

COMMAND AND DATA HANDLING ON PICO SATELLITE FOLLOWING CUBESAT DESIGN CONVENTIONS WITH A CAMERA MISSION.

**Brian Fisker, Christian Hansen, Christian Vase Petersen,
Jimmy Klitgaard, Søren Hansen and Thomas Rasmussen**

** Department of Electronic Systems,
Aalborg University*

Abstract

Satellite development is becoming faster and less expensive with the introduction of the CubeSat concept, where low budget pico satellites are developed within a year using off-the-shelf components. The rigid requirements associated with CubeSat have so far excluded the satellites from demanding missions. However, if CubeSats carry more complex missions they will also be of commercial interest due to the low cost and quick development.

Command and Data Handling of a CubeSat with a camera mission which involves storage, compression, and transmission of large amounts of data, has been investigated. This research is part of the CubeSat project at Aalborg University and is the basis for the development of a design and an actual implementation of Command and Data Handling for the CubeSat currently under development. In general it is concluded that CubeSats can be designed for camera missions, and thereby making them more interesting for commercial applications, provided that the customer is satisfied with approximately one picture per day.

Keywords: CubeSat, Compression, Command and Data Handling, Camera, Satellite

1. INTRODUCTION

The CubeSat concept was inaugurated by Professor Robert Twiggs of Stanford University primarily for educational purposes, to allow students to experience the entire process of building and flying satellites [7]. A CubeSat is by definition a pico satellite of 10x10x10 cm with a maximum weight of 1 kg. Launch vehicles are being constructed which will carry between 1 and 6 CubeSats, and put them into a Low Earth Orbit at about 600km [4]. Currently several CubeSat projects are ongoing at universities around the world with various missions [8].

According to Prof. Twiggs a CubeSat with a camera mission is yet to be developed due to requirements of handling large amounts of data [1]. Combined with the rigid requirements of a CubeSat, a camera mission producing pictures has seemed beyond the capacity. Driven by the desire to push the limit of the CubeSats

as well as breaking ground for commercial applications, several research groups at Aalborg University (AAU CubeSat) have been working to explore the possibilities of a camera mission.

When building a CubeSat it is important to remember to *Keep It Simple Stupid* (KISS). This is achieved in the hardware by choosing off-the-shelf components, and in the software by choosing solutions that might not be as sophisticated as possible, but rather as simple as possible. In general the space industry does not use standard components. When launching a satellite with e.g. NASA, all critical parts of the satellite must be tripple redundant, otherwise it will not be launched [1]. This adds greatly to the budget and complexity of building the satellite. The CubeSat concept breaks away from the conservative industry by using standard components, which can be bought in any well-equipped electronics hardware store.

Aalborg University started a CubeSat project in September 2001 with the ambition of launching the satellite in November 2002. Several groups have been formed, each being responsible for a subsystem of the satellite. This article is the outcome of one of the research groups focusing on the Command and Data Handling (CDH) in a multi-threaded environment, and the object is to discuss the requirements for fulfilling a camera mission.

1.1 *Setting*

Taking a picture of the Earth from a satellite adds strict constraints to each component used in the satellite. The camera should have a high resolution, have a low-power consumption and be as light as possible. To obtain a good quality picture, the satellite has to be stabilized before taking the picture, and the position of the satellite should be known to get predictive results. Picture data must be stored on the onboard computer which requires large amounts of memory. One of the most fragile components onboard a satellite is memory which is sensitive to radiation. Furthermore, all the data should be transmitted to the ground through a communication channel which also requires power. Due to the small size of the CubeSat, power is a limited resource, thus requirements to the transmission rate are dependent on the amount of power available, and the amount of data to transmit to the ground.

1.2 *Content of article*

As participants of the CubeSat project at Aalborg University we developed the CDH for the satellite and investigated the requirements for a camera mission. The following section describes the methods used in the research and development. In the *Validation and Results* section the tests of the system and the results of the tests are described. The article concludes with a discussion.

