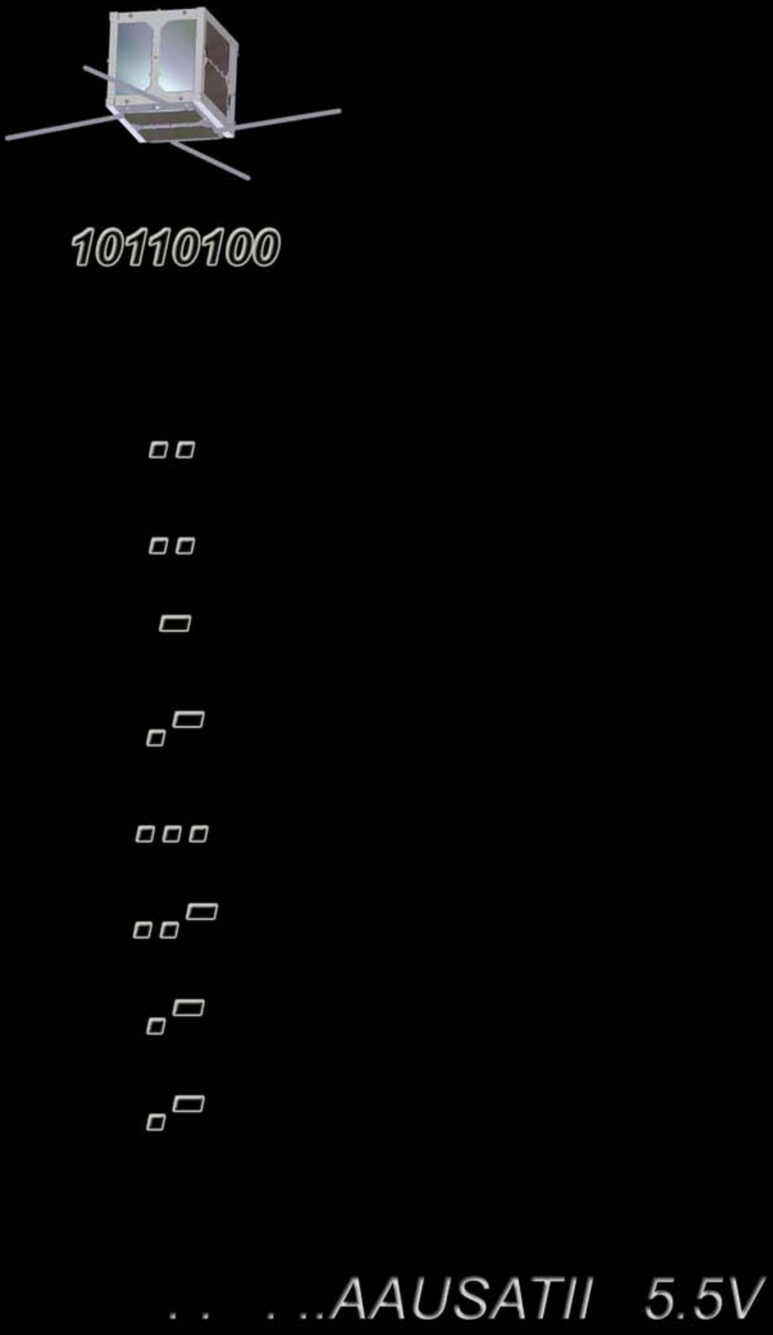
AALBORG UNIVERSITY

THE AAUSAT-II COMMUNICATION SYSTEM



8th semester - group 834

2005

Intelligent Autonomous Systems

Aalborg University

Institute of Electronic Systems

Fredrik Bajers Vej 7 ▪ DK-9220 Aalborg Øst Telephone +45 96 35 87 00

Title:	Communication System for AAUSAT-II
Theme:	Modelling and Control
Project period:	February 2nd 2005 - May 30th 2005

Project group:
IAS8-05GR834

Group members:
Bo Ørsted Andresen
Claus Grøn
Rasmus Hviid Knudsen
Claus Nielsen
Kresten Kjær Sørensen
Dan Taagaard

Supervisor:
Dan Bhanderi

Publications:	9
Pages:	55
Finished:	May 30th 2005

Abstract
The AAUSAT-II satellite depends on a reliable onboard COM system in order to accomplish the primary technical mission: Establish one-way communication. A COM system has been designed with respect to the CubeSat concept, the hardware fits onto a 4 layer, $8.7\,[\text{cm}] \times 8.7\,[\text{cm}]$ PCB and the weight and power consumption is kept on a minimum. COM consists of a microcontroller, a modem and a radio which is built by a German radio amateur, Holger Eckardt.
The software has been designed and low level functions have been implemented and tested with success.
This includes transmitting data on the radio at baud rates of 1200, 2400 and 4800 [bps] with FFSK modulation. Also an 800 [Hz] sine wave for Morse code has been generated using PWM and a low-pass filter.
A prototype PCB has been built and tested with success. The hardware and the low level software have been tested with success, yet no system level software is finished.

Preface

This document is written as documentation of the design and implementation of the AAUSAT-II communication system. The document is made as a secondary project at 8th semester by group 834 at the Department of Control Engineering, Aalborg University. The project is continued from the 7th semester where it was initiated in the middle of the project period as a secondary project.

At the end of the project period the AAUSAT-II communication system has not been fully implemented, therefore this documentation also includes the ideas and thoughts behind the design of the system.

Throughout the documentation, figures, tables and equations are numbered consecutively according to the chapters. Citations are referring to the bibliography at page 44, e.g. [12, Page 5] is referring to page 5 in the data sheet for the modem CMX469A.

In [ACS 2005, Chapter 1] an overview of all subsystems in the AAUSAT-II project can be found. This includes a description of the entire project as a distributed system. A detailed description of the mission objectives and the satellite modes is also presented in the same chapter.

Source files, schematics and other documents used for designing the AAUSAT-II communication system may be found on the enclosed CD-ROM.

Aalborg University May 30th 2005

Bo Ørsted Andresen	Claus Grøn
Rasmus Hviid Knudsen	Claus Nielsen
Kresten Kjær Sørensen	Dan Taagaard

Table of Contents

1 Introduction **6**
- 1.1 Purpose 6
- 1.2 Analysis 6
- 1.3 System Demands 9

2 Hardware **10**
- 2.1 Analysis 10
- 2.2 Design 11
- 2.3 Test 16
- 2.4 Conclusion 19

3 Software **20**
- 3.1 Analysis 20
- 3.2 Design 25
- 3.3 Flowcharts 30
- 3.4 Test 39
- 3.5 Conclusion 41

4 Conclusion **42**
- 4.1 Future Work 42

Bibliography **44**

A Programmers Reference **45**

B Interface Control Documents for COM **48**
- B.1 External ICD 48
- B.2 Internal ICD 49

C Circuit Diagram and PCB Layout **53**

Introduction 1

The first chapter concerns an introduction to the COM project, starting with the purpose of COM and the demands that follows from this. An analysis of the interfaces for COM defines the specific demands the COM system shall fulfil. Since a satellite is a complex autonomous system, robustness of each subsystem is an important issue that must be obeyed.

This document is written as documentation of the design and implementation of the AAUSAT-II communication system. The purpose of the system is defined and an analysis is conducted in order to set up the system requirements. In the following chapters the design of the hardware and the software is described. Finally the constructed system is tested and the document is ended with a conclusion and a list of future work that needs to be done on the communication system for AAUSAT-II prior to the launch.

The AAUSAT-II is a CubeSat designed and built primarily by students at Aalborg University. The satellite follows the CubeSat concept yielding a total weight of maximum 1 [kg] and dimensions of 10 [cm] × 10 [cm] × 10 [cm] [CPSU]. To be able to communicate with the satellite from the ground station, placed at Aalborg University, a communication system is needed onboard the AAUSAT-II satellite.

1.1 Purpose

The secondary mission goal for the AAUSAT-II project states that:

"In order to ensure the communication with the satellite is functioning, one-way communication is established when a basic beacon with the correct syntax is received at the ground station. When two-way communication is established, by sending a command to the satellite and a correct answer corresponding to the command, is received from the satellite, the communication objective is fulfilled." [ACS 2005, Section 1.2]

Based on that, the purpose of the Communication System (COM) is to enable the AAUSAT-II to communicate with a ground station by the means of a radio and thus it shall work as a link between the ground station computer and the AAUSAT-II OnBoard Computer (OBC). One-way communication means that it is possible to receive and decode beacons from the satellite containing basic telemetry data. Two-way communication means that it is possible to send a command to the satellite and get an acknowledgement which can be decoded.

1.2 Analysis

In this section the requirements of the COM system are analyzed. The result is a set of system demands from which the communications system shall be designed. Some components had already been chosen at the beginning of this project by the AAUSAT-II project management group and therefore the analysis of these components will not be as thorough as could be expected.

1.2.1 External Interfaces

The external interfaces have been defined in cooperation with the other subsystem groups and describe the interfaces to the COM system from the outside point of view. A description of these interfaces has been gathered in an Interface Control Document (ICD), see Appendix B on page 48.

Internal Communication All subsystems on the AAUSAT-II satellite communicates with each other using a Controller Area Network (CAN) bus running at 125 [kbps] using a protocol developed by the Payload group. `http://www.aausatii.aau.dk/wiki/index.php/Payload_System`

External Communication The satellite communicates with the ground station using an FM radio with a carrier frequency of 437.425 [MHz] with a $\pm$ [3kHz] deviation. The data transmission uses data rates primarily at 1200 [baud] and 4800 [baud] for transmitting data which should be selectable from the ground station, by sending a command to CDH (Command and Data Handling) which then sends a command to COM to change baud rate. Other selectable data rates are desirable to be chosen in case of a bad link quality.

Electrical Interface To lower the power consumption a supply voltage of 3.3 [V] is chosen for COM in cooperation with EPS (Electrical Power System).

Antenna Interface The antenna is a crossed dipole with an impedance of 50 [Ω]. It is developed by the Communication Department at Aalborg University, and built by the Mechanical team of the AAUSAT-II project.

Debug Interface To ease the development of the COM system an RS232 debug interface is added to the design, because CAN connections to a PC involve non-standard hardware and it can be difficult to get a reliable link running. This has another advantage as well; the debug interface can be used as a connection to a PC on the ground station side of the link. This eliminates the need for two different hardware designs on the satellite and the ground station, it is only a matter of different software.

1.2.2 General Demands for AAUSAT-II

The design of the satellite is based on budgets for mass, power, data and the radio link. These budgets are constrained by the demands for the CubeSat concept developed by California Polytechnic State University, San Luis Obispo and Space Systems Development Laboratory at Stanford University [CPSU]. These demands are a maximum weight of 1 [kg] and a dimension of 10 [cm] $\times$ 10 [cm] $\times$ 10 [cm], and caused by the small size of the satellite the power budget is tight. The low power budget also affects the transmission power which together with the low flight orbit makes the data transmission limited during an overflight of the ground station.

For each of these limitations for the satellite, a budget has been set up with data from all subsystems estimating the amount of power, weight and housekeeping data generated. These budgets have not been made as a task on its own, but have been an ongoing iterative process during the whole project with all relevant subsystems, until each budget has been approved and locked for changes. The communication between subsystems in order to fulfil these demands and budgets are controlled by ICDs. Each subsystem has published an ICD with all the information about the external interfaces for the subsystem. The ICDs for the COM system can be found in Appendix B on page 48.

The ICDs for the other subsystems and the final budgets for mass, power, data and link can be found on the enclosed CD-ROM in the `budgets` folder.

Generally it can be said that each subsystem shall minimize the power consumption, the weight and that only the most relevant data should be gathered for housekeeping data to be sent to the ground station for further analysis. An extra task for the ground station and the COM system is that the radio transmission circuitry should be optimized in order to optimize the link budget.

1.2.3 Preselected Components

Microcontroller The MicroController Unit (MCU) used on all subsystems, except OBC, is a PIC18LF6680 with on-chip FLASH, RAM and EEPROM. It has an internal CAN controller, USART, A/D converters, PWM generators and timers. The MCU runs at clock frequencies up to 40 [MHz]. It has been chosen because it is very versatile and it fulfils the needs of all subsystems. The advantage of using the same microcontroller is that the teams on the different subsystems can help each other with problems regarding software implementation.

Radio The radio is being built by a German radio amateur, Holger Eckardt, it is a half-duplex UHF radio with a frequency of 437.425 [MHz]. It is light, compact and it can run from a 3.3 [V] supply. The FM radio has the following properties [Eckardt]:

- Frequency: 437.425 [MHz]
- TX Power: 610 [mW]
- AF frequency response: 1 [Hz] to 6 [kHz] (-3 [dB])
- AF modulation voltage: 1 [V_{pp}] (350 [mV_{RMS}]) for 3 [kHz] deviation
- AF output voltage: 250 [mV_{pp}] (90 [mV_{RMS}]) at 3 [kHz] deviation
- TX-on delay: < 5 [ms]
- RX-on delay: < 5 [ms]
- Temperature range: -30 [°C] to +70 [°C]

From the frequency response it can be seen that the radio is well suited for transmission of bandwidth limited audio signals, but it is not able to transmit a DC signal. Two different modulation schemes are researched, Frequency Shift Keying (FSK) and Audio Frequency Shift Keying (AFSK). They are chosen because they are simple and easy to implement.

In FSK the radio is modulated with two different voltages corresponding to a binary one or zero. This would be the binary data stream sent directly into the radio, and it would result in two different frequencies sent from the radio corresponding to a binary one or zero. Since the radio is unable to transmit a DC voltage it is not possible to send a digital data stream directly into the radio, because long sequences of ones or zeros would be incorrectly transmitted. Therefore it is decided not to use FSK modulation.

When using AFSK modulation the radio is modulated with two different audio tones at different frequencies corresponding to a binary one or zero. This has the advantage that the radio is able to transmit and receive the signal correct even if a long sequence of ones or zeros are sent. The audio tones for the radio can be generated by FSK modulation of the digital data stream. This can be done by an FSK modem available as a single chip, and therefore it is found that AFSK modulation is a feasible method of communicating with the radio.

1.3 System Demands

The system demands for the COM system is summarized in Table 1.1.

No.	Requirement
1	The COM system shall communicate with other subsystems on a CAN bus at 125 [kbps]. The protocol and software is developed by the Payload group.
2	The COM system shall communicate with the ground station using an FM radio at a carrier frequency of 437.425 [MHz] with a ± [3kHz] deviation.
3	The COM system shall transmit data to the ground station at at least 1200 [baud] and 4800 [baud]. It shall be possible to switch between different baud rates in case of a bad radio link.
4	The hardware on the COM system shall be designed for using 3.3 [V] as supply voltage.
5	It shall be possible to send an 800 [Hz] Morse tone for the basic beacon.
6	An RS232 interface shall be implemented on the COM system to be used for the debug interface.

Table 1.1: Requirements for the COM subsystem

Hardware 2

In this chapter the design, implementation and test of the COM hardware is described. A simple and robust solution has been chosen which complies with the "Keep It Simple, Stupid" design principle used throughout the AAUSAT-II project. In order to test the design, two nearly identical prototypes have been built. One will be used with the ground station and the other is for the satellite. The only difference between them is the radio interface.

2.1 Analysis

The block diagram of the COM system can be seen in Figure 2.1. The main components are shown, and their functionality will be described in this section.

