the whole satellite. It uses COM internal measurement capabilities to collect information about the subsystem and transceiver state, and satellite internal communication to collect main information from other subsystems. Some of the information that is collected from COM include temperature of the board and MOSFET transistor amplifier, RSSI (received signal strength indicator), state information, error count and types, and power consumption.

The data that was measured, will be either forwarded to ICP block for combining it into packet or the data might be converted to Morse code and sent straight to transceiver, depending on the mode of operation.

4. Software solution

4.1. Software overview

The software for the primary COM is written in C programming language and made to run on an MSP430FR5994 embedded microcontroller. The programming was done using Code Composer Studio (CCStudio) integrated development environment and MSP430FR5969 LaunchPad Development Kit.

The software uses layered approach with the bottom HAL (hardware abstraction layer) layer providing universal access to processor functions and the higher layer containing COM logic. The HAL layer also allows the same top-level code to run on different processors of the same type with only changing configuration file in HAL. During this thesis, mostly the top layer

COM logic was developed on top of existing HAL. Only some additions were introduced to the HAL layer.

The software is written directly to the processor without having an operating system. The basic principle of the program is that all different events trigger a flag in a corresponding interrupt. All the flags are then checked and handled in the main loop in the order of priority.

4.2. Main functional blocks

The source code of the program has tree-like structure with the main firmware root folder being categorized into subdirectories according to the file properties – configuration files, driver files, application software files and common(external) files. In the configuration of the current thesis, the board configuration files are stored in “board” folder, driver files for different internal and external peripherals are stored in “drivers” folder, higher level application files are stored in “app” folder, satellite-wide ICP core source is stored in “ICPLib” folder and the underlaying MSP HAL code is stored in “HAL” folder. All these folders include subdirectories for header files and for source files.

4.3. Developed functionality

As part of this thesis some functionality that was stated above was implemented for the PCOM subsystem. Implemented parts are data transmission, including making use of AX.25 protocol, calculating CRC on PCOM hardware, communicating data to transceiver IC and controlling the transceiver to transmit the data. Another implemented module is for data buffering in external FRAM, which includes handling a circular FIFO buffer and usage of the FRAM IC API. Finally, the ICP testing capability was added to the COM firmware, including packet handlers, ICP configuration and testing commands.

To test all the added functionality, other smaller changes to the program were also developed. For testing purposes, debugging commands and debugging logic for all parts were developed to test them individually.

5. Functional testing results

5.1. Testing of individual blocks

For quicker and more efficient testing, in the COM software, serial communication with computer was implemented and used. This allowed to execute commands on COM board and viewing results and program state, without having other communication methods, like ICP and radio, available yet. This however couldn't be final testing approach as it is not part of final solution. It also uses same UART pins on hardware side as the main ICP line, meaning that it cannot be used while ICP is in use.

5.1.1. Testing data transmission

During the development process of the COM software, one of the most complicated things to test was the data transmission. Testing is made complicated by both the fact of great number of configuration parameter combinations and by usage of radio hardware and software on the ground station side.

In order to test the transmitting part most easily, using software radio on regular computer with external USB radio device was preferred. The device that was available for this was NooElec NESDR Mini 2+ 0.5PPM TCXO USB RTL-SDR Receiver with external antenna. To configure and process signal received from the dongle, the HDSDR freeware software was used. This software allowed to set the internal crystal frequency and spectrum of the dongle and do the FM demodulation of the signal. The outputted demodulated signal was then sent to another software for packet handling over virtual audio cable that allows to connect two programs on one computer. For this application VBCABLE software was used. For packet handling an G3RUH hs soundmodem was used. It allows to decode packets in several formats and with different data rates. In current use, the software was configured to decode AX.25 packets with AFSK modulation and output all their fields separately and data field in ASCII format. This allowed to test if packets that were sent from COM board and transmitted over radio signal, arrived correctly.

