DESIGNING ON BOARD COMPUTER AND PAYLOAD FOR THE AAU CUBESAT

Thomas Bruun Clausen, Allan Hedegaard,
Kristian Budde Rasmussen,
Rasmus Löwenstein Olsen, Jørgen Lundkvist,
Peter Egtoft Nielsen

Abstract: This article describes the design of the On Board Computer (OBC) and payload camera for the AAU Cubesat due to launch November 2002.
The OBC was designed to provide a platform for the Control and Data Handling (CDH) program and to interface the sattelite units to CDH. The OBC was based on the Siemens C161PI 16-bit microprocessor utilizing 4MB of external RAM. The OBC was designed to use both a PROM and a flash memory for the boot-up software, enabling the use of a fail safe mode. This lowered the chances of faults during boot-up and enabled the possiblity of uploading new software after launch of the satellite. Due to the environment in space and the constrains of the satellite, different measures had to be taken when the system was designed.
To ensure robustness of the OBC, components were chosen with industrial specifications, small packages and low powerconsumptions enabling the OBC to operate in an extended temperaturerange using minimal resources. The camera unit was developed by the company Devitech and interfaced via the internal bus-system to the MCU. To ensure the correct timing when taking pictures, a hardware unit allowing Direct Memory Access (DMA) was developed. This unit handled the control of the camera and the storing of pictures in RAM without using the MCU. The interface to the external subsystems on the Cubesat was designed around the I^2C-bus standard. A special protocol for errordetection and communication was designed. A hamming code algorithm was designed to protect vulnerable data from being corrupted due to e.g. radiation.
A design draft for a suitable camera lens was made. The OBC and camera design and specifications was completed and a prototype of the OBC was build for testing hardware and software.

1. INTRODUCTION

Over the last decade space has become more and more commercial, and a lot of countries can now call themselves space nations including Denmark. In the year 1999 Denmark launched her first satellite Ørsted, designed to measure the magnetic field around the earth. Ørsted has been a tremendous success for DSRI (Danish Space Research Institute, [DSRI (2001)]) and is still transmitting valuable data to the earth, now more than 1000 days after it was launched. DSRI is currently supporting four Danish satellite projects [Rummet (2001)]:

- Ørsted II, designed to relieve the aging Ørsted satellite.
- Rømer, a new satellite designed to measure oscillation of nearby stars.
- Cubesat, a concept for small satellites specified by Stanford University.

DTU (Danish Technical University) and AAU (AAlborg University) are currently working on two different satellites based on the "Cubesat" concept. The dimension of a Cubesat has to be

$10 \times 10 \times 10$cm with a maximum weight of 1 kg. The Cubesat concept is designed to make space available for "everybody". Until now space has only been accessible to large institutions and companies. With the Cubesat concept small companies or even individuals have gained access to space.
The main mission of the AAU Cubesat, due to launch november 2002, is to take pictures of Denmark and publish them on the internet. It will also be possible for the public to order a picture of a specific area of Denmark. The idea is to increase the interest in space applications amongst the Danish public and industry.
A secondary mission has been implemented on the AAU Cubesat. This mission will be to point the camera towards the stars, in order to measure changes in the brightness of stars and detect new ones.
The AAU Cubesat is designed through a collaboration of student groups from different institutes at AAU, ranging from 5th semester to 9th semester [AAU (2001)].

1.1 The purpose of the project

The purpose of the project was to design an OBC and a camera payload for the AAU cubesat. Designing hardware for a space applicable unit requires a lot of considerations that would not normally be encountered when designing hardware. This is due to the extreme environment for which reason considerations such as temperature ratings, exposure to radiation, heat transfer, robustness of components and redundancy should be taken into account.
The purpose of the OBC was to provide a platform for the Control and Data Handling (CDH), a supervisor program that controls the tasks of the satellite, and to interface the satellite units to CDH. These units were: PSU (Power Supply Unit), ACS (Attitude Control System), COM (Communication unit) and Payload (camera unit). This article describes the main conciderations and solutions chosen during the project.