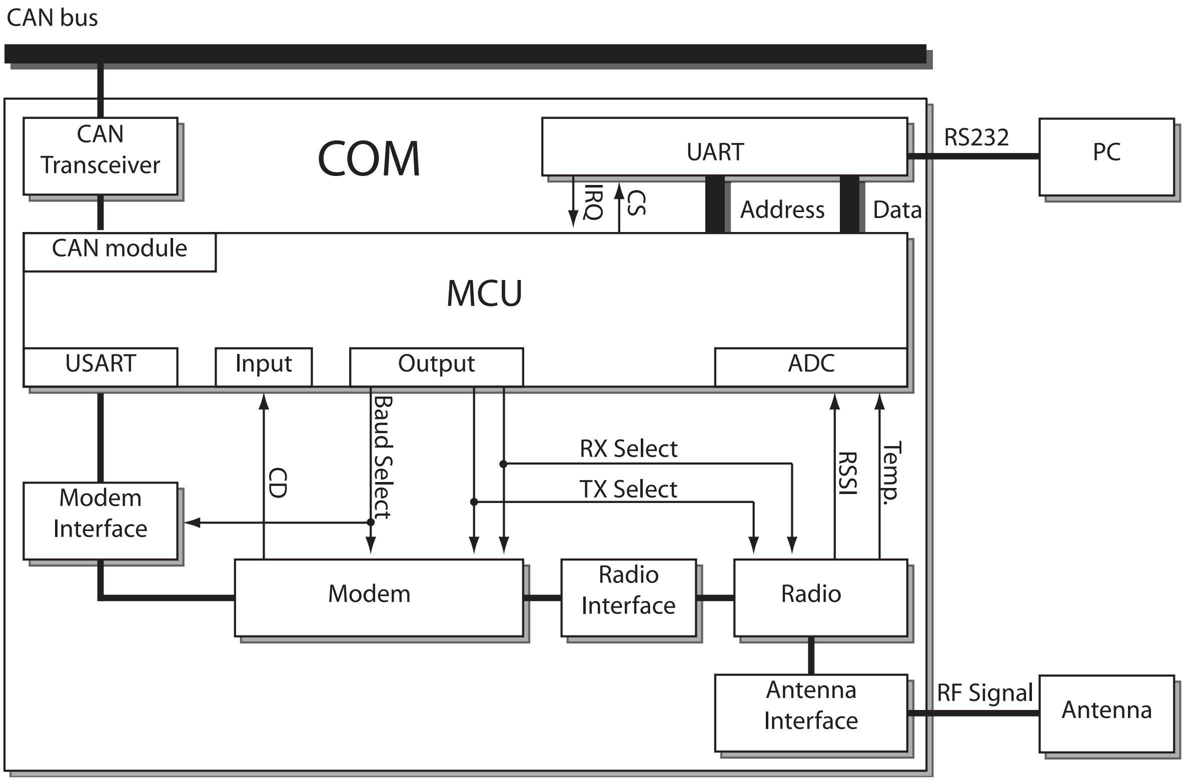


Figure 2.1: Block diagram of the COM system hardware.

The COM hardware design is primarily build up around an MCU, a modem and a radio. The MCU takes care of the data handling between the CAN bus and the radio. The modem converts the digital signals from the MCU into a modulated signal for the radio, and the inverse from the radio to the MCU. Finally the radio demodulates or modulates the signal to and from the modem.

These are the principles in the COM hardware design. Extra components have to be added to be able to interface the hardware. First thing is the interface between the CAN bus and the MCU where a CAN transceiver is needed. An interface between the MCU and the modem is also needed to be able to switch between different baud rates. Between the modem and the radio, an interface is needed to adapt the signal level from the modem to the radio and inverse. Finally an interface between the radio and the antenna has to be designed in order to optimize the link budget. An extra feature included only on the prototypes and the engineering model, but not on the flight model, is a UART used for debugging the MCU software from a PC using an RS232 interface. The UART is also used as an interface for the ground station server. This has the advantage that only one PCB has to be developed and the only difference between the space and ground segment hardware is the UART and the CAN interface.

2.2 Design

While reading the following it is recommended to fold out the circuit diagram in Appendix C on page 53 and use it as reference for the circuit descriptions.

2.2.1 Microcontroller

A PIC18LF6680 was chosen in cooperation with the other subsystem teams, as the MCU to be used, as it meets the demands for all subsystems. Since this is the heart of the COM system, the connections to the MCU will be described in the following under the descriptions of the other components.

The maximum clock frequency for the MCU is given by the following formula [PIC18LF6680, Page 414]

$$\begin{aligned} F_{max} &= (16.36 \cdot 10^6) \cdot (V_{cc} - 2) + 4 \cdot 10^6 \\ &= (16.36 \cdot 10^6) \cdot (3.3 - 2) + 4 \cdot 10^6 = 25.3\,[\text{MHz}]. \end{aligned} \tag{2.1}$$

The supply current to the MCU is 12 [mA] at 25.3 [MHz] and it can be reduced by decreasing the clock frequency. Two critical data rates must be taken into account, 4800 [baud] for ground station communication and 125 [kbps] on CAN. Since a retransmission on CAN is easier than from the ground station the 4800 [baud] is the most critical. At 4 [MHz] the supply current is 2.5 [mA] and since one of the design criteria is to use as little power as possible the clock frequency is set to 4 [MHz]. This entails an instruction frequency of 1 [MHz] and at the required data rate of 4800 [baud] that is $\frac{10^6}{4800} = 208$ instructions per bit and $208 \cdot 8 = 1664$ instructions per byte. Since the communication with the ground station is the most critical, this is estimated to be sufficient to move data back and forth between buffers in the software. If 4 [MHz] is not sufficient to handle communication on CAN and USART simultaneously the COM hardware design is ready for frequency operation up to 25 [MHz].

2.2.2 UART

The UART (U5) is added to the design to ease the implementation of software and to make it possible to use a single hardware design for both COM and the ground station. A SSCC2691 [SCC2691] UART has been selected for this, because it has a simple interface that can easily be connected to the MCU. The TX and RX FIFO buffers are both three bytes long. Port F is used as a byte-wide data port and Port E is used for control of the address to the internal registers of the UART. The read and write strobe and the chip enable signals are also connected to Port E. The interrupt pin is connected to the external interrupt pin, INT2 on the MCU. This is used to signal the MCU when there is data in the RX FIFO or the TX FIFO has space for more data.

2.2.3 Modem

As concluded in the introduction a modem is needed in order to transform the digital data stream into an FSK modulated signal that can be fed into the radio. The modem and its power consumption shall be as small as possible. It must modulate with frequencies that fit within the pass-band of the radio and have bit rates that satisfies the requirements of the data budget.

It proved to be difficult to find an FSK modem that was available in a SMD[1] package and used little power, so instead a CMX469A Fast Frequency Shift Keying (FFSK)/Minimum Shift Keying (MSK) modem was selected [CMX469A]. It can run at baud rates of 1200, 2400, 4800 [baud]. An advantage of using FFSK/MSK modulation is that it gives a better Signal to Noise Ratio (SNR) than FSK which entails a better link margin. An example of the differences between the two modulation schemes can be seen in Figure 2.2.

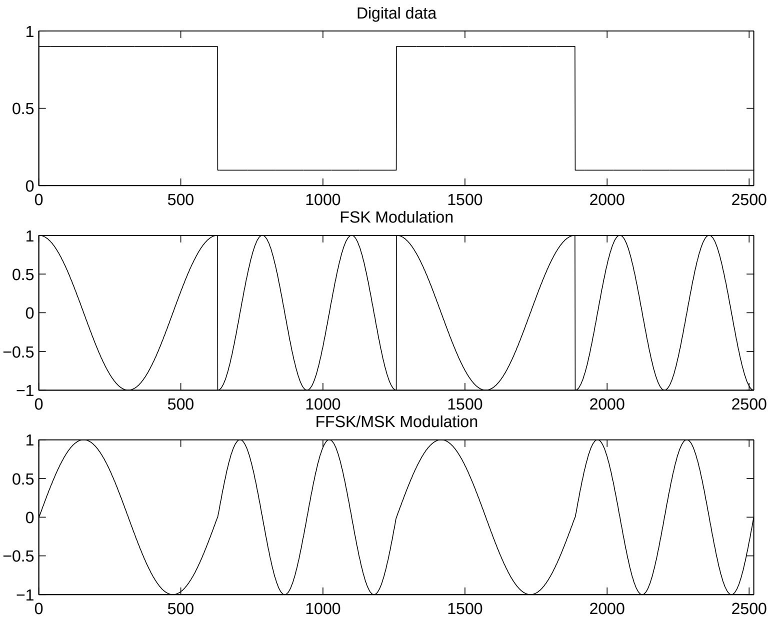


Figure 2.2: A digital signal modulated with FSK and FFSK/MSK.

With FSK modulation the signal is allowed to have a discontinuous phase where FFSK/MSK always is a continuous phased signal. In FSK modulation the information in the phase is only used to synchronize the receiver to the transmitter. With FFSK/MSK all the information in the phase is exploited in the demodulator and this increases the noise performance of the receiver significantly [Haykin, Page 378].

The pins on the modem used for serial data are both connected to the USART data pin, pin 32/DT on the MCU. This is possible because the system is half duplex meaning that the transmitter and the receiver will never be enabled at the same time. Pin 36 is connected to the USART data pin and used for receiver synchronization. The modem TX enable pin is active low and the RX enable pin is active high. They are both connected to the active high RX enable output for the radio, RB0 on the MCU. Port D is used as the modem control port and the baud rate select pins, the carrier detect output is connected to it as well. Refer to the Programmers Reference in Appendix A on page 45 for further details.

The modem datasheet [CMX469A] states that the analog signals to and from the radio shall have an amplitude of 1.5 [V_{pp}].

In order to modulate the signal correctly the modem requires that the serial data signal is synchronized with a data clock provided by the modem. Therefore the MCU USART is set to slave mode and connected to the

[1]Surface Mounted Device

modem. This way the transmitted serial data stream will be controlled by the modem. The modem has a flywheel clock generator that synchronizes to the incoming data when in receive mode.

2.2.4 Clock Divider

The purpose of the clock divider is to enable the COM system to send and receive data at lower bit rates than the ones available in the modem. This is done by dividing the data clock frequency from the modem by an integer: 1, 2, 4 or 8. This results in more than one period per bit and thus a higher energy per bit. This gives a better SNR and the receiver Phase Locked Loop (PLL) a better chance of locking on to the signal. The divider circuit consists of a dual binary ripple counter U4 [74HC393], a dual 4-input multiplexer U3 [74HC153] and an analog switch [DG2020]. The counters divide the RX and TX clocks by 2, 4 or 8. The divided clock signals are fed into the MUX together with the undivided clock signal. The control pins A and B on the MUX are connected to port D on the MCU, and used as divider select pins. Please refer to the Programmers Reference in Appendix A on page 45 for details on these signals. The two clock signals from the MUX are fed into the analog switch, which is controlled by the active high TX select line for the radio. The advantage of connecting components that have an RX and TX mode to the radio pins is that everything is synchronized, and when writing the software, the only thing to be concerned about is switching the radio RX and TX enable pins when changing between RX and TX mode.

When receiving with the divider active there is no guarantee that the divided clock is in phase with the received data and therefore an edge trigger is inserted to reset the receiver divider on rising edges of the incoming data. This ensures that the receiver clock signal is always in phase with the received data.

2.2.5 CAN Transceiver

The signals from the CAN module in the MCU are too weak to drive a CAN bus directly and therefore the signals must be amplified by a CAN transceiver. The CAN bus is a differential bus terminated with $120\,[\Omega]$ resistors. An SN65HVD230 [SN65HVD230] CAN transceiver has been selected in cooperation with the other subsystem groups. In order to minimize the electromagnetic radiation from the bus a $10\,[\text{k}\Omega]$ resistor is connected to pin 8 on the CAN transceiver, which results in a slew rate of $15\,[\text{V}/\mu\text{s}]$.

2.2.6 Radio

The radio consists of an FM receiver and transmitter that can be enabled separately. Both the receiver and the transmitter are connected to the antenna and therefore they must not be turned on at the same time as this could result in damages to the receiver input stage [AAU Radio]. The TX and RX enable inputs are both active high and connected to RB1 and RB0 and the MCU respectively.

The AF input voltage for the transmitter is $1\,[\text{V}_{\text{pp}}]$ for a $3\,[\text{kHz}]$ deviation, that is the maximum allowed deviation allocated for the AAUSAT-II frequency. The AF output voltage from the receiver is $250\,[\text{mV}_{\text{pp}}]$ at $3\,[\text{kHz}]$ deviation.

The Received Signal Strength Indication (RSSI) signal from the radio indicates the strength of the received signal and this information is to be collected as housekeeping data. Therefore it is connected to the internal A/D converter of the MCU on pin RA0.

The temperature of the radio is interesting because the receiver and transmitter frequency depends on it. Therefore a temperature sensor is placed on the heat sink of the power amplifier chip. An LM60CIZ [LM60CIZ] is selected for this because it has a wide temperature range and it gives an analogue voltage that is linearly proportional to the temperature in centigrades. The temperature voltage is connected to the internal A/D converter of the MCU on pin RA1.

2.2.7 AF Amplifier

Since the analog signal voltage levels of the radio and the modem are incompatible, an amplifier is inserted. An LM2904AD operational amplifier is selected for this because it runs from a single 3.3 [V] supply and works from rail to rail [LM2904AD]. The amplifier gains needed are listed in table 2.1.

Signal	Radio voltage	Signal direction	Modem voltage	Gain
AF-TX	1 [V_{pp}]	<-	1.6 [V_{pp}]	0.625
AF-RX	0.25 [V_{pp}]	->	1.5 [V_{pp}]	6

Table 2.1: Signal voltages and resulting amplifier gains.

The AF-RX amplifier U1A is set up in a non-inverting coupling and the gain is decided by the resistors R2 and R4. C1, R1 and R3 is a DC decoupling circuit used to ensure that the signal lies at a certain operating point. The signal from the radio is 0.25 [V_{pp}] so the operating point voltage V_{op} should be 125 [mV]. Since rail-to-rail is an approximation, a 50 [mV] offset is added to the operating point. The operating point voltage is created by the voltage divider consisting of R1 and R3. R1 is chosen to 10 [kΩ] and this yields a value of R3 of

$$\begin{aligned} R3 &= \frac{R1}{V_{op}} \cdot V_{cc} \\ &= \frac{10000}{0.175} \cdot 3.3 \approx 186\,[k\Omega]. \end{aligned} \tag{2.2}$$

A DC decoupling is a high-pass filter and therefore it is important not to filter away the modulation signals from the modem. The lowest frequency from the modem is 1200 [Hz] and since the decoupling is a first order filter, the cut-off frequency, f_c can be calculated as

$$f_c = \frac{1}{2 \cdot \pi \cdot R_f \cdot C_f}. \tag{2.3}$$

R_f is the parallel coupling of R1 and R3 and therefore

$$\begin{aligned} R_f &= \frac{1}{\frac{1}{R1} + \frac{1}{R3}} \\ &= \frac{1}{\frac{1}{10\,[k\Omega]} + \frac{1}{186\,[k\Omega]}} = 9490\,[\Omega]. \end{aligned} \tag{2.4}$$

C1 is chosen to 470 [nF] and f_c will then be

$$f_c = \frac{1}{2 \cdot \pi \cdot 9490 \cdot 470 \cdot 10^{-9}} = 35.7\,[Hz]. \tag{2.5}$$

This is well below 1200 [Hz] and therefore accepted.

The AF-TX amplifier gain must be 0.625 and U1B is therefore used as a buffer with a voltage divider on the input. The voltage divider consists of R5 and R6, and R5 is selected to 10 [kΩ]. R6 can then be calculated to

$$\mathrm{R6} = \frac{\text{R5 - R5} \cdot \text{AF-TX Gain}}{\text{AF-TX Gain}} = \frac{10000 - 10000 \cdot 0.625}{0.625} = 6\,[\Omega]. \tag{2.6}$$

The AF signal must be low-pass filtered before the radio, to prevent spurious transmissions. This is done by inserting a capacitor C2 in parallel to R5. This creates a low-pass filter with a cut-off frequency dependent on the values of R6 and C2. The PWM frequency is set to 250 [kHz] and the highest frequency from the modem is 4800 [Hz]. The 4800 [Hz] signal must not be disturbed by the filter and the 250 [kHz] PWM signal should be damped as much as possible, preferably more than 20 [dB]. The filter cut-off frequency should be placed between the two frequencies. It is therefore set to 15 [kHz] and C2 is calculated:

$$\begin{aligned} \mathrm{C2} &= \frac{1}{2 \cdot \pi \cdot \mathrm{R6} \cdot \mathrm{f_c}} \\ &= \frac{1}{2 \cdot \pi \cdot 6000 \cdot 15000} = 1.77\,[\mathrm{nF}]. \end{aligned} \tag{2.7}$$

The resulting filter consisting of R5, R6 and C2 has been simulated and the results are shown in Figure 2.3 and 2.4.

