LULEÅ UNIVERSITY OF TECHNOLOGY

J. Eickhoff, E. Chester / 2025

# R7023E On-board data handling and Satellite Operations

## Assignment 2: OBDH Model Software Development and Operations

(GROUP ASSIGNMENT)

### Overall Task

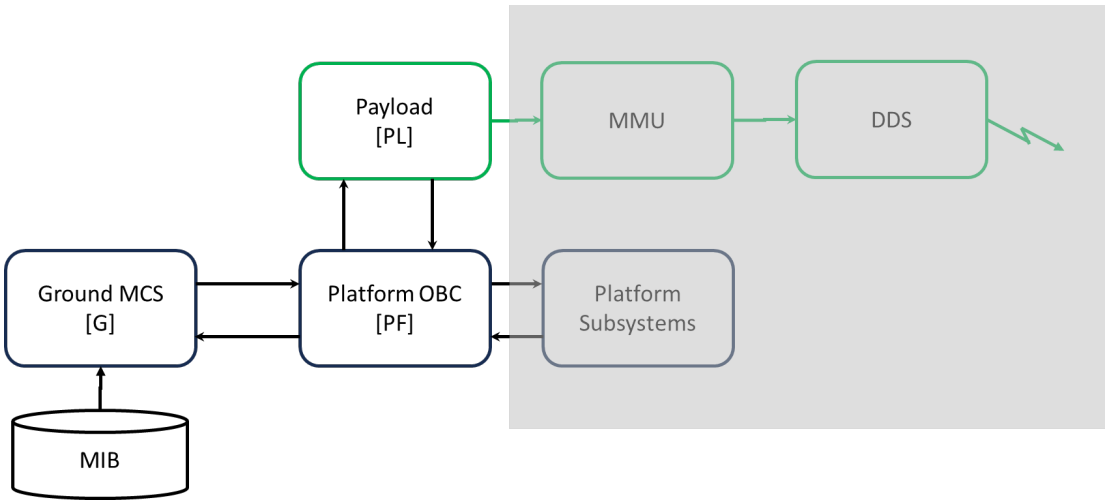
- This assignment is to be completed in small teams of approximately 5 students.
- The objective is to develop (i.e. program) a functional prototype of a spacecraft data handling and operations/control scenario.

There shall be three (3) elements – i.e. three interacting software (SW) programs. Please also refer to the figure below:

- a 'ground' process (Mission Control System – MCS) to send commands and to receive housekeeping Telemetry data
- a 'spacecraft OBDH process (Platform Onboard Computer – OBC) that receives commands from ground and provides interfaces to a mimicked 'payload' (payload instrument control unit (ICU).
- a 'payload' ICU process that executes payload commands provided by the platform OBC when requested and responds with telemetry on execution.

To simplify the detailed description below, these blocks are labelled Gnd, PF and PL. The payload science data chain including Payload science data store (Mass Memory Unit – MMU) and Payload Data Downlink System (DDS) and a Payload Ground System (PGS) are neglected for this assignment to avoid over-complexification. The same applies for Platform subsystems and their units.

The top-level arrangement is shown in this diagram:




Any Ground segment is based on a database (Mission Information Base – MIB) that holds all the Telecommand (TC), Telemetry (TM) definitions and their arguments – e.g. payload photo take time. For the assignment the MIB is considered to be realized as a simple text file.

After development and testing, you will demonstrate the working scenario to complete the assignment along with a written report.

## Detailed Description

### Implementation / Deployment

There is no limitation or requirement about the development method, language, environment. It is expected that most projects will be completed in C/C++ or in Python – but anything else is also accepted.

In the simplest approach (which is considered fully valid for the assignment, although simple), the entire implementation is modelled on a laptop/PC as separated interacting programs – Ground Segment versus Space Segment.