2. DESIGNING THE OBC AND PAYLOAD

The design was split into three parts: The Hardware and Software Architecture and the camera payload. Even though the parts were connected it was possible to work in groups designing each part seperatly. All three parts will be described in the following.

2.1 Hardware architecture

The AAU CubeSat was made up of several units. Figure 1 shows these and the internal architecture of the On Board Computer (OBC). The OBC hardware consisted of the following units:

- An MCU
- PROM modules
- Flash memory modules
- RAM modules
- Adress decoding logic
- Camera control and DMA logic

These components will be described in the following.
The main MCU was selected to be a C161PI. This is a 8051 based MCU, with a 16 bit Central Processing Unit (CPU). It had a 24 bit address space that made it possible to address up to 16MB of memory. Beside a low power consumption ($\approx$10mW/MHz) it had several usefull features including a Real Time Clock (RTC), Analog/Digital converters (ADC's), timers, flexible power management and several watchdogs preventing that the program stalled permanently.
The memory was designed to consist of 256kb Programmable Read Only Memory (PROM) which contained the basic program. A PROM memory was used, because it is very robust to radiation and in a wide temperature range. The memory also consisted of a 256kb flash memory which were able to hold new updated software. This type of memory is not as reliable as PROM, but the MCU was able to rewrite it. Finally the OBC had 4MB of RAM for storing the image from the camera and for holding the MCU stack.
The control logic created chip select (CS) and read/write (R/W) signals to the memory from either the MCU or the camera. The logic was designed to work in two different modes. The first mode was during normal operation and the second mode was during camera DMA.
During camera DMA, the MCU had to be disconnected from the address and data bus. This was done by software, since the C161PI did not offer an opportunity to share these busses. Software interacting with the camera DMA was therefore set up to run from the MCU's internal RAM.
Robustness was introduced e.g. in following ways: An oscillator watchdog and software watchdogs was activated on the MCU. A Flash Memory was implemented to allow the uploading of new software after launch (e.g. if programerrors were detected). The power to the logic was also connected to the camera, so in case the camera DMA malfunctioned, it could be turned off to prevent it from blocking the databus.

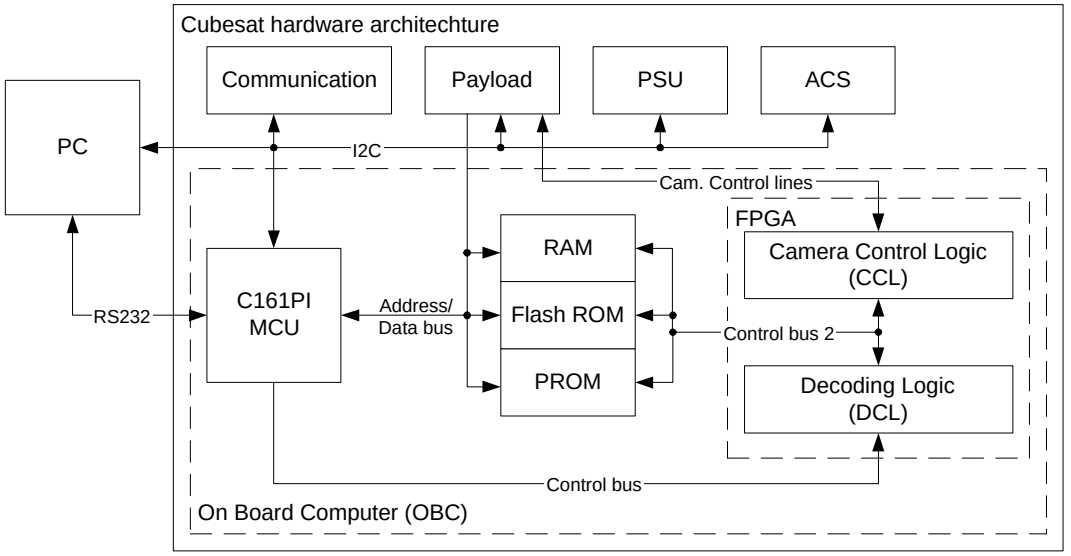

Fig. 1. *Hardware architecture of Cubesat*

2.1.1. Internal interfaces All units inside the OBC were interfaced by four bus systems: An address and a data bus connecting memory to the MCU and two control busses controlling which memory chip has acces to the data bus. One from MCU to the Decoding Logic (DCL) and one from Camera Control Logic (CCL) to memory. See figure 1.