2. METHODS

The investigation of requirements of a camera mission was done by developing a prototype of the CDH. The software was then systematically tested in order to provide a detailed conclusion. This section briefly describes the hardware platform used on the AAU CubeSat, the operating system used, calculations concerning picture resolution versus transmission time and the structure of the software for the CDH.

2.1 *Off-The Shelf Products*

The CDH system is developed using the following development and operating system software.

- Keil PK166 Professional Developers Kit V3.12
- Keil RTX166 Real-time Operating system V4.01

The Keil development kit includes a C compiler, a debugger, an emulator, a simulator board, and a programming environment for Windows. Since the hardware for the satellite was not ready before the project had finished, the only way the software could be tested was to use a Keil's simulator board and emulator. The tools were very useful because it is possible to program the application in C and debug it, either on the simulator board or using the emulator.

There were two choices for the Operating System: the Keil OS and the Micri μ m μ C/OS-II. Both were sponsored by the respective companies. The operating systems complied with the basic demands for the system, meaning they have the following facilities:

- Multi-threading
- Semaphores
- Message passing
- Time management

The multitasking of both OSs is preemptive, which is an advantage since one low priority thread can not block all the other threads. However the Keil OS was chosen over the Micri μ m OS for the following reasons:

- More flexible thread handling - It is possible to start a new thread from any thread in the Keil OS, while all threads have to be started from the main thread in the Micri μ m OS.
- Better mailbox functionality - In both OSs, it is possible to wait for data at a mailbox, but in the Micri μ m OS, it is a busy wait. This means that lower priority threads will not get processor time when a higher priority thread is waiting at a mailbox.
- Error handling - In the Keil OS a default error handler will be created for each thread, which will be called in the event of an error in this thread. It is then possible to replace this default error handler with a customized version which can perform corrective action.

2.2 *Overview of CDH*

The physical domain is modeled by the CDH as illustrated in figure 1.

The different subsystems are:

- Command and Data Handling (CDH) - Controls the onboard computer by receiving, processing and distributing data to and from the other subsystems.
- Power Control Unit (PCU) - Interface to the power hardware to monitor power consumption, charging of batteries, remaining power, current and power levels and other housekeeping.
- Batteries (Batt)

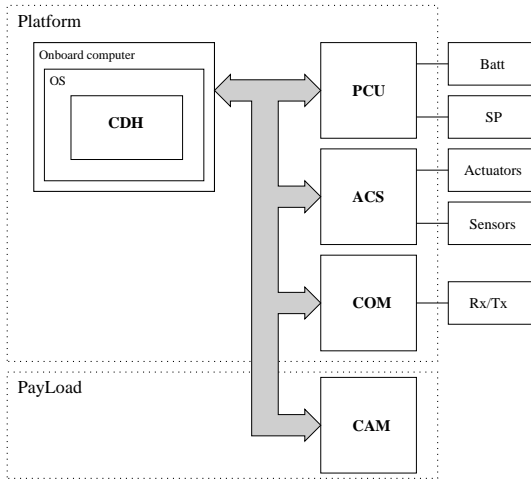


Fig. 1. Functional diagram of the subsystems in the problem domain

- Solar Panels (SP)
- Attitude Control System (ACS) - Controls the orientation of the satellite using attitude actuators.
- Actuators
- Sensors
- Communications subsystem (COM) - Communication to ground station.
- Receiver/Transmitter (Rx/Tx)
- PayLoad (CAM) - Camera.

The subsystems are connected to a shared I2C bus through which communication is established. The external units (Batt, SP, Actuators and Sensors) are all controlled by the individual subsystems, i.e. the PCU controls the batteries and the solar panels.

The CDH is further divided into a supervisor (SPV), a log module (LOG) and a flight plan module (FLP). Each of the modules have its own field of responsibility.

2.3 Software Structure

Based on the findings in the problem domain, a complete structure of the software system was identified. Since the AAU CubeSat project was spread out among several groups, resulting in a lot of different people writing software for the CDH, a highly modular design was the goal. When a lot of people are developing software for the same project, it very important to have the interfaces defined clearly, so the different parts of the software will be able to communicate in a predefined way.