In a more sophisticated approach, the Ground Segment again is suggested to be realized on Laptop/PC and the Space Segment on a dedicated microcomputer (Arduino or Raspberry PI or similar). The connection of both is not prescribed, but a recommendation would be to connect the RasPi to the Laptop via USB and run the Raspi in Gadget Mode. In this 2-component approach with Ground Segment on Laptop/PC and Space Segment on the microcomputer both the Satellite OBC and the Camera are "hosted" on the microcomputer. For the Space Segment Arduino/Raspi it is recommended to use the standard Linux as OS to not over-complexify the task, but for interested students also a solution running an RTOS on the target HW (e.g. free-RTOS) is a valid solution.
*Please note that only Linux offers a remote commanding via USB as Gadget Mode. For other OSes a solution has to be found to couple Space Segment microcomputer and Ground Segment PC.*

### Ground Model

This is the only part of the scenario that will have user interaction. You are NOT required to make any special displays, analysis, graphs etc. The minimum interface is expected: a plain text prompt that can send text telecommands and the same or optionally a second prompt window showing text telemetry replies "from space". As an example, a command could take the form like

```
Set-payload power 1
```

"for the payload, set the power parameter to 1" – meaning, switch the payload on. Something like a complex hex value is not necessary. No PUS or CSP packet definitions are required.

Received telemetry data shall be displayed in a plain text terminal as it is received – e.g.:

```
payload status POWERED
```

Housekeeping data shall be appended with a 'ground reception' timestamp and after display shall be stored in a Logfile.

### Spacecraft OBDH Model – Platform OBC

The PF OBC shall reflect operational modes to a minimum. The satellite (i.e. PF) can be in a SAFE-Mode, where it would reject any payload commands and payload must be OFF. Then for the assignment we consider at least one other mode, e.g. TARGET-Pointing. Other examples are NADIR-Pointing, INERTIAL-Pointing.

Not before the platform is in such a non-SAFE mode, the payload shall be activatable and commandable. Any switch back to SAFE shall imply an automated payload shutdown. Any mode transitions of the PF shall take some time. Mode acquisition shall be confirmed to Gnd by an event TM.

Invalid / mistyped commands to the Ground System should be rejected (with a reply) and not just be ignored. Accepted commands should be acknowledged back to Gnd.

The OBC (PF) shall support time-tagged telecommands – e.g. for “take an Image at time hh.mm.ss” which is in the future. An uplinked TC with an already overdue time tag shall be rejected.

There is no need for implementation of a Gnd timeline. It is sufficient to type TCs and their time tags as arguments into the Gnd console. But for their proper execution the on-board PF and the Ground need to work on the same time base. This must be assured at “boot” of platform and can be achieved by PF and Gnd just using the implementation computer’s local time (we do not need UTC). If the Space Segment is realized by a microcomputer with RTOS, the OBT must be settable after boot.

Time-tagged onboard commands must be executed when its time tag is due.

The payload should be assumed to have at least modes of being ‘OFF’ and ‘ON’, and so proper operations would require commanding the payload ON, commanding it to “take image”, and commanding it OFF again. *There is no physical implementation* for ON/OFF: this is a parameter you must *simulate* in the OBDH process. The payload should confirm “image taking”, i.e. “generate data” back to the PF, both the start as also an assumed finish of the data generation (a good assumption is 5 seconds), and the PF shall forward this payload information down to Gnd in PL housekeeping TM. The platform PF shall also confirm any PL ON/OFF command execution down to Gnd in event TM.

**Standards and Commands**

There is NO requirement to follow the details of the standards for commands or for telemetry. You are NOT asked to implement PUS Packet formats, CCSDS framing, nor PUS packet services. However, you are expected to adopt some of the concepts that the standards provide, specifically:

- Identification of command types (target: payload or platform)
- Time-tags (the capability to schedule a command for execution in the future)
- Sequence count (verifying all expected commands and data are transferred)
- Cyclic housekeeping (HK) telemetry versus Event telemetry (as result of switches on board)

**Telemetry and Onboard Time**

The spacecraft Platform OBC (PF) shall generate some simulated housekeeping telemetry. It shall also maintain an onboard clock to model SCET (Spacecraft Elapsed Time). As long as the OBT has not yet been set, the time shall be reported as simply upcounting SCET value in hh.mm.ss. Before operating the PL and in particular submitting time tagged TCs, the onboard time of the PF must be set from SCET to proper OBT. In a real SC this would be UTC or GPS time.

For the assignment easiest is to use the wall clock time of the computer used for the implementation. The time in TM packets and the time in time-tagged TCs shall have the format hh:mm:ss. For setting the OBT the PF shall accept a set-OBT TC.

The onboard clock time shall be reported periodically in housekeeping telemetry, and shall be used as the reference to set time-tags for future commands and for Event TM that report mode changes of PF or PL on board together with the time it happened.