2.1.2. External hardware interfaces The connection between OBC and all the peripherals of the CubeSat was made with an I^2C bus system. For debugging and programming flash memory or PROM, a serial interface was made, so that the OBC could be interfaced to a PC. By utilizing a bootstrap loading mechanism, it was possible to use the already known development software, Keil.
An external control bus to the camera was needed because this device required a Direct Memory Access (DMA) to RAM, due to its high speed data transfer rate. It was therefore also connected directly to the internal data bus of the OBC.

2.1.3. DMA for the camera The camera worked using a rolling shutter, and therefore delivers streaming data on the bus. A clockfrequency at 12,5MHz was used when taking a picture, which ment that new data was availible on the data bus every 80ns. This required the implementation of a DMA module to fetch the data, because the MCU was not fast. The software used to activate the DMA channel, was designed to run from internal RAM on the MCU, because the MCU did not allow hardware bus arbitration.
When the software was activated, it would put the MCU both in idle and high impedance mode. An interrupt signal, created by the DMA when done transfering data, would wake up the MCU from idle mode. In this way the camera DMA was transparent to CDH.
The DMA was made by using a 21 bit counter to hold the RAM address. The first 18 bits were used directly to address RAM cells, and the last 3 bits to create 8 chipselect ($\overline{CS}$) signals. The chip selects made it possible to contact all needed RAM devices. Figure 2 shows the implementation of this. The address counter was reset and enabled by the MCU before activating the DMA. This was done by first pulsing the reset pin, and then pull the Output Enable pin low on the counter.

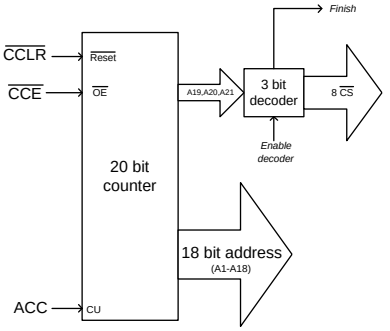

Fig. 2. *DMA address and chipselect creation*

The size of the picture from the camera is 1280x1024x10bit which makes it require 2,5MB of RAM for storage. Since each RAM holds 512kB, the image requires 5 RAM block. The 6th chip select, has therefore been used as an interrupt signal, to awaken the MCU from idle mode.
Since a 40ns (t_{ACC}) delay exists in the counters, the clock pulse must go high 40ns before address change was expected from RAM. To do this a signal called ACC (Address Counter Clock) was generated, by delaying a signal provided from the camera, called HCLK (Horizontal Clock). This camera generates and synchronizes the data

stream. Figure 3 illustrates the logic.

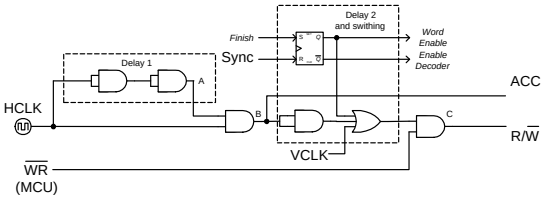

Fig. 3. *Gate logic for creating R/W and ACC signals*

By delaying the ACC further 30ns the HCLK signal was also used for pulsing R/W signals to the RAM (t_{cd}) devices as required when data should be written to the RAM. HCLK has a period of 80ns ($f_{HCLK} = 12,5MHz$), and a duty cycle of 50%, which means that the low period is 40ns. RAM requires a low period of 50ns (t_{WP}), the R/W signal was therefore generated by shortening the HCLK. This was achieved by creating a delayed HCLK and then AND it with a non-delayed HCLK. The required timing is shown in figure 4.