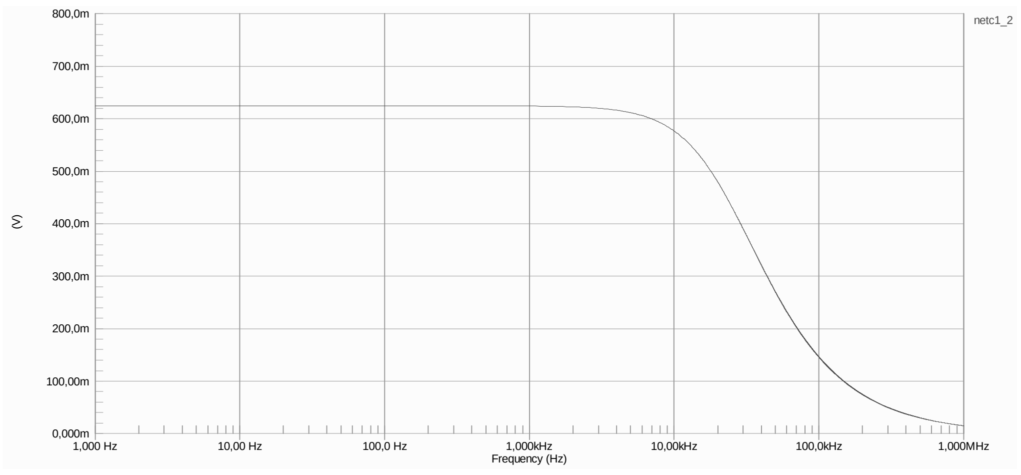


Figure 2.3: AF-TX filter simulated unit gain.

It can be seen in Figure 2.3 that the filter gain is 0.625 until the cut-off frequency and in Figure 2.4 it is shown that the gain at 250 [kHz] is -24 [dB].

2.2.8 Morse Tone Generation Circuit

Since the sine wave is generated by a PWM signal, a low-pass filter is needed to avoid transmitting high frequencies through the radio. The analog switch U7 is used to control whether the signal from the modem or the PWM generator is sent into the filter. The switch is controlled by the RB3 pin on the MCU.

Figure 2.5 shows the signal from the PWM generator filtered only by the AF-TX filter. It is very noisy and the sine curve consists of a series of steps. This indicates that a filter with a lower cut-off frequency is required.

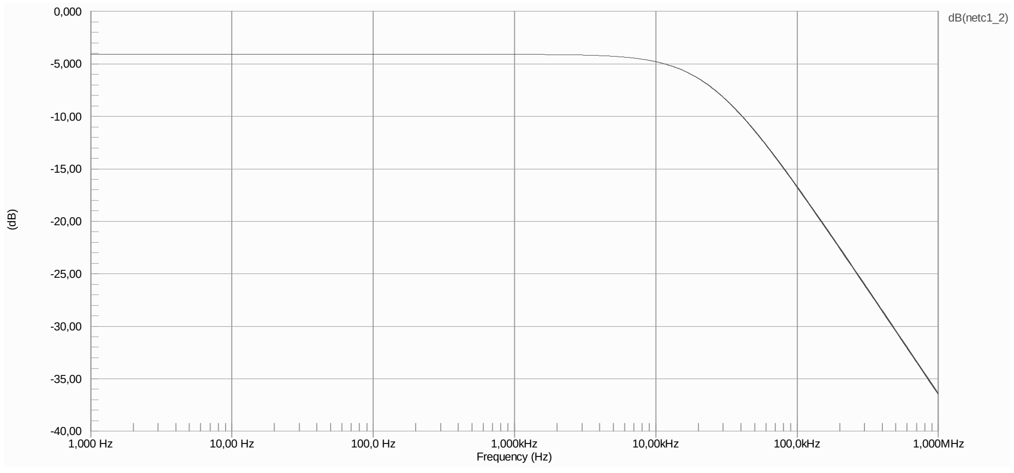


Figure 2.4: AF-TX filter simulated gain.

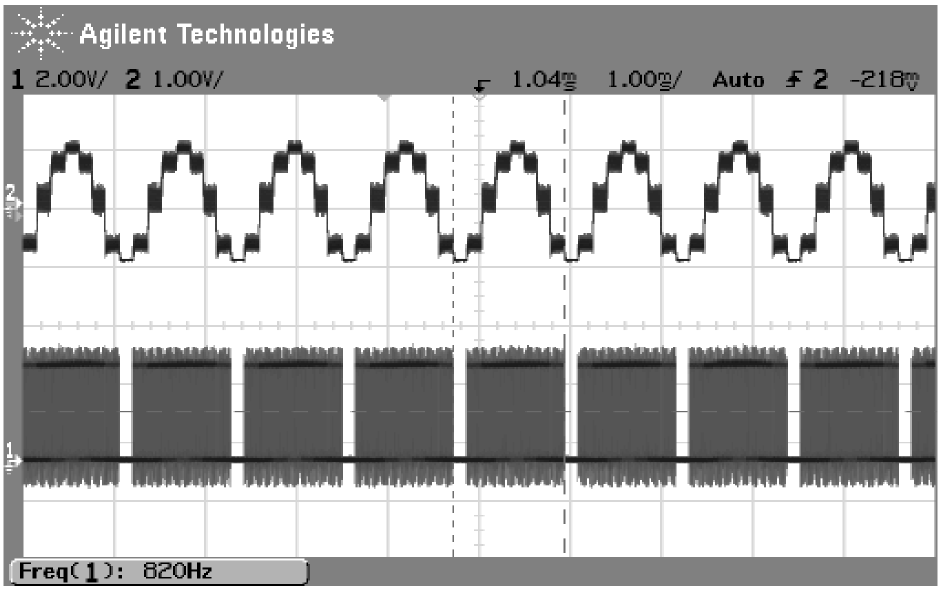


Figure 2.5: The PWM sine signal filtered only by the AF-TX filter.

A first order low-pass filter consisting of R15 and C27 is therefore inserted before U7. The desired cut-off frequency is set to 1000 [Hz] and R15 is selected to 10 [kΩ]. C27 can then be calculated to

$$\begin{aligned} C27 &= \frac{1}{2 \cdot \pi \cdot R15 \cdot f_c} \\ &= \frac{1}{2 \cdot \pi \cdot 10000 \cdot 1000} = 15.9\,[\text{nF}]. \end{aligned} \tag{2.8}$$

2.3 Test

In this section test cases and requirements are set up in order to test all hardware functionalities thoroughly. The hardware tests have been carried out during development of the low level software routines. The advantage of this is that development of software for the higher layers is less complicated when the functionality of the low level functions and hardware have been verified. The tests are then carried out whereafter the results are compared to the requirements.

2.3.1 Test Cases

1. UART Test

To test the UART the MCU must be programmed to send all received data from the UART back through the UART. A 100 [kB] file is sent from a PC to the UART and the received data is saved in another file. The two files must then be compared to determine if any errors have occurred. The test is a success if no errors have occurred and a failure if there are any errors.

2. CAN Test

The two COM boards are connected via their CAN interfaces and loaded with special software that acts as a bridge between the debug UART and the CAN bus. A 100 [kB] file is then sent from a PC to the UART on one of the boards. Then it is sent via CAN to the other MCU, and passed on to its UART and received by a PC. The received data is saved in another file. The two files are then compared to determine if any errors have occurred. The test is a success if no errors have occurred and a failure if there are any errors.

3. Clock Divider Test

To test the clock divider circuit, two tests must be carried out:

In the first test the frequency on the input and the output of the clock divider is measured at all the divisions for both reception and transmission. This can show if the frequency is divided as expected or not.

The second test should test the receiver clock synchronization circuit. To do this two COM boards should be connected and data sent with a divider of 2, 4 or 8. The receiver is set to the same divider and the MCU clock and data inputs are measured with an oscilloscope. It can then be seen if the clock is synchronous to the data or not.

4. AF Amplifier Test

Four tests must be carried out to test the AF amplifier circuit:

The first test is testing the receiver amplifier. A sinusoidal signal with an amplitude of 0.25 [V_{pp}] is applied to the input and the output is measured on an oscilloscope. The amplitude of the output signal should be 1.5 [V_{pp}].

Test number two is a test of the DC decoupling. The setup is the same as in the first test but now the offset of the input signal is varied at a rate of 1 [V/s]. If the output signal offset remains steady the test is a success.

Test number three shall test the low frequency gain of the transmission amplifier. A sinusoidal signal with an amplitude of 1.6 [V_{pp}] is applied to the input and the output is measured on an oscilloscope. The amplitude of the output signal should be 1 [V_{pp}].

The last test shall test the frequency response of the filter on the transmitter amplifier input by applying a sinusoidal signal and varying the frequency between DC and 250 [kHz]. If the damping at 250 [kHz] is more than 20 [dB] the test is a success.

5. Modem Test

To test the modem, two prototype COM boards should be connected through the modems, and 400 AX.25 frames with the maximum frame size are transmitted through the modems. The transmitted frames are compared to the received frames and if there are no errors the test is a success.

6. Morse Tone Generation Test

The Morse tone generator is started and the AF-IN input on the radio is measured with an oscilloscope. The frequency of the signal must be 800 [Hz] and the SNR must be better than 20 [dB] for the test to be successful.

7. Radio Test

To test the radio one COM board is connected to the AAUSAT-II radio and the other is connected to the ground station radio. 400 AX.25 frames with the maximum frame size are transmitted through the radios. The transmitted frames are compared to the received frames and if there are no errors the test is a success.

Later, under the functional testing phase for the satellite, the radio shall be tested at long distance to ensure that COM works with weak and/or distorted signals.

2.3.2 Test Results

1. UART Test

The source files included on the enclosed CD-ROM, include this test as it is basis for the software used for the UART module. The test file was sent and received without errors and therefore the test was successful. The file sent and the file received can be found in the folder `UART Test` on the enclosed CD-ROM.

2. CAN Test

CAN software has not yet been implemented so this cannot be tested at this time.

3. Clock Divider Test

The divisions for transmission and reception was correct and the data clock was synchronized to the received data so the test was a success.

4. AF Amplifier Test

The two low frequency signal gains were as calculated. The DC decoupling worked and the PWM filter damping was as calculated. Therefore the AF amplifier test was successful.

5. Modem Test

There were no errors in the received data so the modem test was successful.

6. Morse Tone Generation Test

Figure 2.6 shows the PWM signal after the PWM filter and the signal after the PWM filter and the AF-TX filter.

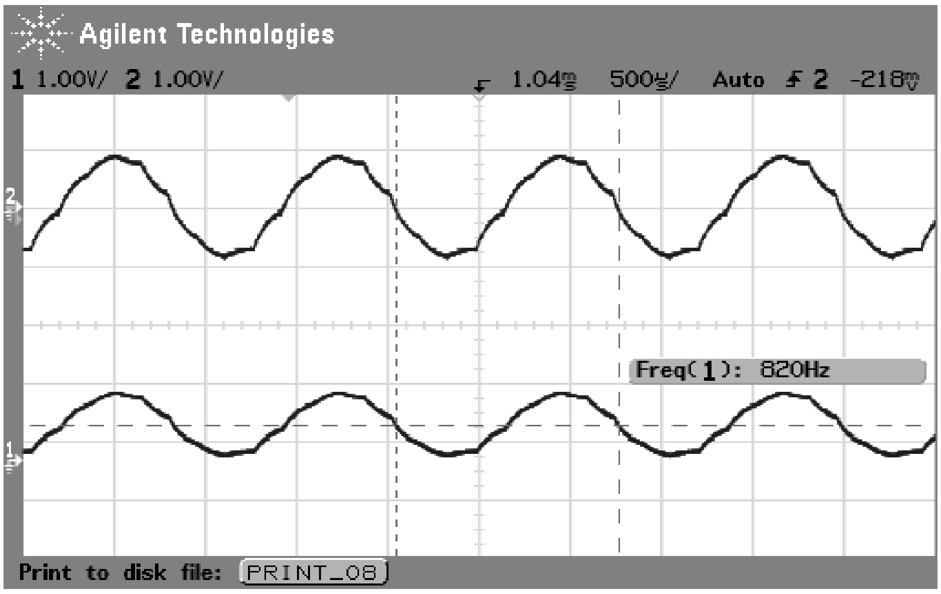


Figure 2.6: The PWM sine signal after (1) the AF-TX filter and the PWM filter, and (2) after the PWM filter only.

The frequency of the Morse tone was 820 [Hz] and the SNR was 24 [dB], hence the Morse tone generation test was successful.

7. Radio Test

All frames were received without any errors so the radio test was successful.

2.4 Conclusion

The hardware tested so far works as expected, but functional testing is needed in order to check that all circuits are performing well over longer periods of time. However this requires the software to be fully functional with all features implemented, and that point of completion has not yet been reached.

Software 3

In this chapter, the principle of the design and implementation of the COM software is described. The source files written in C and Assembler are included on the enclosed CD-ROM.

3.1 Analysis

In order to design the software for the MCU, the principles of the interfaces for the different modules are presented to establish an overview of the software for COM.

3.1.1 Software Overview

The main purposes of the software on the MCU is to pass on received data. On the satellite the CAN and the USART are used for data communication and on the ground station the UART and the USART are used. Besides that the hardware must be controlled through a number of digital output pins. Finally the basic beacon must be generated using PWM to generate a sine wave.

The software can be split into two routines. The Main routine that always runs on the MCU when no other tasks are running. The task of Main is to decode and execute incoming commands and collect housekeeping data. The other routine is the Interrupt Service Routine (ISR), that moves data between small hardware FIFO buffers and larger software buffers. Figure 3.1 shows the division of the software for the COM system and when the modules of the MCU are being used.

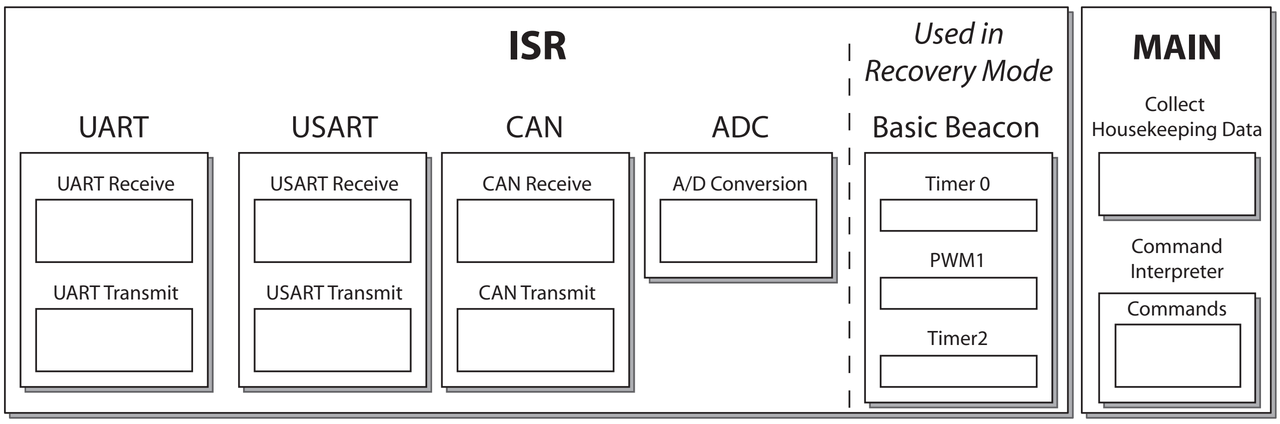


Figure 3.1: Block diagram for the COM software design showing the functions used in the software on the MCU.

The software routines are divided into modules. The ISR is divided into the UART, USART, CAN, ADC and Basic Beacon modules. The Main routine is divided into two modules, Collect Housekeeping Data and a Command Interpreter.

The internal USART in the MCU is used to transmit and receive data to and from the modem. The internal CAN module of the MCU is used for communication over the CAN bus. The ADC is used for A/D conversions of the RSSI signal and the temperature sensor signal from the radio. These are both used as housekeeping data. The UART is used for data exchange with a PC which can be used for debugging the software, and used as data interface to the ground station computer. The Basic Beacon module generates the basic beacon which is sent only in recovery mode [1] and therefore the Basic Beacon will be disabled in nominal mode. The basic beacon will be sent with regular intervals using a timer and consists of the letters "AAUSATII" sent as Morse code followed by a byte sent as binary data where the first six bits are the battery voltage and the last two bits flips every time the beacon is sent. This will help a ground station crew to determine if the system is rebooting over and over again. The tone used for the Morse coded message is generated by sending an 800 [Hz] sine wave into the radio. The sine wave is generated using a PWM signal sent through a low-pass filter, where the duty cycle of the PWM signal is changed by another timer.

The purpose of the ISR is to ensure that data is moved between small hardware FIFO's and larger software buffers with as little error and delay as possible. The purpose of the Main routine is to check data moved into software buffers by the ISR for commands and execute them.

3.1.2 External Interfaces

The external interfaces are all connections to components controlled by the MCU. These are input and output pins that control the baud rate on the modem and the clock divider circuit that enables the system to send at lower bit rates. The input and output pins also controls whether to receive or transmit on the radio by setting RX-enable or TX-enable on the modem and the radio.