As previously described, the Keil RTX166 OS was used, which supports multi-threading, because it enables the option of having each independent part of the system represented by one or more threads, thus fulfilling the modularity requirement. The communication between the different threads was done using message passing implemented via message queues.

Figure 2 shows the structure of the CDH, where each of the boxes depict an independent module in the system.

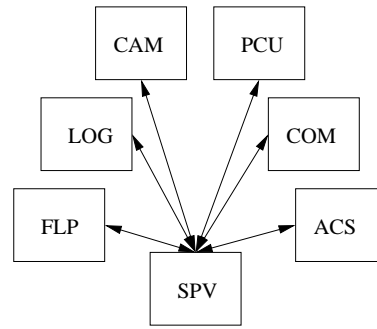


Fig. 2. Thread structure for CDH

From this figure it is also evident that the SPV module plays a central role in the data distribution. For simplicity, it was chosen that all the routing of data in the system, was to be handled by the SPV. One of the benefits of this decision is that the communication paths are simplified. The developer of a particular module only needs to agree with the developers of the SPV about what kind of messages can be sent between them. It also means that the SPV is able to monitor all the communication in the system, and thus is it able to pick out vital data and send it to the LOG module.

Furthermore, in order to simplify the coordination process between the different groups of developers, a standardized format for the messages being passed between the different modules is used.

The CDH uses the message format shown below:

```

struct Message_Agent{
    unsigned char sender ;
    unsigned char command_field ;
    void * data_block ;
    unsigned long data_block_size ;
};

```

When sending a message, the sender identifies itself, by setting the `sender` field, indicates what the message means in the `command_field`, and optionally indicates where the receiver can find data associated with the message by pointing the `data_block` to the memory location of interest.

2.4 Functionality

The main functionality of the CDH is based on a flight plan which is a time scheduler that specifies the execution of tasks in the satellite. Tasks are inserted into a flight plan on the ground and the plan is transmitted to the satellite. The COM module receives all data from the ground and forwards it to the SPV (as indicated

in Figure 2. The SPV sends the flight plan to the FLP module from which the tasks are scheduled for execution.

Four different tasks can be inserted into the flight plan:

- *GetStatus* which collects status from all the sub-systems
- *TakePicture* which informs the CAM to take a picture
- *SwitchACSMMode*, which tells the ACS subsystem to change into another mode
- *Reset*, which resets the OBC

Each task can be either periodic or non periodic. Figure 3 is an example of a flight plan, and it shows how each task in the plan is linked to the next task. Figures a, b and c show the flight plan sampled at three different times to illustrate that periodic tasks are reinserted into the flight plan after execution.

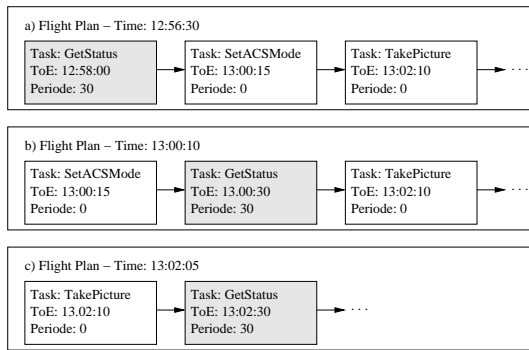


Fig. 3. Illustration of a flight plan sampled at three different times. ToE is an abbreviation for Time-Of-Execution

2.5 Picture

The satellite carries a Kodak 1.3 megapixels camera as a payload. The camera stores picture data in a 10bit format which can be interpolated into a 24bit color scheme on the ground. In AAU CubeSat the OBC uses 16bit memory which means that 10bit pixels are stored in 16bit words. The camera can take pictures in different resolutions which obviously influences the size of the picture data. Table 1 shows sizes of picture data versus resolution of picture.

Picture resolution	Size - 10bit	Size - 16bit
800 x 600	0.57MB	0.92MB
1024 x 768	0.94MB	1.50MB
1280 x 1024	1.56MB	2.50MB

Table 1. Picture resolution and the 10bit and 16bit sizes

In AAU CubeSat the 1280 x 1024 picture resolution is used even though it requires more memory. The reason for this is our interest in obtaining a high quality picture with the additional requirement of obtaining it within a day. The transmission time for a 2.50MB picture is therefore investigated, and found in table 2.