However, this software configuration didn't allow to see why packets were not received correctly as the hs soundmodem software only outputs correct packages. So, to first figure out in development process why packets were not received and what to modify, the packets were investigated more deeply using Audacity audio editing program. Looking at the demodulated

FM audio track from the HDSDR output in Audacity made it possible to look the data bit by bit, measure its baud rate, length and see how distinguishable the data is. The steps of testing the transmission are visualized in appendix, in Figure 8, Figure 9, Figure 10 and Figure 11

To test data reception, another setup was needed, because the dongle and computer-side software used are not capable of transmitting data. But as it was confirmed that transmitting data was working, it was possible to use one COM board as transmitter and another one in receiving configuration. To make the testing easier the transmitter board was configured to continuously transmit data with short intervals and an antenna was connected to the receiver board. As the configuration of the preamble and all transmission properties for both of the sides were configured identically, it was possible by interrupts to detect if the reception works as the transceiver chip provides capabilities to produce interrupts for valid preamble detection. If this part was working, it was easier to test data reception.

5.1.2. Testing ICP

In order to test and monitor ICP, special dongles were used, that allowed to monitor ICP activity from computer software. It also allowed to test one single subsystem at a time by providing capability of the dongle with computer-side software to act as a fully functional ICP node. However, this testing configuration had several problems that prevented from using this configuration for testing.

First, the dongle-side firmware had a communication speed problem that did not allow to relay real-time interrupts fast enough to computer-side software over UART connection. The problem was that whenever an access line of ICP was activated, then the packet follows in 250 microseconds. However, when access goes active, which is detected as falling edge on ICP access line, the dongle sent a 5-byte packet over UART to PC. This packet was first sent with 19.2 kBd baud rate and therefore took 2.1 ms to send, which means the actual data started to come in before the notification that access went active was received. So, the data was not listened. This problem was solved by increasing the baud rate of the UART connection between dongle and PC to 500 kBd, which is near the limit of what the ATmega88 chip on the dongle can handle.

After resolving the first problem, another issue occurred with the UART communication and buffers. The processing of UART packets on computer side has some operating-system level buffers and logic added to it, which does prevent desired data flow in that high speed as 500

kBd. To overcome this problem, the EC-2 OBCS (on-board computer system) team, whose responsibility is developing the PC-side software, added bigger delays to the ICP for testing purposes.

Finally, the PC-side software, which is thread-based to listen on 3 UART channels simultaneously, encountered some threading problems and therefore the testing of ICP on COM with this software was postponed out of the time-scope of this thesis.

However, the primary functionality of ICP implementation on PCOM subsystem was tested with two COM boards. Both COM boards were configured identically, only difference was that one of the boards was configured as COM node and other one as OBCS node to test packets with different destination points. During this test, the ICP1 bus was tested, as the UART lines of ICP0 bus were used for communication with PC. Both nodes also were able to send packets to OBCS and to COM. The packets had random but distinguishable data, so that the validity of the packet could be determined on the other end. Sending the packet and determining which of two test packets will be sent was determined through the debugging serial connection. Also, the received packet was printed to the serial.

This testing approach can validate the basic functionality of ICP, however it could not determine if the implementation is compatible with implementations from other subsystems and neither could it find out all the issues that can occur, when more than two nodes communicate on the same bus.

The testing results showed that there is a misconfiguration of the receiving node, as the packets are sent out, but the access line is never released. This indicates that the packet is not recognized or accepted correctly, and the receiving node does not release the access line. The fact that the sender outputs correctly formatted packet and releases the line, was checked using the dongle on other end and measuring signals on lines using logic analyzer.

5.1.3. Testing packet handlers

To test the ICP packet handlers and the radio communication packet handlers simultaneously, the previous ICP testing configuration was planned to use in slightly modified way. Two testing ICP packets could be added to one COM board, one that requests a reply packet from other node with increased value and other one that requests the ICP packet to be sent over radio.