Besides the control connections, three data lines are connected to the MCU through the USART module, the CAN module and the UART module.

USART Module

When receiving or transmitting data on the radio, the MCU USART is used. The USART is set to half duplex because the radio uses the same frequency for receiving and transmitting. The modem only supports synchronous communication and it is generating a data clock which the MCU must be synchronized to and therefore the USART is set to synchronous slave mode. In this mode, data is received and transmitted on the RC7/RX/DT pin on the MCU, and the clock from the modem is connected to RC6/TX/CK on the MCU. The MCU uses this clock to synchronize the data in the USART. Figure 3.2 shows the principles in the USART Transmit function which are happening in hardware on the MCU.

The main register in the USART Transmitter is the TSR which is loaded with the byte to be transmitted on the USART. It is sent bit by bit on the TX pin through the pin buffer and control on the falling edge of the clock pin. This only happens if the serial port is enabled (SPEN) and the transmitter is enabled (TXEN). All data has been sent when the TMRT register is set. However, the TMRT bit cannot generate any kind of interrupt and must be polled to ensure that all data has been sent. The TSR register is not accessible in software, therefore the byte to be transmitted is written to the TXREG register. The byte in TXREG is loaded into TSR when TSR is empty. When TXREG is empty an interrupt can be generated by setting TXIF high. If TXIE is enabled an interrupt is generated. Using this interrupt, a function can be made in the ISR to load a new byte from a buffer into TXREG. [PIC18LF6680, Page 244, 247]

A similar block diagram for the receiving part of the USART is illustrated on Figure 3.3.

Data is received on the RX pin, which is the same as the TX pin since the USART is set to half duplex [PIC18LF6680, Page 244]. Data from the RX pin is loaded to the Data Recovery block, where data is sampled

[1]See [ACS 2005, Section 1.4] for additional description of the modes for AAUSAT-II.

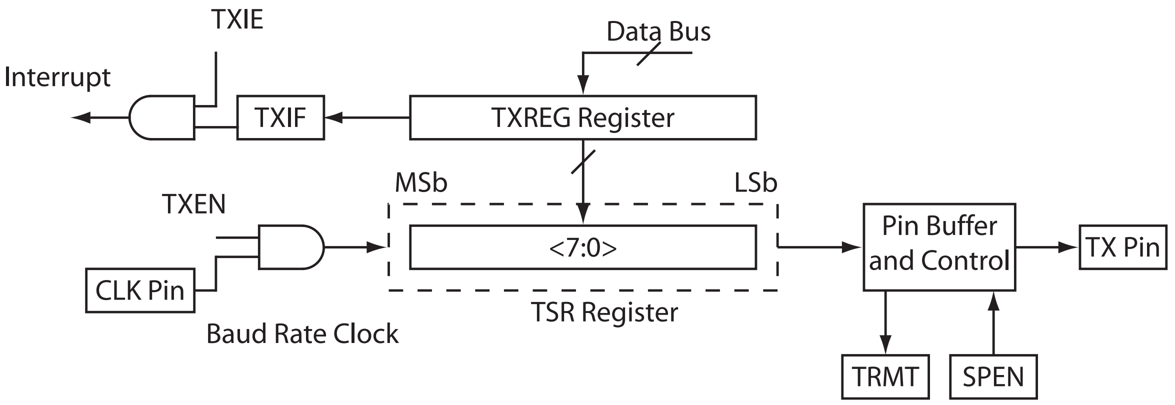


Figure 3.2: USART Transmit block diagram.

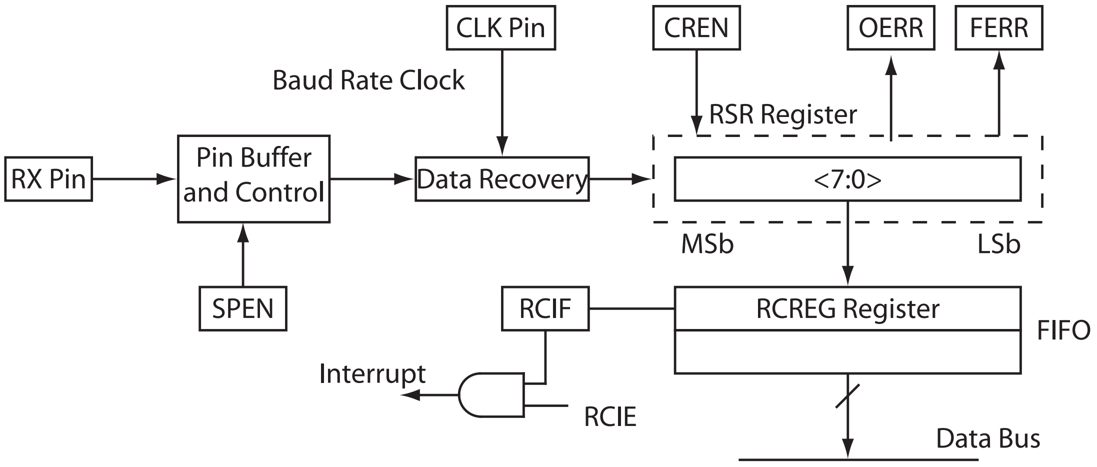


Figure 3.3: USART Receive block diagram.

on the falling edge of the clock signal. If the receiver is enabled (CREN), data from the Data Recovery block is then loaded into the RSR register. Error checking is performed on framing error and overrun error (FERR and OERR). The RSR register is not accessible from software, so the byte is loaded into the RCREG register where it can be read from software. When a byte is loaded into RCREG, an interrupt flag (RCIF) is set, if this is enabled (RCIE). Using this interrupt flag to trig the ISR, makes it possible to read from RCREG and move the data into a buffer where data can be treated in the Command Interpreter. [PIC18LF6680, Page 245, 248]

CAN Module

The CAN module is the interface to the CAN bus using the CAN transceiver. In order to ensure a safe data communication on the CAN bus, a CAN protocol has been developed by the Payload team on the AAUSAT-II project. This includes a CAN module for the MCUs used in the different subsystems, which will be "Plug and Play" software for the MCU. More information on these modules can be found on `http://www.aausatii.aau.dk/wiki/index.php/Payload_System`.

The principles in the communication on the CAN bus is that each subsystem has a unique identification number that is used for both source and destination address for each CAN packet.

When a subsystem wants to send a packet a function is called with the receiver address, a pointer to the data that has to be sent and the length of the data. To check for incoming packets a peek function in the Main routine must be executed on a regular basis.

UART Module

The UART connected to the MCU is used as a communication interface to a PC using an RS232 connection. The UART is set up by writing the appropriate settings to the configuration registers. An address bus is used to select the internal register of the UART. After setting the address, the UART register can be written by placing data on the data bus and strobing the WR pin. The UART is set up to generate an interrupt when data is ready from the PC on the RS232 connection. The MCU then sets the data bus pins as inputs and the RE pin is negated. This makes the UART place the contents of the selected register on the data bus where it can be read by the MCU. When the data has been read the RE pin is set and the read cycle is over. All these functions are taken care of in the UART module in the ISR.

ADC Module

The ADC module is needed for collecting housekeeping data from the radio. The ADC module samples the analog signals from the RSSI signal and the temperature signal from the radio into digital values which can be saved as housekeeping data in the Collect Housekeeping Data module.

3.1.3 Internal Interfaces

The internal interfaces in the software on the MCU is the interfaces between the ISR modules and the Main modules. The interfaces between these modules are buffers which are used for data exchange.

The modules in Main shall be designed such that all data exchange uses buffers. In this way, when modules in Main have data ready to be sent on the radio, the data is placed in the USART Transmit buffer and the whole transmitter function, the USART, the modem and the radio, is enabled using a function `tx_enable()`. The USART Transmit function will then send all data in the buffer, and when the buffer is empty the USART Transmit function shuts down. Receiving data from the radio is almost similar. When a byte is received from the USART Receiver, the ISR places data in the USART Receive buffer and sets a user defined flag that indicates for the Main routine that data has been received and is ready to be decoded in the Command Interpreter module in Main.

The functions in the Main routine will then read from the USART Receive buffer and decode the data from there.

The CAN and UART works in the same way as the USART by having a transmit and receive buffer for exchanging data with external components. The structure of the interfaces between the internal and external interfaces is shown in Figure 3.4.

Main Modules

The Main routine is running independently of the ISR as one function being able to call the modules in the Main routine. The modules in Main are checking flags for incoming data in the USART, CAN and the UART receive buffers. These flags will be set in the ISR to the corresponding flag and buffer. In this way the modules in Main and the external components will be able to communicate without having to wait for the receiving module to be ready.

The modules in Main control when to run the Collect Housekeeping Data and the Command Interpreter in

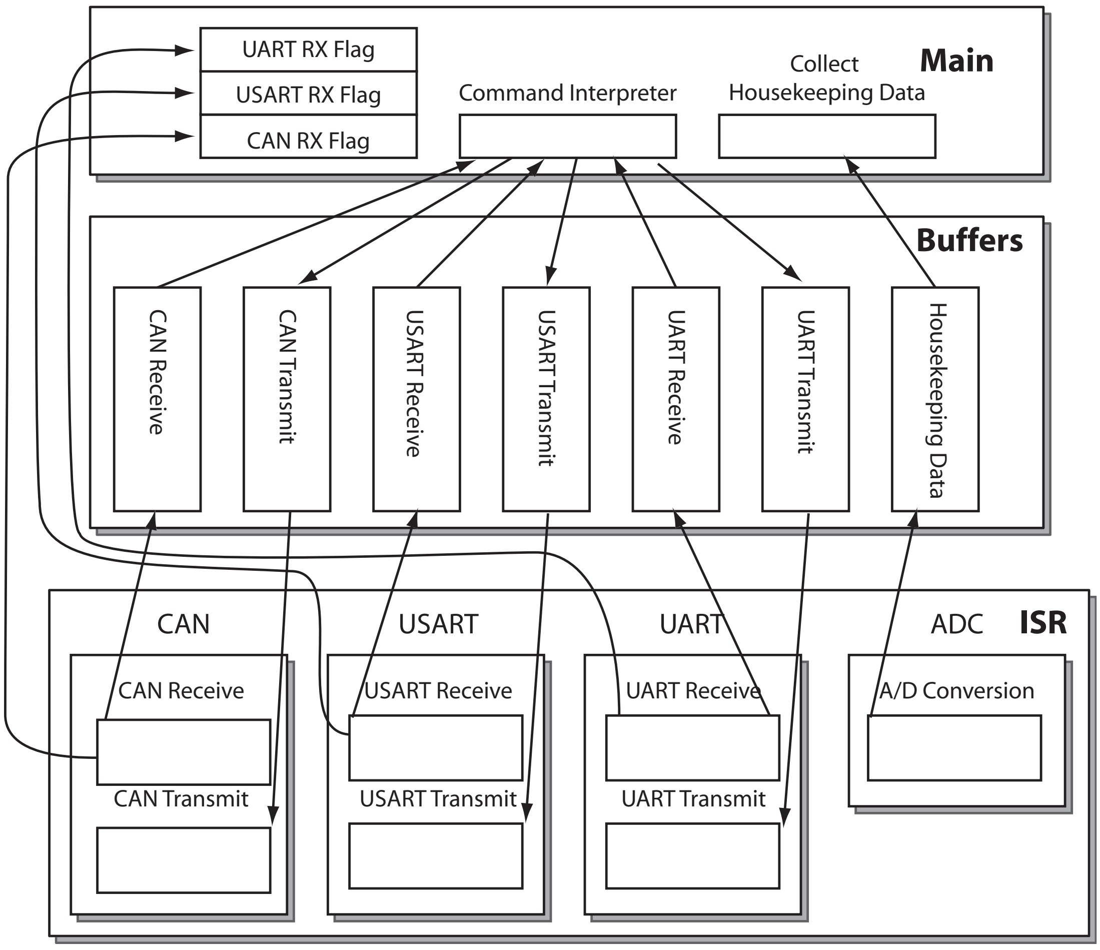


Figure 3.4: Structure of the external and internal interfaces for the software of the COM system. The arrows indicate writing access to the buffers or flags.

order to process these as described above. The commands COM can receive from other subsystems can be seen in Table 3.1.

Microcontroller

The MCU used, is the PIC18LF6680-I/PT from Microchip [PIC18LF6680]. It has 64 [kB] of flash memory, 3.25 [kB] of RAM and additionally 1 [kB] of EEPROM. In space stability of the memory is an important issue, since radiation causes bits to flip when writing to the memory, causing errors in the software on the MCU. The program memory on the MCU is placed in flash, therefore writing to flash memory must be avoided, to avoid errors in the software. This is done by placing only the program in flash memory and placing all variables and buffers in RAM. In this way the probability of a fault in the software is minimized, since the variables have not been changed in flash.

The software development tools used for programming the MCU are the MPLAB IDE with the MPLAB C18 C compiler included, both free software development tools from Microchip [Microchip]. From the source files written in C and Assembler, MPLAB IDE generates a HEX file that is written to the MCU flash memory using the MCU programming tool Broccoli18 [Broccoli18]. All software used for programming the MCU can be found together with the source files on the enclosed CD-ROM.

Command name (from EPS)	CMD	Data size [B]	Sender	Answer
COM_RECEIVE_BAT	`0x33`	1	EPS	-
COM_MODE_REC	`0xF0`	-	EPS	ACK
Command name (from CDH)	**CMD**	**Data size [B]**	**Sender**	**Answer**
COM_TRANSMIT_DATA	`0x03`	2+N	CDH	-
COM_BAUD_RATE	`0x30`	2	CDH	CMD + Baud rate
COM_HK	`0xFF`	-	CDH	CMD + Housekeeping data

Table 3.1: In this table the COM commands from other subsystems are listed. All commands are to be confirmed.

3.2 Design

The software is designed so all communication with external components is controlled by the interrupt service routine which accesses a buffer for either transmitting data or receiving data to or from each module. The modules in Main will then be able to get data and put new data into the buffers used by the modules in the interrupt service routine. The module Command Interpreter e.g. treats data from the CAN Receive buffer and puts data into the USART Transmit buffer depending on what the command received on the CAN bus is. To indicate whether there is new data to be treated in the incoming buffers, a flag will be set for each of these modules. The Command Interpreter will then check if a flag is set and if so, treat the data in the incoming buffer. In this way the modules in the ISR only take care of moving data from the external interfaces into the buffers.

3.2.1 Memory Management

The size of the buffers must fit with the requirements for the modules. The incoming buffers for both USART and CAN must have the possibility to contain one AX.25 frame with a maximum frame size, therefore the USART Receive buffer and the CAN Receive buffer size are both set to 512 [B] which is the size of 2 banks in the MCU. This also applies to the USART Transmit buffer and the CAN Transmit buffer which also each has to contain an AX.25 maximum frame size and therefore will be set to 512 [B] each. More information on the AX.25 protocol can be found in [04GR732 Article] and [Beech et al.]. The buffer used for housekeeping data is set to have 80 [B] for storing the RSSI value and the temperature of the radio according to the data budget in the ICD for COM, see Appendix B. The UART Receive buffer and the UART Transmit buffer are set to have 512 [B] each. The UART will not be implemented on the satellite and on the ground station modem the CAN module will not be implemented therefore the CAN and UART module can share memory for buffers. This of course only applies to the flight model, on the prototype and the engineering model, the UART buffer will be 64 [B] for both receiving and transmitting which will not be included in the calculations. This gives a total amount of

$$\begin{aligned}\text{Memory for buffers} &= 512+512+512+512+80 \\ &= 2128\,[\text{B}]. \qquad (3.1)\end{aligned}$$

This yields that the free memory in RAM on the MCU is

$$\begin{aligned}\text{Free memory} &= 3328-2128 \\ &= 1200\,[\text{B}]. \qquad (3.2)\end{aligned}$$

This memory can be used for different variables in the software.

The memory on the MCU is placed in different banks because the entire memory area cannot be addressed by a single byte. Since some of the buffers are larger than 256 [B] they must be allocated across more than one bank. This has been taken care of by placing two buffers of 256 [B] right after each other using `#pragma udata` which places a variable or an array at a certain address in RAM. [C18 Guide, Page 20].

3.2.2 Interrupt Service Routine

There is a high priority and a low priority interrupt vector placed at `0x000008` and `0x000018` respectively. A high priority interrupt will override a low priority interrupt. There are thirteen registers to control the interrupts. This section will only deal with the interrupts used by COM.