Baud Rate	Transmission time
2400	2 hrs 25 min 38 sec
9600	36 min 25 sec
19200	18 min 12 sec

Table 2. Transmission time relative to baud rate

Noting that the transmission time must be compared to the fact that the satellite will only be in range for communication for a maximum of 12 minutes each orbit, and with an orbit time of approximately 98 minutes, a compression algorithm would decrease transmission time. In order to decrease transmission time and to save power, a compression algorithm was used.

2.6 Compression

The advantage of compressing the picture is that the size is reduced, causing the transmission time of the picture to be reduced. The disadvantage is that bit flips in the compressed data will cause considerable damage to all the data.

JPEG

A logical choice of a compression algorithm is the JPEG algorithm, which is a lossy compression algorithm specifically for pictures [9]. It is known to reduce the size of the picture by up to 10% of the original data without any visible image degradation[5].

One major drawback in using the JPEG algorithm is that it yields the best results when the picture is in RGB format before encoding. The picture made by the camera is stored in a Bayer color filter array and to convert the picture to a format which can be used in the JPEG algorithm an interpolation of the picture data must be performed[6].

This conversion requires memory to store both the input picture and the output of the interpolation. Going from 16bit to 24bit enlarges the output picture by 1.5 times the original picture. At a 1280 x 1024 resolution, the output picture itself will require 3.8 MB. With only 4 MB RAM on board, this must be considered impossible.

It could be considered to do the interpolation in small parts of the picture and overwrite the original data with compressed data to save memory; however, this adds to the complexity of the task. Furthermore the interpolation and the JPEG algorithm itself which uses both discrete cosine transform, quantization and Huffman encoding is a heavy task for the CPU.

The JPEG algorithm is obviously not ideal for a CubeSat, and therefore another compression algorithm which will work under the limitations have been found.

Huffman's algorithm

The compression algorithm chosen is the Huffman encoding algorithm which is a minimal variable-length

encoding based on the frequency of each character in the data [3]. The idea is, that the characters which are used often in the data will get a shorter encoding than the characters which are rarely used. This algorithm is known to compress data to between 20% and 90% of the original size, depending on the characteristics of the file being compressed.

The basic elements of the algorithm are summarized in three steps:

- (1) Count the frequencies of each character in the data
- (2) Find the binary tree and thereby the encoding
- (3) Convert the picture using the encoding

In our CubeSat each character has a 10 bit value, but uses 16 bit. This means that there can be a maximum of 2^{10} different characters. The frequency of each 10 bit character is therefore used to build a binary tree using Huffman's algorithm.

From this binary tree it is possible to find the encoding for each character. Those characters often used will get a shorter encoding than those characters seldomly used. Thus the picture can be converted using the new encoding.

Resource consumption

Basically the data in each pixel of the picture only has to be accessed two times – when the frequencies are found and when the conversion is done.

The calculation of the binary tree does not take much memory as it only works on the frequencies of the 10bit characters. In our case, there will only be a possibility of a maximum of 2^{10} different characters from which the tree should be made. This means that the maximal amount of allocated memory M_{tree} in the tree-building phase will be:

$$M_{tree} = 2 \cdot 2^{10} - 1 = 2047 \text{ nodes} \quad (1)$$

Depending on the implementation each node uses about 26 bytes of memory. According to equation (1) the tree-structure will therefore use a maximum of 52kB of memory.

2.7 Memory management

Memory is a scarce resource on the satellite. Figure 4 shows how the memory is divided in the OBC, and in this section numbers in parenthesis refer to the figure.