**Payload Data**

The payload shall generate data on demand – i.e. when having received the PL TC “take image”. Since we are not covering any MMU or science data downlink, the “data take” can be minimalistic by just making the PL write a log entry with the take time into a text file. This is sufficient to prove a “take” and to correlate it with the requested time tags (hint: the OBT is set by TC as earlier mentioned, but how does your PL know about the actual OBT for documenting it in the “image take” log?).

*Note: the real PL data handling in most cases does*

- *not go through the PF SW !! but directly into the MMU,*
- *not go down through the PF S-Band link to the Gnd, but rather goes via separate downlink channels X-Band or Laser-link and in most cases to a dedicated Payload Gnd station which is not the one through which the PF is commanded.*

**Version Control**

You are expected to use some formal method of managing the software development process and tracking changes and ensuring consistency. Use of tools such as git, subversion, etc. is recommended. The method used shall be briefly described in the project report.

**Process**

The first step is to produce for both the Ground Segment software as well as for the Space Segment a simple **Hierarchical SW Function Tree (FT) in Excel**. This includes generation of mode transition diagrams for both Ground and Space Segment Elements.

- Mind that every function defined in the FT will later have a function representation in Code. 1 Function definition -> 1 Code Item -> 1 Test.
- Please mind that also platform and payload modes and transitions must be represented in Functions and later in Code.
- Also consider “requirements” on how you achieve your SW is cyclically looping and e.g. the PF is cyclically looking for uplinked TCs from Gnd or treats mode commanding/changing of the PL.
- Hold a brief software requirements review in your team to agree that you understand the needed functions / you are sure about your function decomposition, and that nothing is missing.

Prepare a short **Software Design Document (SDD)** that describes how you will implement and deploy the solution.

The function tree from above will directly lead you to the necessary TC and TM definitions you will have to hold in your Gnd MIB file together with any arguments.

The mode transition diagrams from the Function-Tree/SRD will directly lead you to the checks to be performed for TC accept (e.g. mode dependency), TC execution, execution duration etc.

It should contain a description of the functional blocks you will program, and at least one (1) UML diagram defining the behavior of at least one (1) block. An important element of the SDD will be how each block communicates. The SDD should also have a list of requirements and which blocks meet/solve those requirements (i.e. a simple traceability matrix).

Develop your programs: note that all three main blocks Gnd, PF, PL are expected to be separate developments. Start with basic intercommunication of the software items, their cyclic looping / waiting (for next TC), and the transfer of test strings (later TCs/TMs). When this works, adapt the contents to meet your requirements.

If you wish to extend the scope to add more realism or functionality beyond the requirements, this is acceptable but must be documented e.g.

- running everything in UTC,
- letting the PL store some real live image – e.g. the actual IRF Allsky Camera image

**Code the Gnd, PF and PL processes** and demonstrate/verify their proper interaction.

**Expected Assignment Outputs**

There are five (5) outputs from this assignment:

1) A Function-Tree Table for Ground- and for Space Segment
2) A brief Code Documentation for the Space Segment (OBSW SDD)
   *Including a simple traceability matrix back to the Function Tree and to the mode Transition Diagrams*
3) The produced code (either in a .zip file, or available via a git repository)
4) A demonstration that the software runs and meets the requirements.
   *The arrangements for this will be confirmed on-site – a good baseline is to record a simple screen video (there are free tools available such as Captura for Windows).*
5) A short report including:
   - your team approach to version control;
   - a verification matrix of the requirements that have been tested, and which are successful;
   - identification of any requirements not tested or not met, and a brief note about why you think this is;
   - any 'lessons learned' about the process and e.g. how you could make your prototypes even better, or what you might do differently next time.

**Deadline**

The deadline for submission shall be **24 / October**.

Outputs (1) and (2) *may* be delivered at any time, i.e. they can be submitted earlier *if you wish*. Early submission of (1) and (2) will give your team the opportunity for feedback from the 'mission customer'. Otherwise, everything must be delivered at the end.

**Assessment**

This assignment is intended to demonstrate understanding of the *principles* of OBDH and the formality of managing functional breakdown, code implementation, documentation, and testing. It is NOT a test of *details* of OBDH implementation: you are not required to follow any standards. You are expected to be *aware* of the standards and include some aspects in your assignment work that demonstrate this awareness.

**Literature (non-exhaustive):**


1) For loop processes implemented in Python: https://codeinstitute.net/se/blog/how-to-wait-in-python/
2) http://pymotw.com/2/socket/tcp.html
3) https://zeromq.org
4) https://www.datacamp.com/tutorial/a-complete-guide-to-socket-programming-in-python