Interrupt Service Routine Implementation

In COM interrupts are used for USART interrupts, timer interrupts and UART debug peripheral interrupts. All interrupt sources must be set up individually in order to generate interrupts.

Setting the bit Global Interrupt Enable (GIE), located at bit 7 in the INTCON register, high, enables all interrupts globally, if the interrupts are enabled in their respective registers. Setting Peripheral Interrupts Enable (PEIE) high will enable all peripheral interrupts. GIE and PEIE are enabled in the Main routine, when the MCU has been initialized. The COM system only use high priority interrupts, therefore all interrupts branch to the high priority interrupt vector placed at the address `0x000008`. At `0x000008` a `goto high_isr` instruction in assembler is placed. This means go to the ISR to determine the interrupt source by polling the interrupt flag bits. When an interrupt is responded to, GIE is set low to disable further interrupts. The return address is pushed onto the stack and the program counter is loaded with the interrupt vector address `0x000008`. For every interrupt the corresponding interrupt flag bit must be cleared before re-enabling interrupts in order to avoid recursive interrupts. When the ISR is finished, the interrupt is re-enabled and global interrupts are re-enabled before the program counter returns to the return address on the stack.

3.2.3 Basic Beacon Module

The purpose of the Basic Beacon module is to send a radio signal that is easy to decode at a ground station. For this reason a sine signal is needed. The MCU contains a PWM module. This module can be used to generate a sine tone in combination with a low-pass filter. The idea is to use the sine tone to make a Morse signal by sending sequences of the signal corresponding to dots and dashes. For example an A in Morse code is (. -) i.e. dot pause dash.

There are some rules in Morse code. If the duration of a dot is one unit then the duration of a dash is three units. The time between the dots or dashes of one letter is one unit, between letters, three units and between words, seven units. Using the rules and the PWM module then the AAUSAT-II basic beacon (AAUSATII) can be transmitted from the satellite by sending pulses of sine waves. [IEEE]

First the sine wave must be generated by varying the duty cycle of the PWM signal from the PWM module in the MCU. In Figure 3.5 a generation of a sine wave using a PWM signal is illustrated.

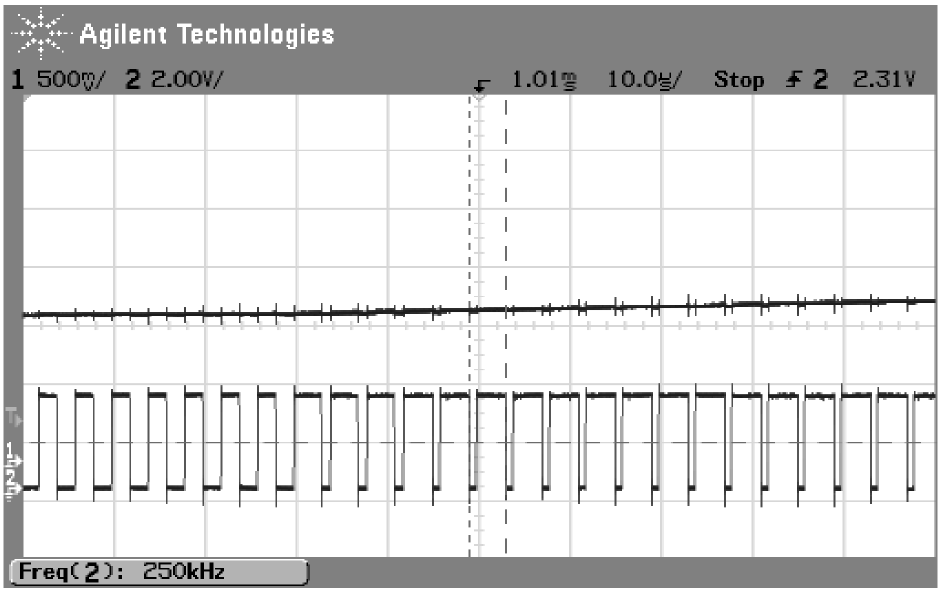


Figure 3.5: An example of a PWM signal generating a sine wave. The bottom signal is a PWM signal with a frequency of 250 [kHz] where the duty cycle is changed with 6400 [Hz]. The upper signal is the PWM signal after it has been sent through a low-pass filter.

PWM Module

The MCU's PWM module produces up to a 10-bit resolution PWM output. A PWM output has a period and a time that the output is high, i.e. the duty cycle as seen in Figure 3.6.

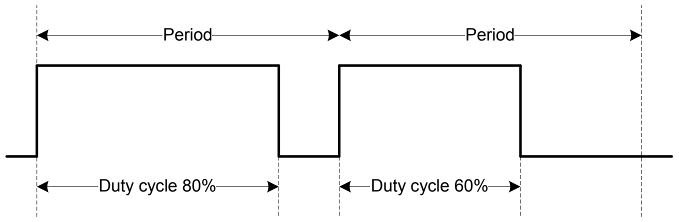


Figure 3.6: An example of a PWM signal with different duty cycle.

The frequency of the PWM signal is found by the reciprocal of the period $\frac{1}{\text{Period}}$. The PWM frequency can be calculated by Equation (3.3).

$$F_{\text{PWM}} = \frac{1}{(\text{PR2}+1)\cdot 4\cdot T_{\text{osc}}\cdot \text{TMR2}}. \tag{3.3}$$

where PR2 is the register to define the PWM period and TMR2 is the register to define the prescaler value of timer 2 in the MCU. Timer 2 is associated to the PWM module. The T_{osc} is the clock period $\frac{1}{F_{\text{osc}}}$ where F_{osc} is the MCU clock frequency.

The resolution of the duty cycle, PWM_{res} is up to 10-bits and these 10 bits are represented by the 8 MSb (Most Significant Bit) in the CCPR1L register and the 2 LSb (Least Significant Bit) in bit 4 and 5 in the CCP1CON register. The duty cycle is changed by writing a new value to the registers. Equation (3.4) defines the PWM high time. [PIC18LF6680, Page 173]

$$\text{PWM duty cycle} = \text{PWM}_{\text{res}} \cdot \text{T}_{\text{osc}} \cdot \text{TMR2}. \tag{3.4}$$

The new duty cycle value will begin from the next PWM period. It is not possible to make use of the entire 10-bit resolution because the PWM duty cycle period gets longer than the PWM period. The maximum PWM duty cycle resolution [b] for a given PWM frequency is given in Equation (3.5). [PIC18LF6680, Page 173]

$$\text{PWM max resolution} = \frac{\log(\frac{\text{F}_{\text{osc}}}{\text{F}_{\text{PWM}}})}{\log(2)}\,[\text{b}]. \tag{3.5}$$

PWM Demands

The following demands are set to get the best sine wave for the basic beacon signal and to use the properties of the PWM module.

- The sine wave frequency shall be 800 [Hz]
- The PWM frequency shall be minimum 20 [kHz]
- The PWM duty cycle resolution shall be as high as possible and still meet the other demands

The 800 [Hz] frequency of the sine wave is a commonly used frequency for Morse code and will generate a listener friendly tone. To suppress high frequencies, the corresponding signal is sent through a low-pass filter. The PWM frequency shall be as high as possible to get a good suppression factor, and furthermore a high resolution of the signal is desired to describe the sine wave as good as possible.

PWM Sine Wave Design

In order to meet the demands for the Morse tone, it is necessary to calculate a resolution for the PWM specifications. To design a sine wave from a PWM signal, the resolution of the signal is found to be a minimum of 8 values. Then the PWM frequency shall be minimum

$$800\,[\text{Hz}] \cdot 8 = 6400\,[\text{Hz}]. \tag{3.6}$$

to generate an 800 [Hz] sine wave. The PWM signal shall be higher then 100 [kHz] to get a high damping in the low-pass filter.

Setting the PWM resolution to 4 [b], the PWM frequency is calculated from Equation (3.5) to be maximum 250 [kHz]. To get this PWM frequency it is found by Equation (3.3) that the timer 2 prescaler must be set to 1 and the PWM period to 3. The 8 values resolution for the sine wave is calculated in MATLAB dividing the unit circle into 8 steps and which results in the corresponding sine as illustrated in Figure 3.7. The 8 values are converted into a resolution of 4 [b] i.e. into 8 steps from 0 to 15. The 800 [Hz] sine wave is generated by switching these values with timer 0.

A timer is used to call the function that shifts the duty cycle value. In order to get the 800 [Hz] sine wave, the timer shall change the duty cycle of the PWM signal at a frequency of 6400 [Hz].

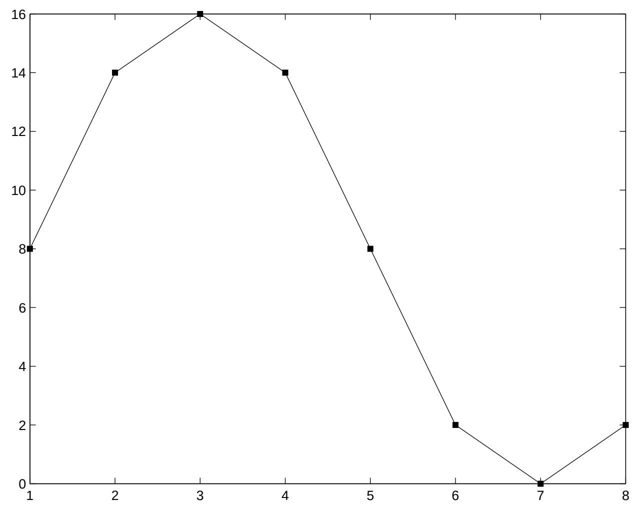


Figure 3.7: The 8 values for the sine wave.

The following steps are used to setup the PWM module for output on CCP1, port C3, at a frequency of 250 [kHz] and a start duty cycle of 0 [%].

1. The PWM period is set by writing `0b00000011` to the PR2 register.
2. The PWM duty cycle is set be clearing the CCP1L register and bit 4 and 5 in the CCP1CON register
3. Make the CCP1 pin an output by clearing the corresponding TRIS bit.
4. Set the timer 2 prescaler value to 1 and enable timer 2 by writing `0b0000100` to the T2CON register.
5. Configure the CCP1 module for PWM output

3.2.4 A/D Conversion

The MCU Analog to Digital (A/D) conversion module has 12 inputs. As described above COM only contains 2 analog signals, the RSSI measurement on the radio is connected to A/D 0 on port A0 and the temperature measurement is connected to A/D 1 on port A1. The MCU allows a conversion of an analog signal to a corresponding 10-bit digital number. The A/D conversion of the RSSI signal and the temperature will run as a part of the interrupt service routine. The procedure to do an A/D conversion is described here: [PIC18LF6680, Page 253]

1. Configure the A/D module
   - Configure analog pins and voltage reference (ADCON1) register
   - Select A/D input channel AD0 or AD1 (ADCON0)
   - Select A/D acquisition time (ADCON2)
   - Select A/D conversion clock (ADCON2)
   - Turn the A/D module (ADCON0)
2. Wait the required acquisition time
3. Start the conversion
   - Set GO/$\overline{\text{DONE}}$ bit (ADCON0 register)

4. Wait for A/D conversion to complete
   - Polling for the GO/$\overline{\text{DONE}}$ bit to be cleared
5. Read the A/D results registers
   - Read the (ADRESH:ADRESL) register
6. Next A/D conversion
   - Go to step 1 and change the port and wait a minimum of 2 T_{ad} (Defined as the A/D conversion time per bit)

3.2.5 Preamble

Since the transmission is half duplex and uses the same carrier frequency for both receiving and transmitting, the radio has to shift between receive mode and transmission mode. Shifting between the modes causes a delay, because of the internal switching circuits that have to stabilize before the radio is ready for transmitting or receiving. The radio on the satellite has a setup time of 5 [ms] for the transmitter, and 5 [ms] for the receiver [IC-910H]. The ground station radio has a setup time of 70 [ms] for the transmitter and 70 [ms] for the receiver. This means data cannot be sent before the slowest radio is ready, in all cases 70 [ms] for the radio used on the ground station. To take care of this problem, a preamble is introduced. This preamble is a series of equal bytes which is sent in 70 [ms] before the real data is sent.

The number of bytes in the preamble to be sent before sending data can be calculated as a function of the baud rate

$$\text{Preamble}\,[\text{B}] \quad = \quad \frac{\text{baud rate}\,[\text{b/s}]}{8\,[\text{b/B}]} \cdot \text{setup time}\,[\text{s}]. \tag{3.7}$$

Using Equation (3.7), the number of bytes in the preamble can be calculated to be:

- 1200 [baud] : 11 [B]
- 2400 [baud] : 21 [B]
- 4800 [baud] : 42 [B]

When using synchronous transmission on the MCU, no start and/or stop bit is available in hardware, which means if the receiver cannot detect the first bit sent but detects the second bit sent, the receiver will treat the second bit as the first bit in the received byte. This causes the whole AX.25 frame packet to be lost. The preamble byte is chosen to be in such a way that it is possible to detect if the first byte received has been received correctly. Furthermore it should also be possible to synchronize the receiver after the preamble byte. In this way the receiver will be able to synchronize the reception on the preamble, and just toss away the preamble saving only the AX.25 frame. The preamble byte shall be different from the AX.25 flag byte, which is used for indicating the start and the end of an AX.25 packet.

Based on this short analysis, the preamble byte is chosen to be `0x01` (`0b00000001`).

3.3 Flowcharts

In this section the flowcharts for the implemented functions will be described. Not all the functions have been fully designed or implemented but the flowcharts illustrates the predefined principles and functions.

The procedure for data exchange on the CAN module will not be described, as this is implemented as a plugin module designed by the team working on the Payload System.

3.3.1 Interrupt Service Routine

Figure 3.8 shows the ISR implemented in the COM System.

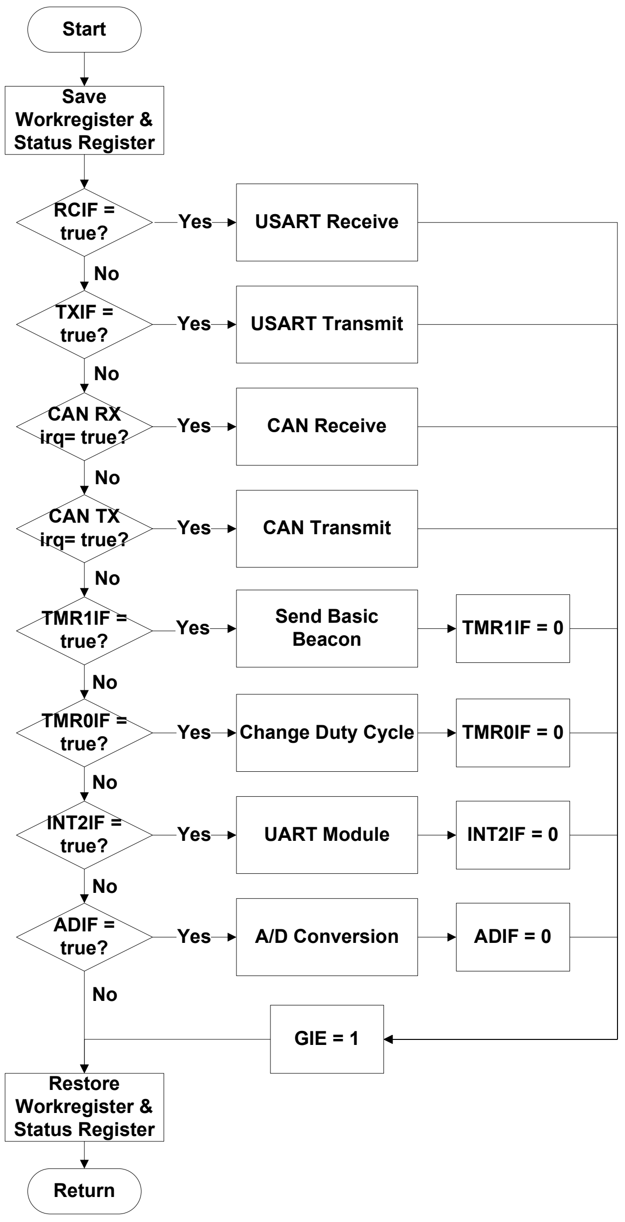


Figure 3.8: The ISR flowchart.

The ISR is described in Subsection 3.2.2 on page 26 including the MCU hardware operations combined with interrupts. When the MCU gets a high priority interrupt the work registers, status registers and the program counter are saved, and the global interrupt is enabled, since no low interrupt levels are used. Then the ISR searches for the interrupt source by polling the interrupt flag bits which are masked out in the software. This means only the interrupts used will be checked. When the ISR is finished, the global interrupt is enabled again to allow new interrupts to occur, and the work registers, status registers and the program counter are restored.