The packets could be analyzed on the other ICP endpoint by the ICP packet handler that dependent on the ICP command field value, takes the corresponding action. To verify the result

of the packet transmitting command, the same configuration can be used that was explained in the testing data transmission section. For the reply request packet, it can be checked if the correct reply packet is printed to serial, from where the initial request packet was sent. This testing can be done once the ICP testing gives positive results.

5.2. Discovered problems and their solutions

During the development of the COM system several problems were faced. Some of them were solved during the development process, but some of them still need more advanced research to work as supposed to. For those problems, temporary solutions have been offered.

5.2.1. ICP multiple nodes

ICP was first designed so that all subsystems use same core library for handling the sending and receiving of packets and controlling ICP lines. The ICP protocol and therefore also the core library was designed so that if the packet is not broadcast then only the system that the packet was meant to, will listen for the packet. Correspondingly the library had option to specify the node that is used to specify which packets should be listened and what will be the source when sending out packets. The problem was that during the development it was not considered that the COM subsystem must have two different destination nodes to listen to – packets destined for COM itself but also packets destined for the ground station. Fixing the problem was made harder by the fact that the library defined the same node to be used when sending and receiving. Making fix in the entire library would have added overhead to other systems as well which is not necessary and would require modifying the library that was by that time finalized.

The temporary solution to this problem was to modify the library locally on COM subsystem. However, this fix has some drawbacks to it. Firstly, it makes the library inconsistent and in case of any updates it must be remembered that the fix has to be applied again. Also, it makes the code harder to read as the fix did not follow the best coding standards.

5.2.2. AX.25 not full bytes for transceiver FIFO

One of the problems encountered during data transmission programming was the combining of individual data bits into full 8-bit bytes. The problem source is that AX.25 standard specifies bit stuffing as described above, which means adding individual bits into the data array after five one-bits. However, the number of such occurrences in data can vary a lot and most of the times is not divisible by 8. This means that after the bit stuffing the length of data in bits might not be

divisible by 8 either. The problem then is that if not providing data bit by bit to the transceiver chip but using the chips internal FIFO buffer instead, then the data must be converted to bytes as the FIFO is byte based. Also, the chip does not know anything about the packet ending and will recognize that the packet is transmitted only if the FIFO is emptied. Therefore, the last byte that will still have some data in it, but might not be full byte, must be filled with some dummy bits that are still transmitted although not needed. To reduce any excessive noise, one-bits were added there, as then there are no transitions and constant frequency is retained.

5.2.3. Challenges with common HAL

Using common HAL over several processors has several challenges. First, as the HAL must have functionality for all processors of the family, some functionality, that only few processors have, can easily be forgotten to add. Any HAL level errors are hard to debug as there is lot of code. Additionally, functionality that works with the same HAL on other processor, is not considered as the cause for the problem in first place, so debugging might take long time. In current development the problem with HAL occurred with GPIO interrupts, when the MSP processor in use had more port interrupts available than other versions used in other EC-2 subsystems. Thus, the missing of those declarations in HAL was not discovered earlier and ended up in long debugging.

Another problem with HAL is that as it contains a lot of potential functionality description, that is not used or not present on a current processor, it uses up lot of program memory. In future this problem might become problematic and for this reason it is planned to make possible to specify the modules that will be used from HAL, discarding rest of the code, making the HAL usage more effective.

5.3. Possible firmware enhancements for future

As the development and testing of ESTCube-2 continues, then also the COM software must be modified, improved and new functionality needs to be added. Also testing of the COM operation continues and at some point, the whole satellite as a combined system must go through several tests, meaning there is still lots of work to be done with COM system as well.

Some firmware enhancements that are known to be added but were not part of this thesis include adding custom bootloader to COM, implementing bitmap protocol, uplink securing protocol, exact mode switching algorithm and logging.