Picture data

There is a total amount of 4MB available, and 2,5MB is served for the raw image data (1). Adhering to the KISS principle, it was decided that the 2,5MB used for the raw image should be excluded for other storage purposes. This leaves 1,5MB of free space for other storage purposes, such as uploading new software

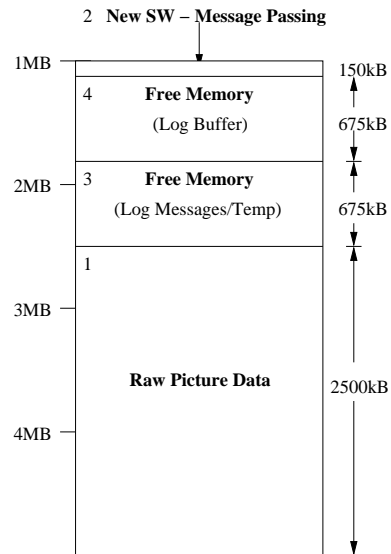


Fig. 4. User memory map of the OBC

and temporary storage used when compressing the image, but primarily it is used for housekeeping log information(3).

Message passing and new software

Passing messages around in the system is a dynamic process where each message is allocated from the available memory space. It is therefore vital that memory is reserved for communication as well. Furthermore if something should go wrong, there has to be enough space to upload new software. The requirements for the size of the software is approximately 150kB, which therefore means, that at anytime there must be at least 150kB of memory free (2).

Housekeeping data

When the logging function is flushed, which is done just before transmitting data to the ground, it allocates a continuous memory block with the same size as that of the log (4), and transfers the complete log data to this buffer (KISS principle in action). In essence this means that when there is less than 600kB of free memory, it is no longer possible to store any new housekeeping data. Optimally the oldest log messages should be purged whenever a new log arrives.

There is obviously a limit to the amount of log data that can be stored in the memory without losing data. Each housekeeping log takes around 100bytes of memory, and with an interval of 30 seconds between each housekeeping log entry, it is possible to orbit:

$\frac{675 \text{ kB}}{0.1 \text{ kB}} \cdot \frac{1}{2} = 3375 \text{ min}$, or around 56 hours, without having to either purge or stop recording log messages. In normal operation, there is a maximum of 10 hours between transmissions, which means that there is sufficient memory to hold log messages even with this very simple memory management [2].

Finally there is an area of the memory used for the running stack, switching threads and compiler variables. This memory is not in figure 4 as it is considered a

minor part of the memory, compared to the relatively large amount of free memory available.

Fragmentation

Memory fragmentation will be handled in the most brutal way (KISS) with a reset of the CDH.

3. VALIDATION AND RESULTS

A prototype of the designed system has been implemented with the crucial functionalities of *execution of flight plan, data and event logging and compression of picture data*. The system has been tested using the Keil development software environment, since the actual hardware was not build, only designed, during the test phase.

The hardware modules from figure 1 are all simulated in their respective software modules, i.e. the COM module simulates input from ground.

3.1 *Verification of specifications*

To verify the single modules and the entire system the black box test method was used. The system was given a flight plan and the behavior of the CDH was monitored with the Keil tools. The output from the system, housekeeping data and picture data, was also examined and evaluated.

Each of the modules were also tested with the black box method to see if a given input produced the expected output.

3.2 *Results*

The implemented parts of the system were tested according to the verification of specifications from section 3.1 and the following results were achieved.

Picture and compression

The handling of 2.5MB picture data was one of the major issues when the CubeSat camera mission was introduced. The allocation of 65% of the entire memory for picture data, the desire to compress picture data, and the large amount of data that needed to be transmitted to the ground were some of the main problems that needed to be solved. The compression of the picture was found to be power saving compared to transmitting the uncompressed picture. The Huffman encoding was chosen because of the simplicity of the algorithm and the fact that it promised no data loss. The compression rate clearly depends on the picture but tests have shown an approximate compression rate of 50%.

The compression time was approximately 35 minutes, which is acceptable with an orbit time of approximately 98 minutes.