The order in which the flags are checked indicates the priority of the interrupts. The USART gets the highest priority, since this is the communication with the ground station. The receive interrupt is chosen to be higher than the transmitter interrupt. It is more important to finish the reception of an AX.25 frame from the ground station than start a transmission which cannot be received by the ground station. The ground station will still be transmitting and therefore cannot receive data. The CAN has the second highest priority since it is the communication with the other subsystems. The timers and the interrupt input pins from the UART, which are all being used internally on COM, gets the lowest interrupt priority .

If the RCIF is set, this means that there is a byte in the USART Receive buffer and the USART Receive function is called to move this data to the right buffer. By reading the RCREG register the RCIF will be cleared automatically.

If TXIF is true the TXREG register is empty and ready for a new byte to be sent. By writing a byte to the TXREG register, TXIF is automatically cleared.

The CAN interrupts are not yet completely defined by the Payload group but the illustrated CAN interrupts are the predefined interrupts from the Payload group.

If TMR1IF is set, the functions in the Basic Beacon module are called to change the values in the basic beacon. Before returning from the ISR the TMRIF has to be cleared manually. Timer 0 calls the function that changes the duty cycle of the PWM signal and works in the same way as timer 1.

The INT2 flag indicates there is an interrupt on pin RB2 from the UART, causing the UART module to be called. Before returning from ISR the INT2 flag has to be cleared.

The functions and modules called from the ISR will be described in the following subsections.

3.3.2 USART Transmit

The USART Transmit function is entered when the USART transmitter has been enabled in the Main routine. Since the transmission line is half duplex, the Main routine has to decide when to receive and when to transmit on the USART. The mode of the USART is changed using two functions `tx_select()` and `rx_select()` that takes care of the settings for the USART, the modem and the radio. These must be set correctly to avoid hardware failure. The USART, the modem and the radio have to be set in the same mode, i.e. receiving, at the same time.

When entering the USART Transmit module, a check is made to see if the USART transmitter is ready to send data which means the preamble has been sent. If the preamble has not been sent, a new preamble byte is loaded into TXREG, from where the MCU automatically transmits data. Then the counter used for the number of bytes in the preamble is decremented and the ISR is finished. When the preamble byte has been sent, a new interrupt is generated and a new preamble byte can be transmitted. This is done until the setup time has been exceeded. When the preamble has been sent, the USARTtxRdy flag is set, and a byte from the USART Transmit ring buffer is loaded into TXREG. It is then checked if the USARTtxDone flag is set, to see if all data from the ring buffer has been sent. If this flag is not set, a byte is moved from the USART Transmit ring buffer to TXREG and the ISR returns. This is done until the ring buffer has been emptied and all data has been sent. Since the USART transmitter has to be disabled before the USART Receiver can be enabled the last byte of data will not be sent if the USART transmitter is disabled right away, so an extra preamble byte is loaded into TXREG. The last interrupt will clear the flags used in the function and set the radio to listen for more data by calling `rx-select()`. In this way the USART transmitter will be disabled at the next interrupt generated when the last byte of data has been sent, and the preamble byte has been loaded into TXREG. The preamble byte will be transmitted when the USART transmitter is enabled again.

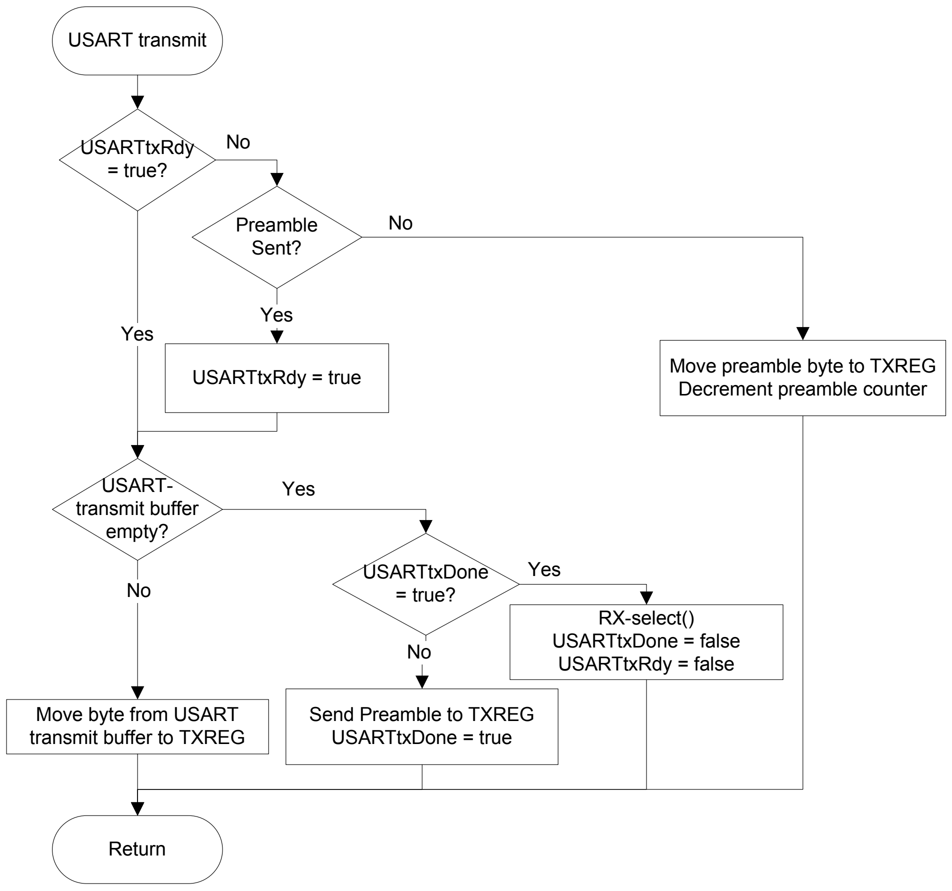


Figure 3.9: Flowchart for the USART Transmit module in the ISR.

3.3.3 USART Receiver

The flowchart for the USART Receive module is shown in Figure 3.10.

The USART Receive is called when a byte is ready in the RCREG. Then a check is made to see if the Carrier Detect (CD) is high, meaning that a carrier wave is present. If CD is not high, the synchronize flag is cleared and the data is discarded. If CD is set a check is made if the data is synchronized. If it is not synchronized, the receiver is synchronized and the flag is set so the next data received while CD is high, will be moved to the USART Receive buffer.

3.3.4 Basic Beacon Flowchart

In Figure 3.11 the design for the Basic Beacon module is illustrated by a flowchart.

Timer 1 initiates the Basic Beacon module 3 times pr. minute when the satellite is in recovery mode. The basic beacon will be sent in Morse code as described in Section 3.2.3 on page 26 therefore the Basic Beacon module transmits a predefined Morse code sequence with a big break between letters and a small break between dots and dashes. The basic beacon AAUSATII will be predefined as Morse code in an array.

- 1 for a big break
- 2 for a small break

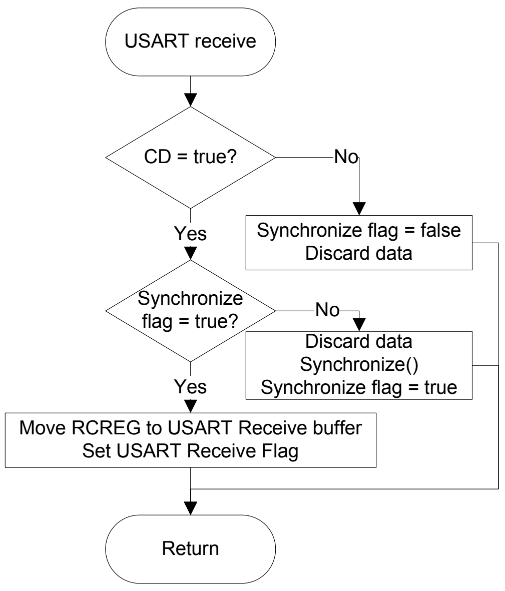


Figure 3.10: Flowchart for the USART Receive in the ISR.

- 3 for a dot
- 4 for a dash
- 5 indicates that the last received battery voltage will be send in binary

As described in Subsection 3.2.3 on page 26 a sine signal will be generated when timer 0 and 2 are running. Timer 0 changes the duty cycle in the PWM signal and timer 2 defines the PWM period, therefore the sine signal is disabled when timer 0 and 2 are closed.

When Timer 1 generates an interrupt to start transmitting the basic beacon, the function called by timer 1, will enable transmit on the radio and start to go through the predefined array and check each character for the value 1, 2, 3, 4 and 5 and then carry out the corresponding operation. If the value is 1 a big break is generated by disabling timer 0 and timer 2, a new value is written in timer 1 (`0x0000`) to define the big break period and generate the next interrupt. Before termination the array counter is incremented and it is checked if the counter has reached max. If yes the counter is reset and an ACK will be sent to the EPS subsystem to indicate that COM is alive and that a basic beacon has been sent. TX is disabled and RX is enabled on the radio in order to detect an ACK or a command from ground.

At the next interrupt the element, pointed to in the array by the counter in the Basic Beacon module, is checked for its value. If the value is 2 (small break) the same procedure as before is followed only another value (`0xFF00`) is written to timer 1, because the period is 3 times shorter than for a big break. For the value 3 a dot is sent by enabling timer 0 and 1 and writing `0xFFF0` to timer 1. For the value 4 a dash is sent by enabling timer 0 and timer 1 and writing `0xFF00` to timer 1. For the value 5 the battery voltage is sent binary by enabling transmit and moving the data to the USART Transmit buffer. The time to send the one byte battery voltage at 1200 [baud] is

$$\frac{1}{1200} \cdot 8 \approx 7\,[\text{ms}]. \tag{3.8}$$

Timer 1 is therefore set to 10 [ms]. Timer 1 has a 16-bit counter register and a prescale value from 1 to 8. Hence Equation (3.9) can be used to calculate the duration time before a new interrupt is generated.

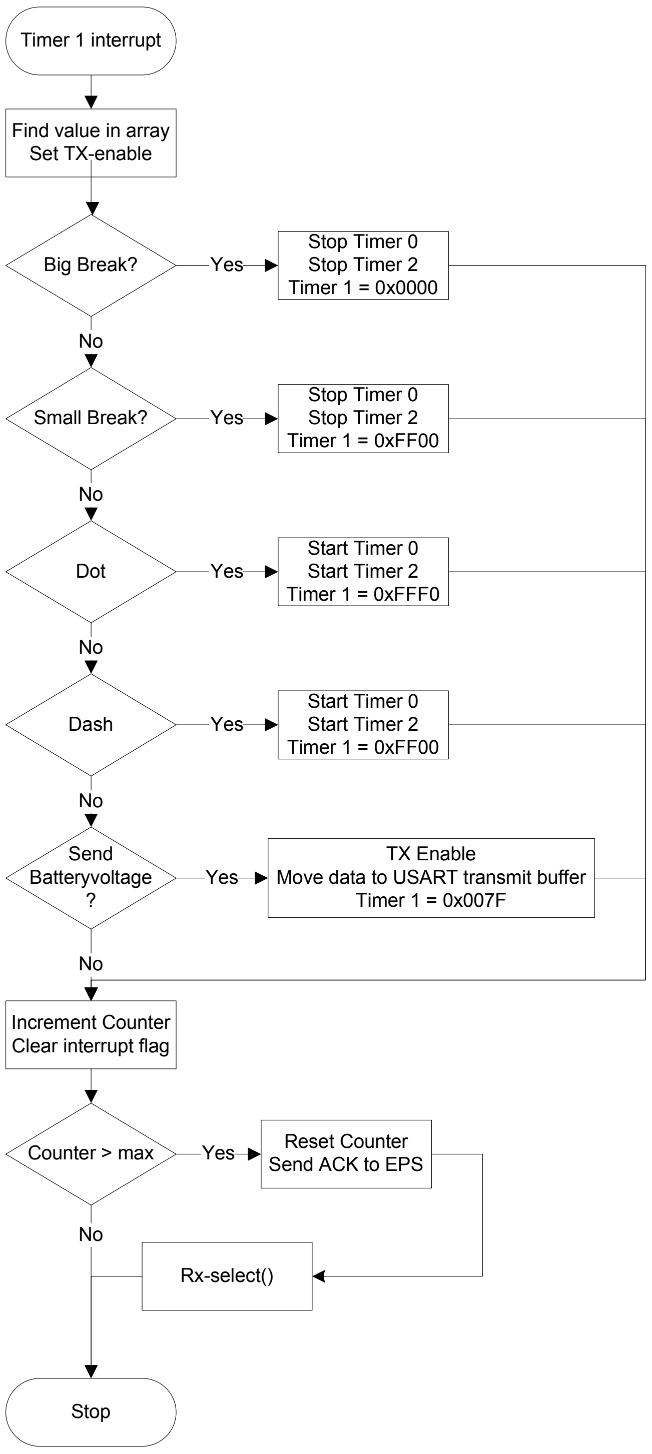


Figure 3.11: The Basic Beacon module.

$$\frac{1}{\frac{F_{ocs}}{4\cdot 16\text{bit counter}\cdot\text{prescale value}}} = \text{duration time.} \tag{3.9}$$

Where 4 is inserted because the timer only increments for every instruction cycle, four clock cycles. To get a duration time of 10 [ms] the timer 1 prescaler is set to 8 and writing (`0x04EC`) is written to the 16-bit counter register for timer 1. This will generate an interrupt after 10 [ms]. See Equation (3.10).

$$\frac{1}{\frac{4\,[\text{ms}]}{4\cdot 1260\cdot 8}} \approx 10\,[\text{ms}]. \tag{3.10}$$

3.3.5 Main

The flowchart for the Main routine is shown in Figure 3.12.

The Main routine is running on the MCU while the ISR is inactive. The purpose of the Main routine is to treat data from the incoming buffers from CAN Receive, USART Receive and UART Receive. After initialization of the MCU, the ISR is enabled. Then the Main routine checks the flags for incoming data and if none of them are set a module that collects housekeeping data is called. This module uses the A/D converter to sample RSSI and the temperature of the radio. The housekeeping data is saved in a ring buffer which is overrun when full. Hence the newest data is always kept as housekeeping data.

If the USART Receive buffer flag is set, it is checked if the AX.25 frame flag is true. If the AX.25 frame flag is not true it is checked if it is an AX.25 header frame flag which indicates the beginning of an AX.25 frame and the byte is saved in the USART Receive buffer. The following data from the USART is saved in the ring buffer, while the AX.25 frame flag is set. This is done until an AX.25 flag is received, indicating it is the end of the AX.25 frame. Having a whole AX.25 frame it is checked if this packet is to be decoded and the commands executed directly by COM or it shall be forwarded to the CAN Transmit buffer to be sent to CDH on the OBC. After having received an AX.25 frame, the AX.25 frame flag is cleared so a new AX.25 frame can be received.

If either the CAN Receive flag or the UART Receive flag is set, the data is immediately decoded in the Main routine. COM can receive 5 commands.

Command name	**CMD**	**Data size** [B]	**Sender**	**Answer**
COM_RECEIVE_BAT from EPS	0x33	1	EPS	-
COM_MODE_REC from EPS	0xF0	-	EPS	ACK
COM_TRANSMIT_DATA from CDH	0x03	2+N	CDH	-
COM_BAUD_RATE from CDH	0x30	2	CDH	CMD + Baud rate
COM_HK from CDH	0xFF	-	CDH	CMD + Housekeeping data

COM_RECEIVE_BAT is the battery voltage sent from EPS when the satellite is in recovery mode. The 6 MSb of the battery voltage are sent in the basic beacon and the 2 LSb are incremented to indicate a new battery voltage has been received from EPS. The byte is replacing the former value of the battery voltage, so it is always the latest battery voltage which is sent in the basic beacon.

COM_MODE_REC is a command from EPS to COM to start sending basic beacon again. This means the USART Receiver must be enabled and the three timers used for sending the basic beacon must be enabled too. Finally an acknowledge is sent to EPS to indicate that the basic beacon has been sent from COM.

COM_TRANSMIT_DATA is a command sent from CDH that includes data that shall be sent to the ground station. The two first bytes of data in the command tells the number of bytes to be sent. Before the data is sent, the Basic Beacon module is deactivated by disabling the three timers related to the Basic Beacon module, and call the `TX-select()` function which sets up the USART, the modem and the radio for transmission. Before moving any data to the USART Transmit buffer it is checked if the buffer is full, and it polls until all data has been moved into the USART Transmit buffer.