The main goal of custom bootloader is to allow updating the COM firmware while in space. The working principle of firmware update is that new firmware is sent over radio communication to satellite, then it is saved to processors internal FRAM memory and checked for validity. Then command is issued to reboot and start from new firmware, meaning that the new firmware is copied by bootloader to executable memory part and then executed. The custom bootloader should also allow to boot from backup firmware image in case the original image gets corrupted. The bitmap allows to detect missing packets and request resending of those. Uplink has to be secured, so that others can not control the satellite, even if they record and resend the commands.

5.4. Overall functionality rating

Based on the testing results of the implemented software parts, it can be stated how well the implemented architecture solution works on the firmware.

It can be said that the implementation was mostly successful, because all the parts of the PCOM firmware that were implemented, worked at least at base level and fulfilled the minimum requirements. The implementations that were tested and taken into account, were for data transmitting, data reception, data buffering in external FRAM and ICP communication.

The data transmitting worked in the level that was desired in the thesis. In future it can be made more configurable and needs boundary and reliability testing also in more realistic environments. Data reception however still has problems to it, and currently only reading raw bits from transceiver works. It might be the best solution possible, as the transceiver does not recognize AX.25 packets, but this is a drawback compared to initial architecture plan for the thesis. The FRAM buffering works in current mode, on ground but another backup mode should also be developed for space, because current implementation is vulnerable by bit errors. The second mode would be benefit compared to initial architecture. ICP communication works well on transmitting side, but there are still line-handling bugs on PCOM implementation that have to be solved.

Considering all those results, it can be said that the prototype software development was successful for some parts and partly successful for others, however, the implementation proves the concept that the PCOM subsystem can operate in the way planned with no major drawbacks. There are still several more parts to the PCOM subsystem to implement, and several tests to carry out before the final functionality for the system can be stated.

6. Summary

In this thesis the requirements and architecture of the EC-2 PCOM firmware were stated. As part of the thesis, also the prototype firmware was developed and tested and the approaches and results for both were described in the thesis.

The main outcomes of the thesis are:

- Full system architecture description for EC-2 PCOM as text and as diagram
- Prototype firmware for PCOM, with following parts implemented:
    - Receiving and transmitting data using AX.25 standard and AFSK modulation
    - Communication with other satellite subsystems over ICP protocol
    - Packet handlers for ICP and AX.25 communications
    - Internal command parser with basic commands
- Test results showing that the hardware and software of PCOM allows to fulfil the main requirements determined for the subsystem.

The development of the PCOM firmware however is not over with the part described in this thesis. There is still functionality to implement and improve according to tests and future needs. The final testing of the satellite can be only done when all the subsystems are finished and tested individually and even then, modifications can be done.

Ülevaade tööst eesti keeles

Käesolevas töös kirjeldati ära nõuded ja tarkvara arhitektuur EC-2 peasidesüsteemi jaoks. Lisaks valmis töö raames algne tarkvara implementatsioon, mida ka testiti. Töös on kirjeldatud nii tarkvara arenduse kui ka testimise protsesse ja tulemusi.

Töö peamised väljundid on:

- Täielik süsteemi arhitektuuri kirjeldus, vastavalt välja toodud nõuetele, nii diagrammi kui lihttekstina
- Prototüüp tarkvara peasidesüsteemijaoks, milles on implementeeritud järgmised osad :
    - Andmete saatmine ja vastuvõtmine üle raadioside, kasutades AX.25 kommunikatsiooniprotokolli ja AFSK tüüpi modulatsiooni
    - Suhtlus ülejäänud satelliidiga kasutades kohandatud ICP suhtlusprotokolli
    - Pakettide käsitlejad nii AX.25 pakettidele kui ka ICP pakettidele
    - Sisemine käsuinterpretaator koos funktsionaalsusega peamistele käskudele
- Testi tulemused, milles kajastub, et peasidesüsteemi riistvara ja tarkvara võimaldab täita sidesüsteemile määratud peamisi nõudeid