CDH implementation

The implemented parts of CDH have been tested with the following results:

- *Receive and execute a flight plan*
The FLP is capable of receiving a flight plan from the COM module and executing it. Periodic tasks are sorted and inserted into the flight plan again at the specified time.
- *Supervisor*
The supervisor is capable of distributing data in the satellite and executing tasks from the flight plan.
- *Get status*
It is possible to retrieve status data from the different parts of the satellite.
- *Log housekeeping and messages*
The collected status and messages are saved in the LOG module for further transmission to the ground. When communication with ground is established the LOG module supports methods for returning the log to the SPV which sends it to the ground through the COM module.
- *Simple power control*
A simple power control has been implemented. Before a task is executed, the power level is checked. The level is compared to a defined minimum level and if the actual power level is too low the task is not executed.

4. DISCUSSION

In this article the possibility of building a CubeSat with a camera mission has been investigated. The main problems associated with having a camera mission on a CubeSat is the storage and transmission of data. With the chosen compression algorithm and the transmission rate it is possible to obtain one image per day.

4.1 *Compression*

The compression algorithm is time consuming, as expected. Tests have shown that compressing the picture takes about 35 minutes. With an orbit time of 98 minutes, and at best a transmission time of 12 minutes, this is not a problem. Actually, it is possible to utilize the time when the satellite is not within communication range for compressing the picture. Another issue with the camera mission is the storing of actual picture in memory onboard the satellite. The raw picture uses approximately 2.5MB and when compressed with the Huffman algorithm it uses approximately 1.5MB, thereby eliminating the possibility of storing more than one picture at a time.

4.2 Keep It Simple Stupid

The KISS principle implies that the most effective methods are not always used, but throughout this project simplicity and modularity have been prioritized. This is mostly expressed in the CAM and LOG module. In the CAM module compression is done using Huffman encoding. This is not the most efficient algorithm, however, it was simple to apply to our system considering the data format given from the camera. JPEG would have been resource consuming because it would have been necessary to convert the picture before compression.

When flushing the LOG module a continuous space in memory is allocated to contain all the data. When allocating this memory, twice the amount of LOG data is required. The advantage of this approach is that the modularity of the structure has been maintained. Thus when transmitting the LOG data to the ground station, the COM module is not required to know anything about the data being transmitted.

4.3 Conclusion

The object of this article was to investigate the use of a CubeSat for a camera mission. We have revealed the possibilities of such a mission through a description of a prototype developed for the AAU CubeSat project. The main goal for the prototype was to design and implement the Command and Data Handling system for the CubeSat. This was accomplished by designing a modular structure for the system, allowing each subsystem to be designed and implemented separately. The following components have been implemented and tested:

- Supervisor
- Flight plan
- Log
- Camera

while the remaining components COM, ACS and PCU have only been implemented for testing purposes.

Even though there are limitations associated with a camera mission on a CubeSat, it is apparent that CubeSats can also be designed for commercial applications, provided that the customer is satisfied with approximately one picture per day.

5. REFERENCES

- [1] Group 720. Interview with Prof. Robert J. Twigg, Stanford University. Interview, October 2001.
- [2] Group 930 and Group 931. Attitude control system. Technical report, Aalborg University, 2001.
- [3] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*, chapter 17.3, pages 337–344. The MIT Electrical

Engineering and Computer Science. McGraw-Hill Book Company, 1990.

- [4] Hank Heidt, Jordi Puig-Suari, Augustus S. Moore, Shinichi Nakasuka, and Robert J. Twigg. CubeSat: A new Generation of Picosatellite for Education and Industry Low-Cost Space Experimentation. In *14th Annual/USU Conference on Small Satellites*, 2001. http://www.cubesat.auc.dk/documents/CubeSat_Paper.pdf.
- [5] Andrew B. King. Optimizing web graphics: Compression - webreference.com. WWW, Februar 2000. <http://www.webreference.com/dev/graphics/compress.html>.
- [6] D. Darian Muresan, Steve Luke, and Thomas W. Parks. Reconstruction of color images from ccd arrays. In *TI DSP Fest, Houston T*, August 2000.
- [7] Cuesta Collega Amateur Radio Organization. <http://www.cubesat.com>. Internet, 2000.
- [8] Elmo Schreder. <http://www.cubesat.auc.dk>. Internet, December 2001.
- [9] Gregory K. Wallace. The Jpeg still picture compression standard. *IEEE Transactions on Consumer Electronics*, December 1991.