COM_BAUD_RATE includes 2 [B] that indicate the new baud rate and the synchronization clock division. The baud rate can be set to either 1200, 2400 or 4800. The clock divider can be set to either 1, 2, 4 or 8. Changing the baud rate or the clock divider also affects the size of the preamble which is calculated using Equation (3.7).

COM_HK is a request for the housekeeping data that has been collected. The housekeeping data is copied into the CAN Transmit buffer with the command as header, to be sent to CDH.

The commands will have to be confirmed in cooperation with the other AAUSAT-II subsystems in order to avoid conflicts on the CAN bus and to have a secure communication system.

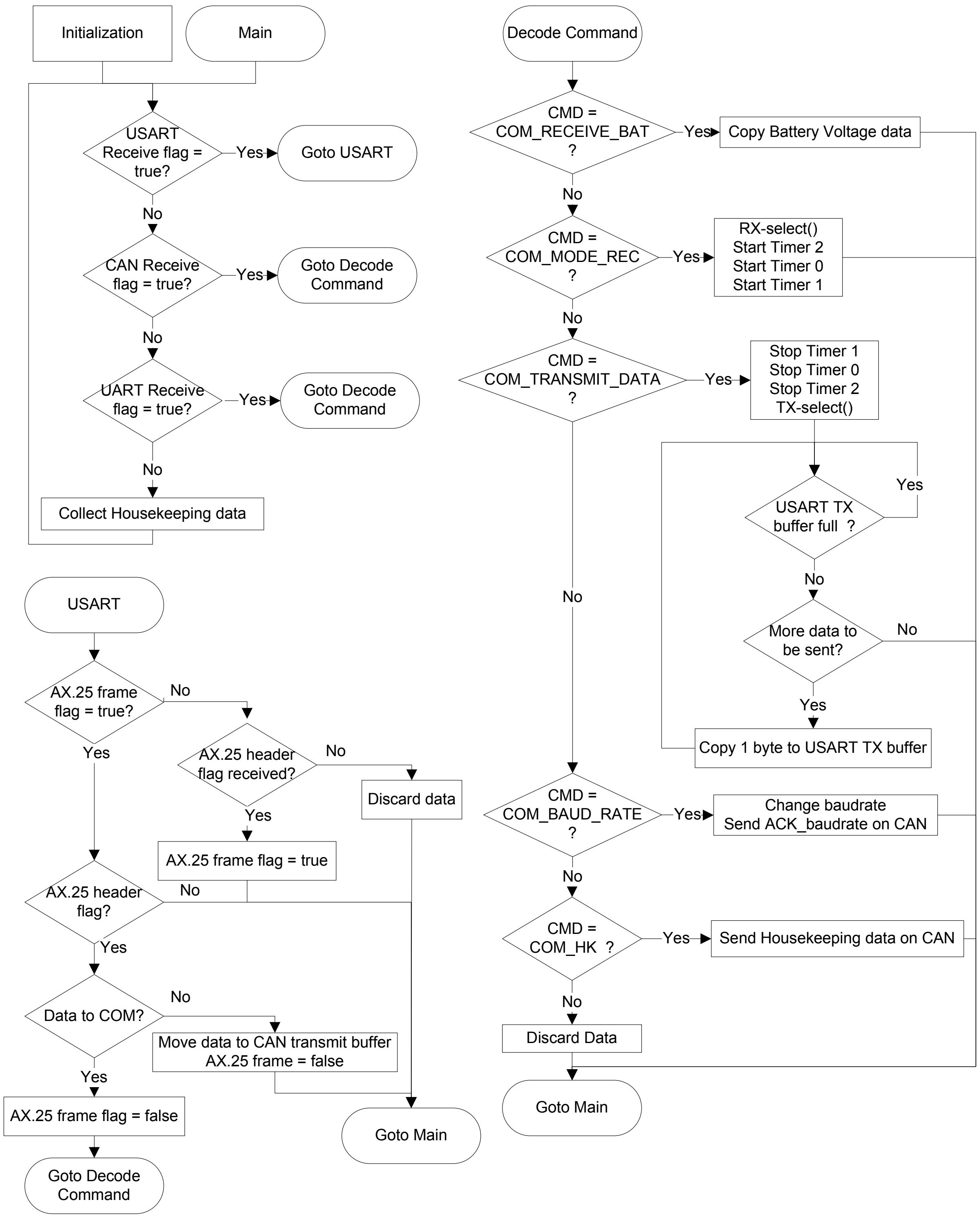


Figure 3.12: Flowchart for the Main routine on the MCU. The flowchart is split up only to make it more readable, but should be considered as interconnected.

3.4 Test

The software is not yet fully implemented, but lower level software for the hardware functions has been implemented to test the hardware. Higher level software, such as the Command Interpreter, cannot be tested at this time.

3.4.1 Test Cases

1. ISR Test

The purpose of this test is to verify that the ISR is fully functional. All interrupt flags used in the ISR must be tested, see Figure 3.8 on page 31 for details. In order to test this, events that create the single interrupts must be introduced. The interrupts that must be generated are:

- USART Transmit and USART Receive.
- UART Transmit and UART Receive.
- CAN Transmit and CAN Receive.
- Basic Beacon timer.
- A/D Converter.

2. CAN Test

To test the CAN software, special software that acts like a data bridge between the UART and CAN shall be loaded to the COM system. The CAN interface is connected to a PC with a CAN card and the UART is connected to a PC through a serial port. A file with random data is sent from the CAN PC to the serial port PC and back to the CAN PC. If the received files match the files that were sent, the test is a success.

3. Basic Beacon Test

The sine wave generation software has been tested successfully in the Morse tone generation test in Subsection 2.3.1 on page 18 and therefore it is not tested here.

In order to test the Basic Beacon the COM system must be put into recovery mode and supplied with a battery voltage from EPS or an EPS dummy. The ground station radio is set up and connected to a computer with a sound card. The received basic beacon is recorded on the computer and can then be decoded off-line. It is also possible to hear the Morse signal on the radio as the beacon is received, but the battery voltage must be decoded off-line. This is because it is sent with the modem as a single byte at 1200 [bps] and therefore it is a $\frac{8}{1200} = 6.66\,[\text{ms}]$ long pulse, that is impossible to hear. It is also impossible for the ground station modem to synchronize to a single byte so it cannot be decoded in hardware either.

For the test to be successful the Morse signal shall be clearly audible and possible to decode by listening to it. It shall also be possible to decode the battery voltage from the recording, by visual inspection of the waveform.

4. Command Interpreter Test

To do a complete test of the command interpreter it is necessary to test it with commands consisting of all possible combinations of characters, but since this will take infinitely long time another scheme is selected. Instead the test is conducted with the following types of commands:

- 1. Correctly formatted commands.
- 2. Wrongly formatted commands with too short length.
- 3. Wrongly formatted commands with too long length.
- 4. Correctly formatted commands wrapped in random garbage data.
- 5. Correctly formatted commands with random garbage data.

5. Functional Test

The functional test is a system level test carried out in order to test that the subcomponents of the COM system works together. In order to conduct a realistic functional test, two COM systems are set up, one for the satellite and one for the ground station. The system for the satellite is connected to the OBC or a PC with a CAN interface and OBC emulation software. The system for the ground station is connected to a PC with a serial port and ground station software. The communication between the two COM systems shall go through the radio and therefore the radios for the satellite and the ground station are connected to the two COM systems. The radios are then equipped with attenuators and test antennas to allow a short range test without receiver saturation. If everything works it is now possible to operate the OBC via the radio link. The test is carried out by sending tele commands (TCs) from the ground station computer to the OBC and verifying the answer. Data upload and download shall also be tested, preferably with test files containing random data.

3.4.2 Test Results

1. ISR Test

The interrupts listed in test case one was generated and the ISR reacted correctly to each of them. Therefore the test was a success.

2. CAN Test

CAN has not yet been implemented and therefore it cannot be tested.

3. Command Interpreter Test

The Command Interpreter has not yet been implemented and therefore it cannot be tested.

4. Basic Beacon Test

The Basic Beacon module has not yet been implemented and therefore it cannot be tested.

5. Functional Test

The software development is not yet finished and therefore the functional test cannot be carried out.

3.5 Conclusion

The software for the communication system has not been finished yet and therefore it has not been possible to do functional testing. The low level software routines to control the hardware was developed in order to test the hardware. The tests of these functions were all successful. The routines that have been implemented and tested, are the ISR with full support for ring buffers, the UART and the USART. The functions used for configuration of the modem, radio and clock divider has been implemented and tested. The features that has not yet been implemented is CAN, the Command Interpreter, the Basic Beacon module and the Main routine.

Conclusion 4

The COM system is one of the subsystems on the AAUSAT-II satellite that creates the basis for the satellite, just as well as the MECH, the EPS, the OBC and the CDH. Without any of these subsystems the satellite would be just another piece of space debris. For this reason every decision taken, has to be well considered in cooperation with all the other subsystems, making sure that it does not affect any of the other subsystems in a negative way. This of course means that building a satellite is a reciprocating process with lots of discussions before an actual decision can be taken.

The hardware for the COM system is currently in the prototype state and satisfactorily tested. The prototype is on a PCB of the correct size for the satellite, so it is possible to fit everything on a single PCB as required. The next step will be to design an engineering model of the hardware where the print layout will be the same as the flight model, only without the UART which is only needed at the ground station. The same layout can be used at the ground station, where CAN is removed and the UART is used for data exchange instead. All hardware has been individually tested, but no system level tests have been conducted, as the software has not been implemented yet. Furthermore the radio has to be implemented on the COM PCB. On the prototype the radio is on a separate PCB, though space is reserved for it on the COM PCB.

The low level software functions have been implemented and successfully tested. The higher levels of the software design are ready, but not yet implemented. As soon as this is done, and a system level functional test has been carried out, the engineering PCB design can be completed.

4.1 Future Work

The development of the COM system is not yet finished, but since the launch will be approximately in December 2005, the work on COM and the whole satellite will continue during the summer vacation. Remaining work to be done is implementation of software and functional testing, before the engineering PCB is ordered. The software modules still not implemented are:

- The CAN module will be implemented when Payload finishes the design of the CAN protocol. Some changes in the CAN module may be necessary, when the CAN module from Payload is adapted to the COM system. The buffer design in the CAN module differs from the buffer design in the COM System and this incompatibility issue has to be solved.
- The Basic Bacon module should be implemented in the ISR and verified.
- The USART is implemented missing only the synchronization function used for making sure the transmission is correctly received.
- The Command Interpreter is implemented, but needs to be tested for stability when the ISR modules are finished.

Bibliography

[04GR732 Article] Group 04GR732 Aalborg University. *"Optimizing Throughput in CubeSat Communication by Variable AX.25 Link Protocol Frame Size, Relative to Antenna Elevation Angle"*. 2004. /doc/article.pdf on the enclosed CD-ROM.

[ACS 2005] Group 05GR834 Aalborg University. *Attitude Control System for AAUSAT-II*. 2005. /doc/ACS.pdf on the enclosed CD-ROM.

[Eckardt] Holger Eckardt. *Email from Holger Eckardt*, 28-11-2004. /doc/Email-Holger.pdf on the enclosed CD-ROM.

[AAU Radio] Holger Eckardt. *AAUSAT-II Transceiver Unit*, March 2005. /datasheets/Radio.pdf on the enclosed CD-ROM.

[Haykin] Simon Haykin. *Communication Systems, 4th edition*. John Wiley and Sons, 2000. ISBN 0-471-17869-1.

[IEEE] IEEE. *"Virtual museum"*, May 4th 2005. `http://www.ieee-virtual-museum.org/collection/event.php?id=3456871&lid=1`.

[IC-910H] Icom Inc. *Instruction manual for VHF/UHF All mode Transceiver IC-910H*, 2000. Datasheets/IC-910H Instruction Manual.pdf on the enclosed CD-ROM.

[SN65HVD230] Texas Instruments. *3.3V CAN transciever*, June 2002. /datasheets/SN65HVD230.pdf on the enclosed CD-ROM.

[LM2904AD] Texas Instruments. *Dual operational amplifiers*, September 2004. /datasheets/LM2904AD.pdf on the enclosed CD-ROM.

[CMX469A] CML Microcircuits. *"Datasheet for CMX469A - 1200/2400/4800 FFSK/MSK Modem"*, 2001. /datasheets/cmx469ad.pdf on the enclosed CD-ROM.

[PIC18LF6680] Microchip. *"PIC18F6585/8585/6680/8680 Data Sheet"*, 2004. /datasheets/PIC18F6680.pdf on the enclosed CD-ROM.

[Microchip] Microchip. *MPLAB IDE and MPLAB C18 - Development Tools from Microchip*, April 2005. `http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=81`.

[74HC393] Philips Semiconductors. *Dual 4-bit binary ripple counter*, December 1990. /datasheets/74HC393.pdf on the enclosed CD-ROM.

[74HC153] Philips Semiconductors. *Dual 4-input multiplexer*, December 1990. /datasheets/74HC153.pdf on the enclosed CD-ROM.

[SCC2691] Philips Semiconductors. *Universal UART*, September 1998. /datasheets/SCC2691.pdf on the enclosed CD-ROM.

[LM60CIZ] National Semiconductor. *LM60CIZ Temperature sensor*, July 2001. /datasheets/LM60.pdf on the enclosed CD-ROM.

[DG2020] Vishay Siliconix. *Low voltage single SPDT analog switch*, August 2001. /datasheets/DG2020.pdf on the enclosed CD-ROM.

[Broccoli18] Dave Sullin. *Broccoli18: PIC18F Programming Tools*, April 2005. `http://home.earthlink.net/~davesullins/software/pic18f.html`.

[C18 Guide] Microchip Technology. *MPLAB C18 C Compiler User's Guide*, 2004. /programming software/MPLAB doc/C18_userguid.pdf on the enclosed CD-ROM.

[CPSU] California Polytechnic State University. *CubeSat Program*, 2005. `http://cubesat.calpoly.edu/_new/index.html`.

[Beech et al.] Jack Taylor William A. Beech, Douglas E. Nielsen. *"AX.25 Link Access Protocol for Amateur Radio"*, 1998. /doc/AX25.2.2.pdf on the enclosed CD-ROM.

A Programmers Reference

```
COM subsystem signal description/programmer reference

-===Radio===-

RSSI:
Connected to: RA0/AN0, pin 24
Description: Analog voltage between 0.95V to 2.05V indicating the received signal strength

RX_Enable: Shared with modem
Connected to: RB0, pin 48
Description: Active HIGH control signal. Shall never be HIGH while TX_Enable is HIGH.
Procedure for activating this signal: Both TX_Enable and RX_Enable shall be set LOW
before RX_Enable is set HIGH.

TX_Enable: Shared with modem
Connected to: RB1, pin 47
Description: Active HIGH control signal. Shall never be HIGH while RX_Enable is HIGH.
Procedure for activating this signal: Both TX_Enable and RX_Enable shall be set LOW
before TX_Enable is set HIGH.

-===Modem===-

Modem/PWM Select:
Connected to: RB3, pin 45
Description: Switches the radio input between modem signal and PWM signal from PIC. Is used
while sending Morse tones in basic beacon mode. A LOW signal selects the modem and a HIGH
signal selects the PWM.

PWM for Basic Beacon Morse code:
Connected to: RC2/CCP1, pin 33
Description: PWM signal for the beacon Morse tone. The PWM signal shall be at the highest
possible frequency and be  modualted with a 800 Hz tone.


Truth table for modem data rate selection (WITHOUT overmodulation):
Modem shall be set to 4800 bps to overmodulate

        |Clock rate |1200/2400 select |   4800 select
-----------------------------------------------------------------------------------
1200bps |   1       |       1         |       0
2400bps |   1       |       0         |       0
4800bps |   1       |       0         |       1
9600bps |   0       |       0         |       0     # Modem overclocked - NOT recommended
            4               16                8

1200/2400 baud select
Connected to: RD4/PSP4, pin 52
Description: Logic 1 to select 1200bps, logic 0 to select 2400bps or 4800bps
```

4800 baud select
Connected to: RD3/PSP3, pin 53
Description: Logic 1 to select 4800bps, logic 0 for 1200bps or 2400bps

Clock rate
Connected to: RD2/PSP2, pin 54
Description: Logic 1 to select 4.032 MHz crystal, logic 0 to select 1.008MHz crystal.