Käesolev töö on aga vaid osa peasidesüsteemi arendusest ning alamsüsteemi arendustegevus jätkub kuni satelliidi lõpliku valmimiseni. Peasidesüsteemile, nagu ka teistele satelliidi alamsüsteemidele on veel vaja lisada uut ja täiendada olemasolevat funktsionaalsust. Seda nii puuduste likvideerimiseks kui ka lisanduvate nõuete täitmiseks. Viimased testid, mille käigus võib samuti esineda muutuste tegemise vajadus, saavad toimuda alles siis kui kõik satelliidi alamsüsteemid on valmis ja kokku pandud. Sellepärast on rõhku pandud ka koodi loetavusele ja dokumenteerimisele.

Acknowledgements

I would like to thank the ESTCube team for giving me the chance to work in this great team, where great knowledge is spread to all team members. This gives a great change to work in a big team and obtain skills that are not part of the university program. Special thanks to Ervin Oro and Indrek Sünter, who thoroughly explained more complicated concepts to me.

I even more thank my supervisor Janis Dalbinš who agreed to arrange great number of meetings with me, gave me a great base for understanding the whole ESTCube communication system and always helped me to understand the base code. Also, big thanks to my co-supervisor Erik Ilbis who helped a lot in debugging and finalizing the document.

Finally, I would like to thank my girlfriend and relatives who often had to wait more, because of ESTCube debugging sessions that took longer than expected.

References

[1] The CubeSat Program, Cal Poly SLO, "CubeSat Design Specification, Rev. 13," 20 February 2014. [Online]. Available: https://static1.squarespace.com/static/5418c831e4b0fa4ecac1bacd/t/56e9b62337013b6c063a655a/1458157095454/cds_rev13_final2.pdf. [Accessed 23 April 2018].

[2] NASA CubeSat Launch Initiative, "Basic Concepts and Processes for First-Time CubeSat Developers," October 2017. [Online]. Available: https://www.nasa.gov/sites/default/files/atoms/files/nasa_csli_cubesat_101_508.pdf. [Accessed 23 April 2018].

[3] J. W. Pang, B. Bo, X. Meng, X. Z. Yu, J. Guo and J. Zhou, "Boom Of the Cubesat: A Statistic Survey Of Cubesats Launch In 2003-2015," September 2016. [Online]. Available: https://www.researchgate.net/publication/311279467_BOOM_OF_THE_CUBESAT_A_STATISTIC_SURVEY_OF_CUBSATS_LAUNCH_IN_2003-2015. [Accessed 23 April 2018].

[4] A. Slavinskis, K. Reinkubjas, K. Kahn, E. Hendrik, K. Kalnina, K. Erik and e. al, "The Estonian Student Satellite Programme: providing skills for the modern engineering labour market," December 2015. [Online]. Available: https://www.researchgate.net/publication/286921508_The_Estonian_Student_Satellite_Programme_providing_skills_for_the_modern_engineering_labour_market. [Accessed 23 April 2018].

[5] H. Ehrpais, I. Sünter, E. Ilbis, J. Dalbins, I. Iakubivskyi, E. Kulu, I. Ploom, P. Janhunen, J. Kuusk, J. Šate, R. Trops and A. Slavinskis, "ESTCube-2 Mission and Satellite Design," June 2016. [Online]. Available: https://www.researchgate.net/publication/303974831_ESTCube2_mission_and_satellite_design. [Accessed 18 April 2018].

[6] S. Lätt, A. Slavinskis, E. Ilbis, U. V. K. Kvell, E. Kulu, M. Pajusalu, K. Henri, S. Indrek and et.al, "ESTCube-1 nanosatellite for electric solar wind sail in-orbit technology," 2014. [Online]. Available: http://www.kirj.ee/public/proceedings_pdf/2014/issue_2S/Proc-2014-2S-200-209.pdf. [Accessed 18 April 2018].

[7] ESTCube team, "Foundation | ESTCube," [Online]. Available: https://www.estcube.eu/en/foundation. [Accessed 7 May 2018].