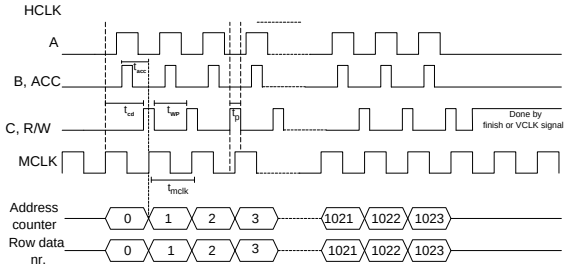

Fig. 4. *Gate logic for creating R/W and ACC signals*

Furtermore the combination of a flip/flop and a OR gate was used to turn off the logic, by holding the input high on the OR gate. This was implemented to ensure that noise on the HCLK input did not trigger a write access. When the rolling shutter of the camera makes a lineshift (vertical shift), the camera holds the signal VCLK (Vertical Clock) high. This also disables the write mechanism on RAM.

2.2 Software architecture

The designed software was made to boot-up and afterwards serve the CDH. Figure 5 shows the software architecture of the OBC.

As seen on the figure, the CDH could execute different functions on the OBC. These functions acted as hardware drivers for the CDH. By using these drivers different tasks were made transparent to CDH. CDH was equiped with a simple softwareinterface to different hardware functions.

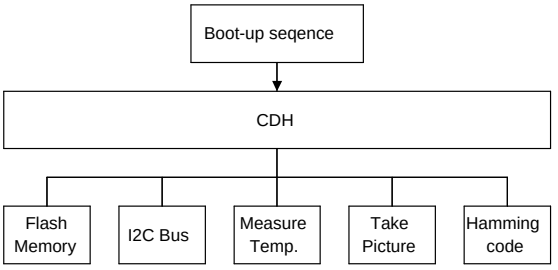

Fig. 5. *Software architecture of CubeSat*

The drivers and the Boot-up sequence is described in the following:

2.2.1. CubeSat I²C-bus The peripheral bus was selected as the I²C-bus standard [I2C (2001)]. Only two bus lines are required: A Serial DataLine (SDA) and a Serial Clock Line (SCL). The two wires, SDA and SCL, carry information between the devices connected to the bus. A master (in this case the MCU) is the device which initiates a data transfer on the bus and generates the clock signals to permit transfers. When this is done the device addressed is considered a slave.

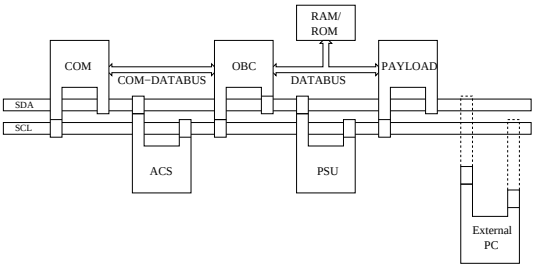

Fig. 6. *I²C-Bus structure*

The Cubesat structure was designed to consist internally of five different modules (COM, OBC, ACS, PSU and Payload) as seen in figure 6. The last module on figure 6 is an optional external PC used for testing and simulation of the system.

Initialization of the I²C-bus: The I²C-bus was configured, monitored and controlled by setting and reading different internal registers: (ICCFG, ICCON, ICST and CRTB) [MCU (2001)].

- **ICCFG**, I²C configuration register. This register enabled the desired I²C bus ports. The MCU had two complete I²C interfaces. The register also set the bit rate for the bus. I²C-Interface 1 on the I²C was used, with a bitrate of 97.6 kb/s.
- **ICCON**, I²C control register. This register controlled in which mode the MCU was set, Master or slave. It also decided whether 7 or 10 bit addresses were used. The MCU was set in master mode, and 7 bit addresses were used.

- **ICST**, I^{2}C status register This register contained the current status of the I^{2}C bus and indicated if an interrupt was pending
- **ICRTB**, I^{2}C transmit and receive buffer The buffer contained the received data or the data about to be send.

I^{2}C Protocol: A protocol was designed on top of the I^{2}C-bus. The complete protocol therefore consisted of the following (ordered as it would be transmitted):
Start, Address, R/W, Acknowledge bit, followed by a header and checksum designed especially for the cubesat, followed by the Data and a Stop bit.