Carrier Detect O/P
Connected to: RD0/PSP0, pin 58
Description: Carrier detect signal from the modem

Clocked Data O/P
Connected to: RC7/RX/DT, pin 32
Description: Serial data from the modem

Tx Data I/P
Connected to: RC7/RX/DT, pin 32
Description: Serial data to the modem

Serial data clock:
Connected to: RC6/TX/CK, pin 31
Description: Receive RX/TX synchronisation clock from modem

-===Overmodulation circuit===-

Truth table for clock divider

Divider	S1	S0
1	0	0
2	0	1
4	1	0
8	1	1

B (S1) (pin 2):
Connected to: RD7/PSP7, pin 49
Description: Data select S1

A (S0) (pin 14):
Connected to: RD6/PSP6, pin 50
Description: Data select S0

-===Debug UART===-

UART_INTR:
Connected to: RB2/INT2, pin 46
Description: Active LOW interrupt line from debug interface UART. Interrupt events can be programmed in the Interrupt Mask Register, please refer to the SCC2691 datasheet for details.

Read_Enable:
Connected to: RE0/RD, pin 2
Description: Active LOW read enable. Shall be programmed to be automatically activated when reading from port F.

Write Enable:

Connected to: RE1/WR, pin 1
Description: Active LOW write enable. Shall be programmed to be automatically activated when writing to port F.

A0,A1,A2:
Connected to: RE3,RE4,RE5, pin 63,62,61
Description: Address bus for the UART used to select internal UART registers. Please refer to the SCC2691 datasheet for details.

UART Data bus:
Connected to: Port F, pins 11-18
Description: Data is transfered back and forth between UART and PIC on port F.

Chip_Enable:
Connected to: RE6, pin 60
Description: Active LOW chip enable. Activate this signal to enable UART data transmissions.

Interface Control Documents for COM

B

B.1 External ICD

```
COM ICD

-=Harness=-

Stack connector:
  - Connector type: TBD
        Pin 1: TBD
        Pin 2: TBD
        Pin 3: TBD
        Pin 4: TBD
        Pin 5: TBD
        Pin 6: TBD
        Pin 7: TBD
  - Placement: TDB
  - Destination: TBD

Antenna Connector:
  - Connector type: MMCX
        Pin 1: Signal
        Pin 2: GND
  - Placement: TBD
  - Destination: Antenna

-=Mech=-

Mass:    110g incl. antenna

The radio shielding might need a thermal connection to the satellite structure.

-=Electrical=-

Channel 2, 3.3V

Power consumption:

Max consumption [mW]:
 Off:           0
 Listening:     184.8
 Transmitting:  2085.6
 Receiving:     283.8

-=Communication=-
```

COM destination/source address: TBD

Supported commands on CAN in Nominal mode:

Mnemonic:	CAN priority	CMD	Data size	Description:
COM_HK	TBD	0xFF	0	COM returns housekeeping data, the last 64 bytes of RSSI and temperature data. Used as COM heart beat.
COM_TRX_DATA	TBD	0x03	2+n	COM transmits data to GND. Specification TBD
COM_REC_MODE	0	0xF0	0	COM changes mode to recovery mode if 0xF0 is received from EPS. COM responds by with an ACK
COM_BAUD_RATE	TBD	0x30	2	Changes baud rate and clock divider settings:

BAUD_RATE settings

First byte		Second byte	
Baud rate	byte value	Clock divider setting	Byte value
9600	00	Divide by 1	00
4800	10	Divide by 2	01
2400	04	Divide by 4	02
1200	14	Divide by 8	03

Supported commands on CAN in Recovery mode:

Mnemonic:	CAN priority	CMD	Data size	Description:
COM_TRX_DATA	TBD	0x03	2+n	COM changes mode to nominal mode and sends data to ground station
COM_BAT_VOLT	1	0x33	1	COM receives battery voltage from EPS to transmit in basic beacon

Supported commands directly from GND in both nominal and recovery mode:

Mnemonic:	CMD	Data size	Description:
PING_RSSI	TBD	TBD	COM returns RSSI data package and/or more(TBD) to GND

Data budget:

RSSI data:	80B
Acknowledge to EPS:	1B

B.2 Internal ICD

Internal ICD for COM on AAUSAT-II

```
Internal interfaces:
=========================================================================================

--CAN-bus--> PIC18 <--RS232 async--> ASYNC/SYNC <--RS232 sync--> MODEM <--FFSK-->RADIO<-
-437.425MHz-->Antennas

Satellite modes and functions of COM:
=========================================================================================

Recovery mode: COM receives battery voltage from EPS and sends it in the basic beacon.
COM listens for incoming ping requests from Earth. COM responds to the ping request by
transmitting last RSSI data to Earth.

Basic Beacon: AAUSATII<battery voltage>
AAUSATII is sent as an 800Hz Morse code followed by the battery voltage from EPS sent
binary at 1200 baud FFSK

Nominal mode: COM listens for data from GND and sends data from CDH to on the radio.

CDH sends data to COM to be transmitted on the radio. This includes the advanced beacon
from CDH.

Internal Modes for COM
==========================================================================================

Off:             Turned off.
Listening:       Listening on radio for incoming data from GND. Listening on CAN.
Transmitting:    Transmitting data to GND. Listening on CAN.
Receiving:       Receiving data from GND. Listening on CAN.

Budgets for COM ->TBC
==========================================================================================
Power Budget for COM
------------------------------------------------------------------------------------------
|Power: 3.3V bus (Ch. 2 from EPS)   |Max current incl margin    |Peak values|
------------------------------------------------------------------------------------------
|Off:                               |  0mA                      |           |
|RX:                                |103mA max                  |TBD        |
|TX:                                |660mA max                  |TBD        |
|Listening:                         | 69mA max                  |TBD        |
------------------------------------------------------------------------------------------

Mass budget for COM ->TBC
------------------------------------------------------------------------------------------

Radio, modem, PIC, CAN transceiver, PCB etc.                    | 50 g (TBC)|
Antennas + feed lines                                           | 40 g (TBC)|
Shielding                                                       | 20 g (TBC)|
------------------------------------------------------------------------------------------
Total                                                           |110 g (TBC)|
------------------------------------------------------------------------------------------

COM<->GND -=> Confirmed
******************************************************************************************
Link: 437.425 MHz FFSK radio link with right hand circular polarization modulated with
1200/2400/4800 baud
Protocol: AX.25, half duplex
Communication is controlled from GND

Interfaces to other subsystems on the AAUSAT-II
```

```
=========================================================================================
COM only communicates with EPS and CDH on the CAN BUS

COM<->EPS
*****************************************************************************************
Power: +3.3 V DC. (Not allowed under 3.1 V because of the radio)
       Max 660 mA when transmitting.

CAN bus
EPS CAN ID 0

EPS -> COM
-----------------------------------------------------------------------------------------
|Message name    |Message description                         |Data size   |CMD
-----------------------------------------------------------------------------------------
|COM_BAT_VOLT    |Battery voltage for basic beacon (2 LSb is
                    flipped by EPS)                           |1 Byte      |0X33
|COM_REC_MODE    |Sends the modem into Recovery mode          |0 Byte      |0xF0
-----------------------------------------------------------------------------------------

COM -> EPS
-----------------------------------------------------------------------------------------
|Message name       |Message description                      |Data size   |CMD
-----------------------------------------------------------------------------------------
|ACK                |ACK That COM is alive                    |0 Bytes     |0xAA
-----------------------------------------------------------------------------------------

CDH <-> COM
*****************************************************************************************
CAN BUS
CDH CAN ID 1

CDH -> COM:
-----------------------------------------------------------------------------------------
|Message name        |Message description                         |Data size |CMD
-----------------------------------------------------------------------------------------
|COM_BAUD_RATE       |Set baudrate and clock divider              |2 Bytes   |0x30 TBC
|COM_HK              |Requests housekeeping data                  |0 Bytes   |0xFF TBC
|COM_TRX_DATA        |Send data to ground station(length+data)    |2+N Bytes |0x03 TBC
-----------------------------------------------------------------------------------------

------------------- BAUD_RATE settings ----------------------------
      First byte       ||             Second byte
-------------------------------------------------------------------
Baud rate | byte value || Clock divider setting | Byte value
-------------------------------------------------------------------
   9600   |    00      ||     Divide by 1       |    00
   4800   |    10      ||     Divide by 2       |    01
   2400   |    04      ||     Divide by 4       |    02
   1200   |    14      ||     Divide by 8       |    03

COM -> CDH:
-----------------------------------------------------------------------------------------
|Message name       |Message description                      |Data size      |CMD
-----------------------------------------------------------------------------------------
|COM_HK             |COM housekeeping data to CDH
                     (length + data)                          |1+n bytes      |0xFF
|COM_DATA_FROM_GND  |incoming data from GND to CDH
                     (length + data)                          |2+n bytes      |0xA0
|ACK_BAUD_RATE      |ACK cmd and the new Baudrate             |1 byte         |0x30 TBC
```

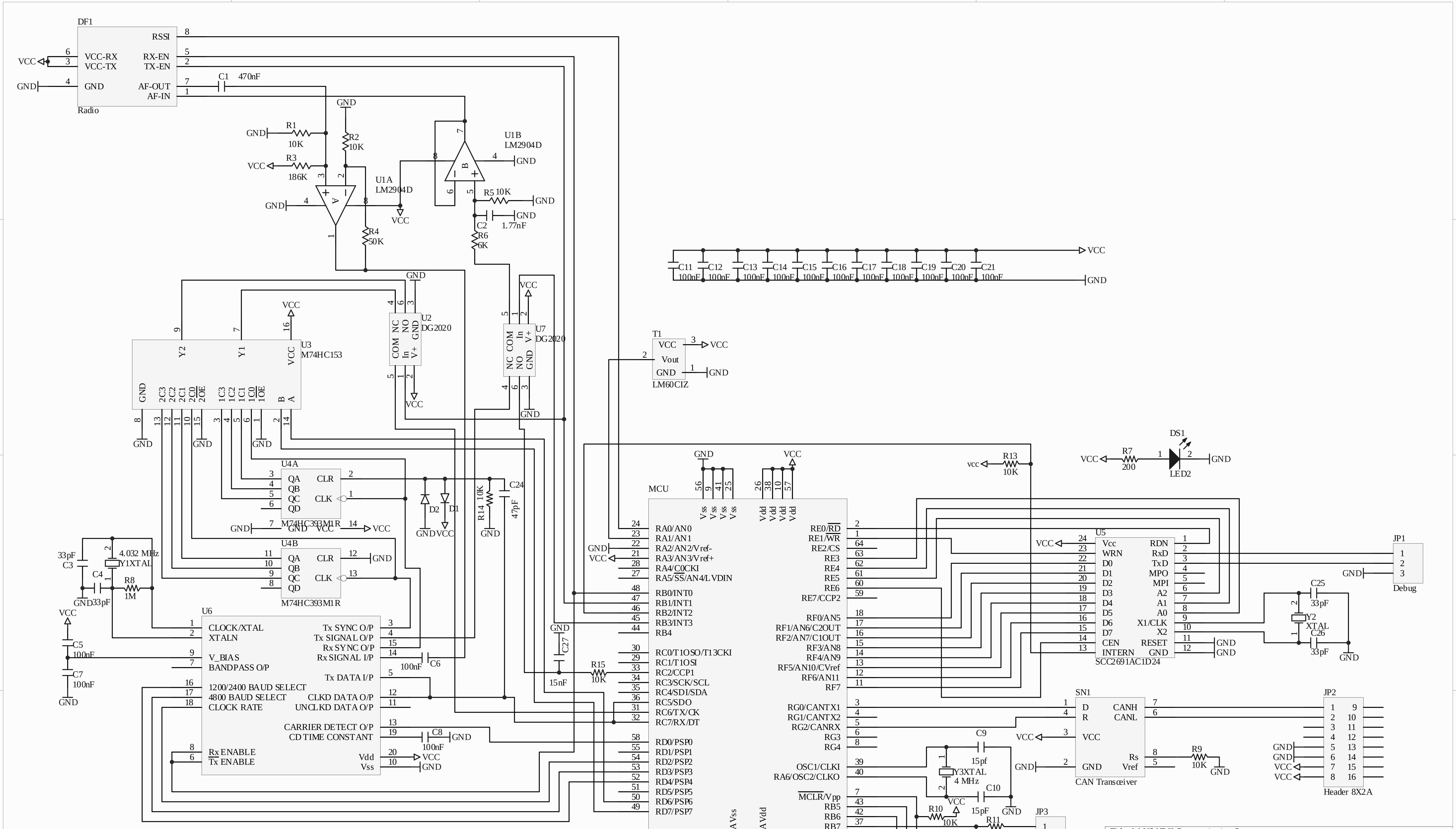
```
------------------------------------------------------------------------------------------

COM<->OBC
******************************************************************************************
CAN bus

COM<->MECH
******************************************************************************************
PCB 87 mm x 87 mm - minus mounting area for bolts
Se tecnical drawing ftp://ftp.control.aau.dk/subsystems/mech/PCB.pdf
Modem
PIC
Op-amp
CAN transceiver
Radio
Antenna
==========================================================================================
```

C Circuit Diagram and PCB Layout

The schematic of the COM system can be found on the backside of this page.



Title AAUSAT-II Communication System		
COM Schematic		
Size B	Number	Revision 5.2
Date: 27-05-2005		Sheet1 of 1
File: C:\COM\Schematics\COM.SchDoc		Drawn By: 05GR834

The prototype to the PCB layout has been made. The prototype layout is fully functional though the layout will change in the 4-layer engineering PCB, as power planes will be included in the two layers in the middle of the four layers. The prototype PCB is shown in Figure C.1 and C.2.

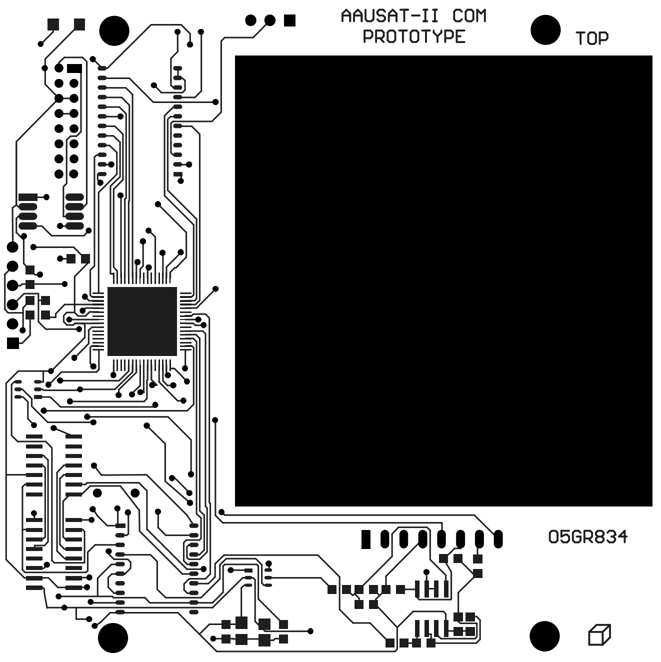


Figure C.1: Top view of the prototype PCB top layer. Scale 1:1.

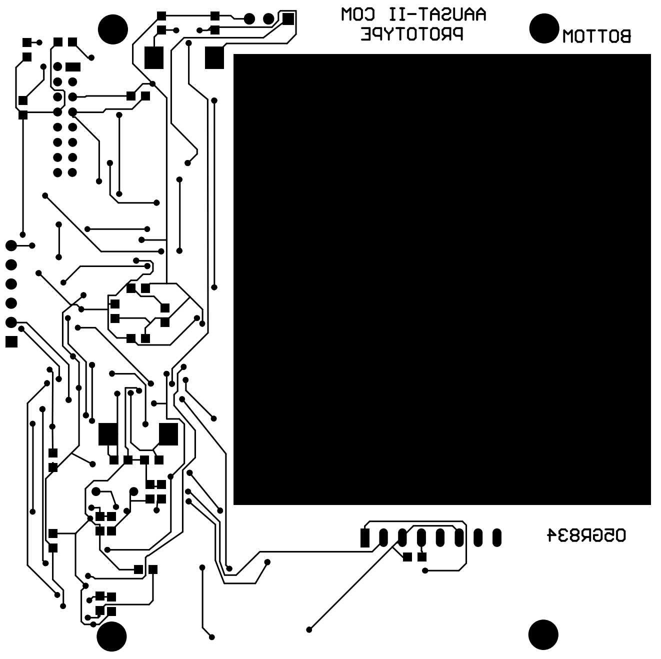


Figure C.2: Top view of the prototype PCB bottom layer. Scale 1:1.