[8] AAUSAT team, "AAU Student Space - AAUSAT homepage," 16 April 2016. [Online]. Available: http://www.space.aau.dk/index.php?n=Main.HomePage. [Accessed 5 May 2018].

[9] AAUSAT team, "AAUSAT6 Space to ground communication," 29 January 2016. [Online]. Available: http://www.space.aau.dk/aausat6/index.php?n=Main.Space-to-groundCommunication. [Accessed 28 April 2018].

[10] PolySat team, "Gunter's Space Page - CP missions," 15 April 2018. [Online]. Available: http://space.skyrocket.de/doc_sdat/exocube.htm. [Accessed 27 April 2018].

[11] PolySat team, "PolySat homepage - Misions Launched," [Online]. Available: http://www.polysat.org/launched/. [Accessed 27 April 2018].

[12] L. Alminde, M. Bisgaard, D. Vinther, T. Viscor and K. Z. Østergard, "AAU Student Space," 2004. [Online]. Available: http://www.space.aau.dk/aausatii/homepage/en/dok/Handover/Project%20References/aausat/AAUCubesatProject.pdf. [Accessed 20 January 2018].

[13] B. Klofas, "A Survey of CubeSat Communication Systems," November 2008. [Online]. Available: http://www.klofas.com/papers/CommSurvey-Bryan_Klofas.pdf. [Accessed 7 May 2018].

[14] C. Noe, "Design and Implementation of the Communications Subsystem for the Cal Poly CP2 Cubesat Project," 11 June 2004. [Online]. Available: http://www.crn2.inpe.br/conasat1/projetos_cubesat/subsistemas/COM/CP2%20-%20COM%20-%20Communication%20Subsystem.pdf. [Accessed 27 April 2018].

[15] AAUSAT team, "AAUSAT-II launch info - GndSetup," 2008. [Online]. Available: http://www.space.aau.dk/aausatii/eng/index.php?n=Site.GndSetup. [Accessed 7 May 2018].

[16] W. A. Beech, D. E. Nielsen and J. Taylor, "AX.25 Link Access Protocol," July 1998. [Online]. Available: https://www.tapr.org/pdf/AX25.2.2.pdf. [Accessed 5 May 2018].

[17] J. Miller, "9600 Baud Packet Radio Modem Design," 31 October 2005. [Online]. Available: http://www.amsat.org/amsat/articles/g3ruh/109.html. [Accessed 4 May 2018].