Header: The header as shown on figure 7 contained information about the length of the sent data packages. It also contained a module number used to describe either which module in the slave was contacted or used as a specific command.

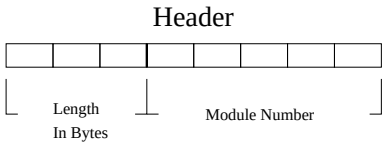

Fig. 7. *I^{2}C-header.*

- **Length:** The first three bits of the header determined how many data packages would follow the header
- **Module:** The last 5 bits indicated a command or the device module number.

Checksum: The data package following the header was an 8 bit CRC Checksum. The checksum was calculated in the following way:

$$[FFh] - \sum [DATA] = [checksum] \quad (1)$$

All the data packages and the header send in one transmission were added together, the 8 bit result of this calculation was then subtracted from [FFh], this yielded the CRC checksum. The data was checked in the following way:

$$[checksum] + \sum [DATA] = [FFh] \quad (2)$$

The sum of received data were added with the checksum. If the result was different from [FFh] the data was corrupted and would be retransmitted (see example in figure: 8).

Conducting housekeeping and controlling units: A function was designed to collect housekeeping. When a collection of housekeeping was initiated the master were only to address a unit and set the R/W condition to "Read". The unit would then deliver all its housekeeping data to the MCU. Each sensor on a unit was identified by the module number transmitted in the header.
The function for collecting housekeeping was

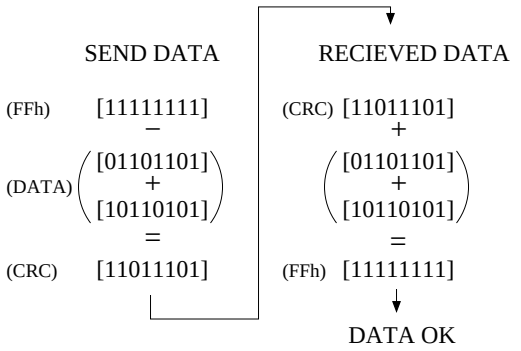

Fig. 8. *Calculation of checksum*

called I2CHousekeeping().
When a unit was controlled from the MCU, the MCU would have to send data along with the address, header and checksum. In order to write data to a unit the I2CWrite() function was used. It was also possible to read data from a specific sensor. In order to do this, the function I2CRead() was used.

2.2.2. The Boot-up sequence One of the most vital functions of the OBC is the Boot-up sequence. The purpose of the Boot-up sequence is to configure the MCU and activate CDH. After the activation of CDH the OBC was designed to become a passive unit taking orders, like the I2CWrite() or I2CHousekeeping() command, from CDH. The boot sequence was to be carried out autonomously, because during boot-up there would be no intelligent control (CDH) and no communication between the earth and the satellite. If the boot sequence failed and the OBC was not prepared to deal with the fault by itself the satellite could be lost. To ensure that software errors could be corrected if detected after launch, it was made possible to upload, boot up on and use new software. The new software would be stored in the on board flash memory. Data in flash memory can be corrupted due to radiation. To ensure that the critical boot software was not corrupted, the internal and very robust PROM module was used. This module would hold the original and well tested software used when the satellite was launched. In this way the PROM provided the OBC with a fail safe mode. If an error occurred during boot-up, using the new software the OBC had been designed to automatically reboot and try to boot up using the original software. This also worked the other way around in case of the system trying to boot-up using the original software. By implementing this into the Boot-up sequence, the OBC could deal with faults in one of the two boot software versions thereby making the Boot-up sequence more reliable.

The OBC Boot-up sequence will be described in the following. The algorithm of the sequence is shown in figure 9. The Boot-up sequence would

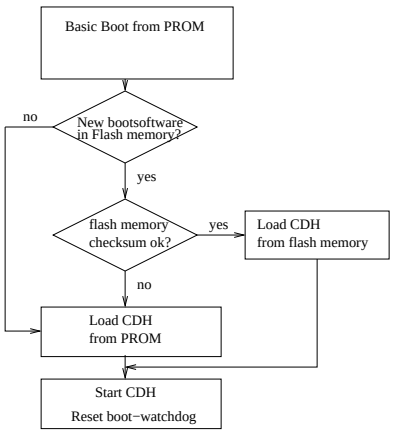

Fig. 9. *OBC Boot-up sequence*

be executed as soon as the MCU was turned on. The MCU was designed to always start by contacting the PROM module. The first part of the module, the basic boot, was made to contain the basic software needed to set up the different parameters internally in the MCU. This part was kept in the PROM because it was the most vital part of the boot software and e.g. indicated what the internal MCU setting requirements should be. By keeping the basic boot sequence at a minimum, it was possible to alter more internal MCU configurations in the boot even after launch. If a port had to be reconfigured later, this could be done by uploading new software as described later on.
When the CDH uploaded new software it would send a command through the I^2C bus to the PSU, telling it that new software was uploaded. PSU would then signal this back by either a high or low signal on a connection directly to the OBC. This connection was to be used during the boot. After the basic boot the MCU would then examine this connection. If the connection did not indicate new software the MCU would continue the boot from PROM. If it indicated new software the MCU would examine if the software stored in the flash memory was valid. This was done by testing the software using a checksum, as described later on. If the software was valid it would continue its boot from the flash memory. If not it would continue the boot using the old software in the PROM module hence working in a fail safe mode.
The PSU was set to control this because it had the possibility of rebooting the OBC without being affected by it.
A special algorithm was designed on the PSU utilizing a watchdog. The algorithm is illustrated in figure 10.

The initial state of the connection (PORT X in figure 10) was low indicating that the MCU should boot from the original software. When the PSU powers up the OBC the first time the OBC would therefore boot from PROM. As the

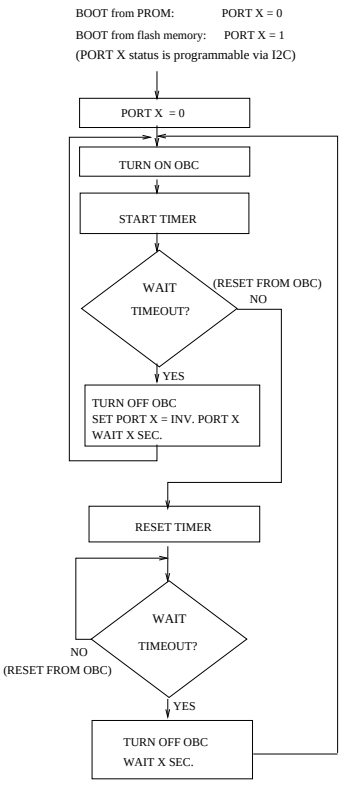

Fig. 10. *PSU Boot-up control*

MCU continued its boot-up the PSU started a timer. When the MCU had finished the boot-up and started CDH the first thing CDH did was to write a RESET command to the PSU. This was done using the I^2C bus. When PSU recieved the command it reset its timer. If the boot-up of the MCU failed and the system stalled this timer would timeout. This indicated to the PSU that the current boot-up had failed. This could be due to the software being corrupted. PSU would then turn off the OBC, invert the status of PORT X, wait a predefined period of time to let the system settle and then turn the OBC back on. This routine would be continued until the OBC boots succesfully. This means PSU would keep trying to reboot the OBC, switching between the PROM and the Flash memory until a boot from one of the two software versions were successful. After the boot the PSU would start a timer. This timer should then be reset in specific timeintervals. If the OBC stalled, this would not be done and the PSU timer would timeout. When this occured the PSU would turn of the OBC and restart it.

2.2.3. Uploading new software to the CubeSat It was designed so that after launch of the satellite, new software could be uploaded. This would be desirable if the software onboard did not work properly. Another reason could be e.g. trying to optimize the performance of the satellite.
Before the new software could be uploaded, the flash memory modules had to be preflashed, which

meant that all the bit cells in the flash memory were set to '1'. The Flash memory modules had a build-in function that flashed the memory, if particular codes were written to the address and data bus. The function returned an error if a bit cell could not be set to '1'. This would make it possible to monitor if the modules malfunctioned during the satellites lifetime. When the modules were flashed, the OBC could start storing new software into the modules as soon as the new software was recived and stored in RAM. When the new software was stored the the OBC could be rebooted and the new software used.