Appendix

I – Used hardware

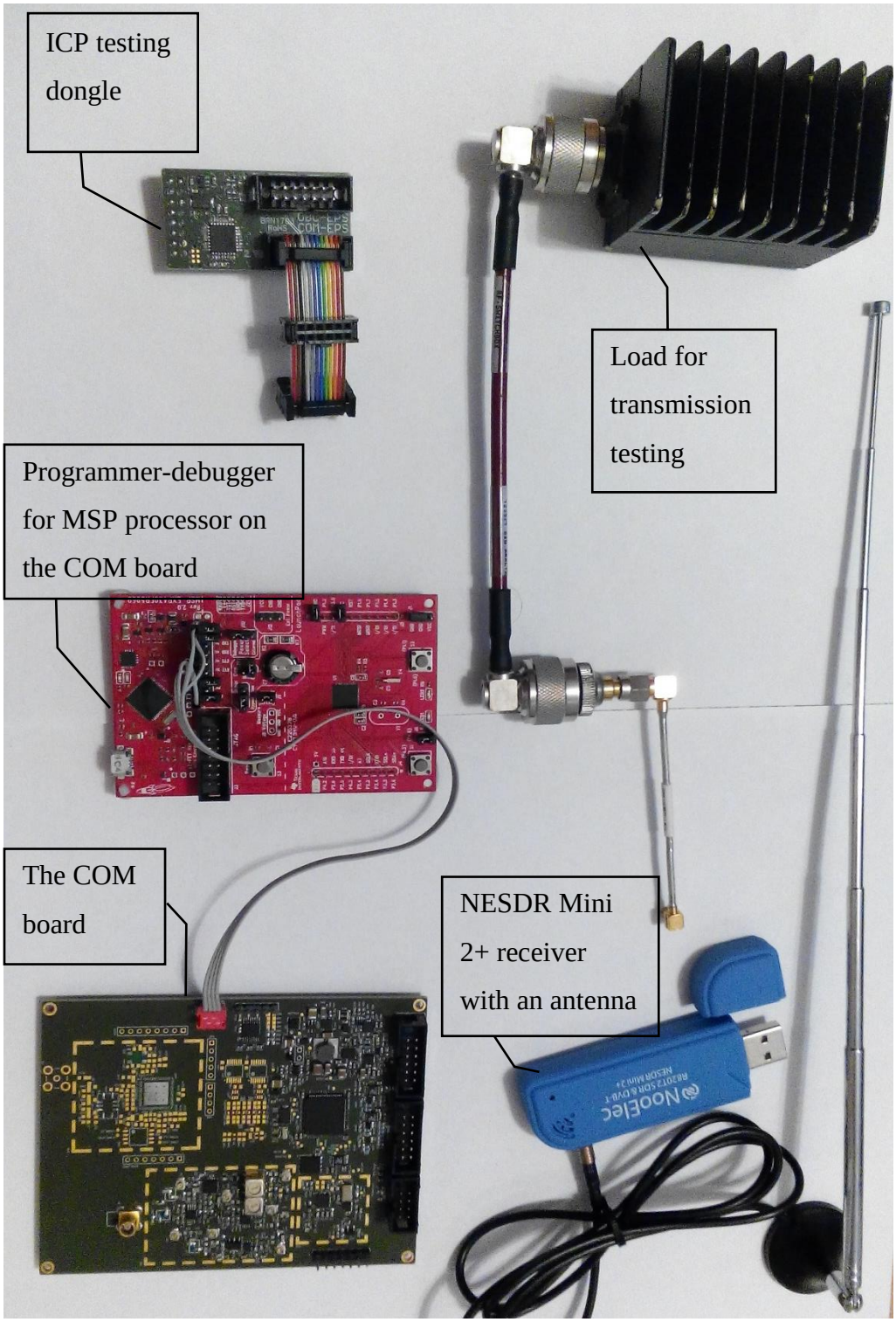

Figure 7 - Hardware, used in the development and testing process

II – Transmission testing

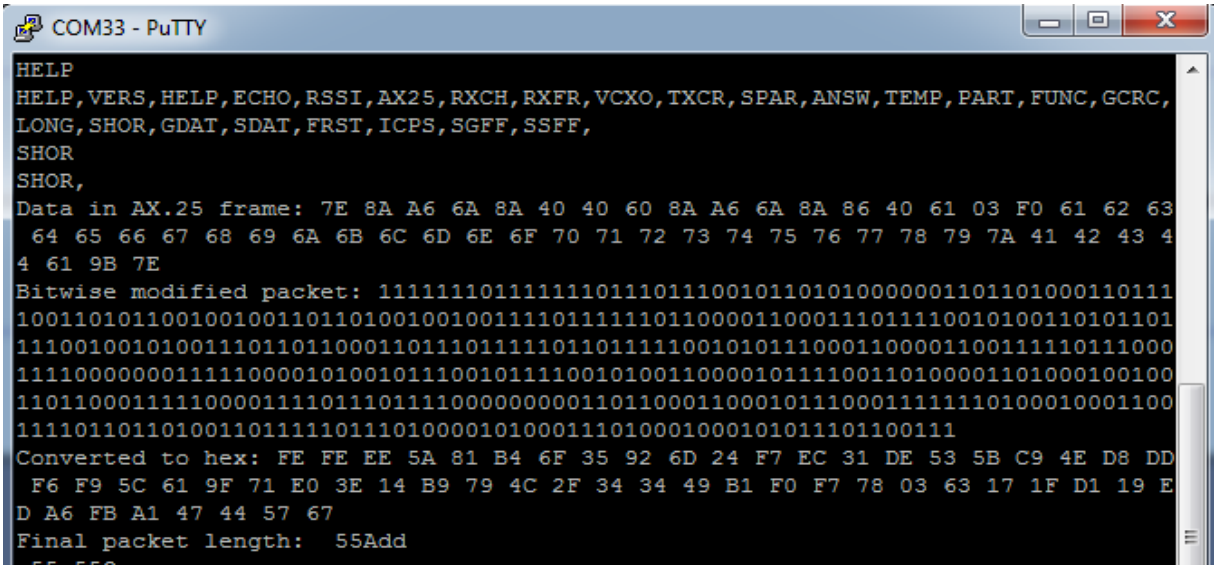

Figure 8 - Transmitting short AX.25 packet initiated over serial communication with COM board

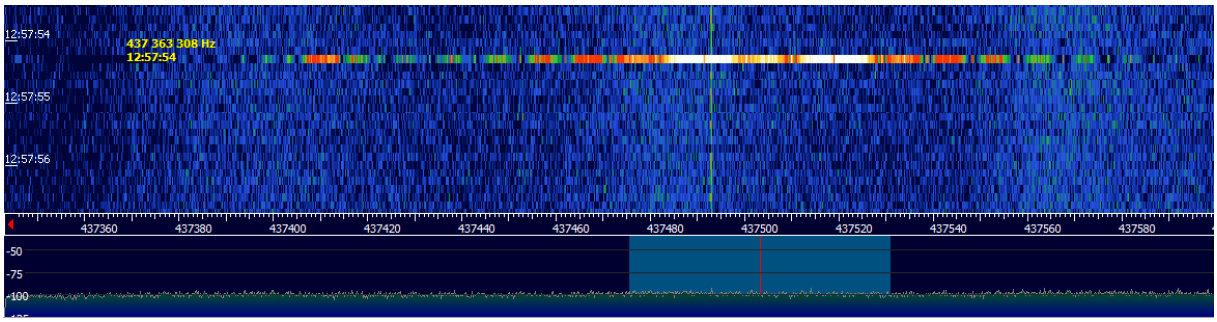

Figure 9 - The packet sent out by COM board detected by HDSDR software, shown in waterfall

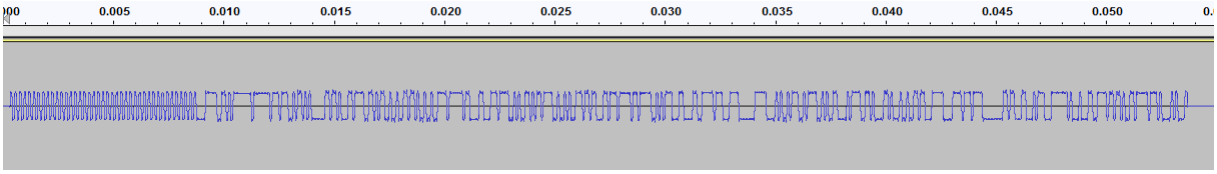

Figure 10 - The packet sent out by COM and demodulated by HDSDR software, shown in Audacity program

Figure 11 - several packets sent by the COM and demodulated by HDSDR, successfully received and decoded by the High-Speed SoundModem software

Non-exclusive license to reproduce thesis and make thesis public

I, Erik Amor,

1. herewith grant the University of Tartu a free permit (non-exclusive license) to:
    a. reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and
    b. make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright,

    of my thesis

    „Design and implementation of prototype firmware for ESTCube-2 primary communication subsystem”

    supervised by Janis Dalbinš and Erik Ilbis.

2. I am aware of the fact that the author retains these rights.
3. I certify that granting the non-exclusive license does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu, 20.05.2018