2.2.4. The Hamming code The purpose of implementing error correction code on the Cubesat was to avoid error in the bit information. It was mainly when the bit information was stored in memory, errors were expected. This was due to the high exposure to radiation. The error correcting code was based on the Hamming (12,8)-code algorithm. This algorith was able to find and correct 1 bit in each byte. The Hamming (12,8)-code algorithm encodes the desired byte with a specific code enlarging the size of the information from 8bit to 12bit. When the data was needed or checked the hammingcode would use a specifc algorithm. This algorithm would find and correct errors and return the original byte. On the Cubesat there were two algorithms for handling the Hamming code: One to encode and another to decode and correct the data.

2.2.5. Getting temperatures of the OBC and Camera A function was made to collect readings from the temperaturesensors on top of the OBC and camera. The temperature measurement were to be used as a part of the housekeeping collection. The two sensors were of the type LM19. It had a linear dependence between output voltage and temperature making the readings simpler. An A/D-converter was integrated on the MCU and this was used to convert the analog signal from the censors to an 8bit digital signal. The A/D-converter was set to convert in single mode, meaning that it only converted one signal at a time. The input to the function was which sensor to convert. So if both temperature were to be used, the function Get_temperature(), needed to be called twice.

2.2.6. Taking the picture When taking the picture the command from CDH only copied the needed software into the internal RAM and initiated the hardware logic.

2.3 The payload camera

The camera unit was designed primarily to take pictures of the earth covering an area of roughly 100km by 100km. Secondly it was to be used as an observation platform enabling scientists to measure the changing brightness of stars. The camera unit which was a standard component only slightly modified to meet the requirements of a space mission, was to be supplied by the Danish company Devitech. The biggest problem when interfacing the camera to the OBC was to transfer and store data with a high bit rate from the camera. The high transferrate was needed to prevent dataloss, since the camera had no possibility of buffering the data. DMA was implemented instead of a buffer to save components.
The camera was able to have its internal parameters changed. These parameters could be e.g. the gain of the photo-chip inside the camera (this can be compared to the aperture of an ordinary camera). These parameters and the general control of the camera could be accessed via the I^2C bus. The photo-chip supports the I^2C bus and by using this, it was possible to achive a simple and robust system. Using the I^2C bus would also make it easy to read the commands sent to the camera unit for debugging purposes. Since it was not possible to find a photo chip with industrial specifikations (the working temperature of the camera ranges from 0-40 °C) an alternative solution was chosen. By placing the camera in the centre of the satellite the smallest possible change in temperature should be insured, when the satellite was moving from shade to sunlight. By placing the camera right above the MCU, it would be possible to use the MCU as a heat source. To ensure a good picture quality a simulation of the optical system was made. This simulation showed that a triplet lens would be sufficient. It turned out that the lens had to be made from special radiation hardend glass to be used in space.

3. IMPLEMENTATION OF THE DESIGN

This section describes the measures that had to be taken when the design was to be implemented in the total design of the satellite.

3.1 Hardware implementation

The OBC will be build in two versions; an engineering model and a flight model. The engineering model is used for final testing and the flight model for launch. It was not possible to build these two models due to the complexity of the Printed Circuit Board (PCB), but a prototype was build in order to later test the functionality of

the hardware and debug the software. To ensure a properly manufactored PCB, a PCB producing company will be involved in order to produce the final prints. It was found necessary to use multi layer prints with at least industrial standards, in order to meet the requirements of operational temperatures and physical space available in the cubesat. Also components with SMD packages was chosen to minimize the dimensions of the OBC PCB.
Instead of PEELs and counters, an FPGA was designed instead. The FPGA eX128 from Actel offered the possibility of having both the DCL and CCL implemented in one chip. This would save both space for the components and power. About 100mW could be saved by using the FPGA. The FPGA also offered less internal delay. This again made the DCL more transparent for the MCU. The strict timing requirements for the CCL and DMA could also be met, since the FPGA offered the possibility of using special internal delay functions. An importaint factor was also that the FPGA is only One Time Programmable (OTP). This makes it more robust than PEEL circuits.
For the prototype PCB, flash memory were used instead of PROM. This was done in order to allow the developing and testing of software without having to change PROM each time new software were to be tested.

3.2 Software implementation

Since the OBC hardware was not realised during the project, the software was neither implemented nor tested on a final system. The different software modules were designed though.
The design was documented by illustrating the functionality of the programs with the help of flowcharts. Some of the needed code was added into the description of most modules.
As far as possible software principles behind the design were tested on a MCB C167 board. The test board used a C167 microcontroller which was part of the C161PI microcontroller family.
The code might be modified when it is later implemented on the OBC. But when the OBC hardware has been fabricated, the final code has been programmed and the software is going to be implemented, this can, due to the described design process, hopefully be carried out without too much trouble. The later interfacing of the OBC design to the design of the other satellite subsystems showed that modifications had to be made to parts of the software design:

3.2.1. The I^2C bus The I^2C bus was supported by the MCU to a degree that only small hardware modifications were needed to be taken into account when implementing the I^2C bus. The software control of the I^2C bus from the MCU was done by controlling different internal registers and buffers in the MCU. This control had to be modified from the normal procedure due to two reasons:

- **Interrupt control:** Due to the later implementation of the operating system (CDH) the use of the I^2C buffers had to be changed to use interrupts. The interrupts would ensure that the control and timing of program execution was kept in the hands of CDH.
- **Repeated start:** The units that the OBC would contact over the I^2C bus had microcontrollers implemented that did not support "Repeated start". "Repeated start" is a vital part of the I^2C standard. The I^2C protocol designed for the OBC therefore had to be redesigned to compensate for the lack of this.

3.2.2. The Hamming code algorithm A Hamming code algorithm was made to detect errors in the data stored in the RAM modules. The Hamming algorithm was able to detect 1 error and correct 1 error in a byte. When the sattellite is exposed to radiation, 2 errors can occur simultaneously though. The Hamming code was therefore redesigned later to be able to detect and correct up to 2 errors in a byte [Pretzel (1992)].

3.3 Payload camera

The interface between the OBC and the camera was designed, but not constructed. The software for the interface has been designed but the actual program is neiter written nor tested. The hardware interface between the camera and the OBC was designed and constructed in a provisionally form. It will not be possible to finish it before the OBC has been completed and a prototype of the final camera unit has been received and implemented. A draft for the design of the lens was made and contact with a company from New Zealand, willing to produce the lens, was established. When the lens design is finished and a suitable glass type has been found, the lens can be produced. The camera and the lens must be calibrated together e.g. to ensure that the picture is in focus. Tests of the dataflow timing between the camera and the OBC must also be conducted.

4. CONCLUSION

During the project the On Board Computer (OBC) was designed. Hardware requirements were defined, functions and methods defined and suitable components found. The printed circuit board

(PCB) of the (OBC) was also designed and a prototype was constructed for later testing and debugging. Due to the complexity of manufactoring the final PCB, mounting the components and getting the hardware parts from the producers, a final OBC was not constructed. This also ment that the software could not be tested. Instead all software was designed and described to help later implementation of the software into the OBC hardware. The I^2C bus system was chosen and a special protocol was designed for the bus. Arranging hardware tests, establishing external contacts to companies and individuals, doing initial lens designs and researching possible satellite missions was also done during the project.

REFERENCES

AAU. *http://www.cubesat.auc.dk/.* 2001.

DSRI. *http://www.rummet.dk/.* 2001.

I2C. *http://www.philips.com/.* 2001.

MCU. *http://www.infineon.com/.* 2001.

O. Pretzel. *Error-correcting Codes and Finite Fields.* Oxford University Press Inc., New York, 1992. ISBN 0192690671.

Rummet. *http://www.rummet.dk/.* 2001.