OBC Meeting Outcomes, 21/08/2019

AcubeSAT-OBC-MI-005

Konstantinos Kanavouras

July 20, 2020
Version: 1.2

Aristotle University of Thessaloniki

Aristotle Space and Aeronautics Team
CubeSat Project

2020

Contents

Changelog

Date	Version	Document Status	Comments
20/07/2020	`1.2`	INTERNALLY RELEASED	Fix a small typo on APID length
23/08/2019	`1.1`	INTERNALLY RELEASED	Add explanation for the *response band* TC field
23/08/2019	`1.0`	INTERNALLY RELEASED	Initial revision

This is the latest version of this document (`1.2`) as of July 20, 2020. Newer versions might be available at `https://helit.org/mm/docList/AcubeSAT-OBC-MI-005`.

1 Meeting Outcomes

1.1 CCSDS and ECSS headers

packet primary header						
packet version number	packet ID			packet sequence control		packet data length
	packet type	secondary header flag	application process ID	sequence flags	packet sequence count	
3 bits	1 bit	1 bit	~~11~~ 7 bits	2 bits	~~14~~ 16 bits	16 bits
1 octet				2 octets		2 octets

Table I: Modifications to the CCSDS Space Packet primary header

TM packet PUS version number	spacecraft time reference status	message type ID		message type counter	destination ID	time
		service type ID	message subtype ID			
enumerated (4 bits)	enumerated (4 bits)	enumerated (8 bits)	enumerated (8 bits)	unsigned integer (~~16~~ 8 bits)	enumerated (16 bits)	absolute time
		2 octets		1 octet		4 octets

Table II: Modifications to the ECSS TM secondary packet header

TC packet PUS version number	response band	acknowledgement flags	message type ID		source ID
			service type ID	message subtype ID	
enumerated (4 bits)	enumerated (1 bit)	enumerated (4 bits)	enumerated (8 bits)	enumerated (8 bits)	enumerated (16 bits)
1 octet			2 octets		

Table III: Modifications to the ECSS TC secondary packet header

- A discussion was made to reduce unused fields from the headers of ECSS-E-ST-70-41C packets.
- The results of the discussion are shown on Tables I, II and III
- The final sizes of the headers are as follows:
  - **CCSDS header**: 5 bytes (*-1 byte*)
  - **ECSS TM header**: 7 bytes (*-4 bytes*)
  - **ECSS TC header**: 3 bytes (*-2 bytes*)
- This decision automatically prevents full compliance with ECSS-E-ST-70-41C. This was seen as a minor issue, as compliance is already partially broken on our implementation.
- The resulting gain of *-5 bytes* on TM headers might seem negligible, but can be substantial on small TM messages, such as housekeeping beacons.
- Fields with constant values (such as *version numbers* and *flags*) were removed
- The **message type counter** was reduced to 1 byte (from *2* bytes), as its purpose can be served by the packet sequence count. We are also considering removing it entirely.
  - The advantage of counting messages by their types in addition to the packet sequence count, is that the ground station operator might know which type of message was not sent.
- There was a discussion to reduce service type and message subtype IDs to 4 bits each, but that would deviate too much from the standard, would require a full reassignment of all IDs, and would prevent us from using more than 16 services or 16 messages for each service.
- The **time** format of the packet was decided:

C1	C2	C3	C4
1 octet	1 octet	1 octet	1 octet

***Table IV**: Current time format of AcubeSAT, where* $t = C1 \cdot 256^3 + C2 \cdot 256^2 + C3 \cdot 256 + C4$.

- Given the requirements from other subsystems, a resolution of **100 ms** is requested.
- A modification of the **CUC** time format was chosen to be used for our timestamp.
  - A custom **epoch** (such as 2019) can be used
  - We can store the date using **UTC** (provided by the GPS) instead of TAI. This is *not compliant with the CCSDS 301.0-B-4 standard*.
  - No range constraints for the *C1* field.
  - The time base will be **100 ms**, and time will be measured in increments of 100 ms. For 4 octets, that resolves to:

    $$0.1 \cdot (256^4 - 1) = 13.61 \text{ years}$$

    which is more than double the estimated mission development and operation time, an acceptable value.
- As such, the **absolute time** on our implementation will be **4 bytes** long.

- If the number of ground stations with TX capability is > 1 AND the spacecraft requires the information of the ground station transmitting the command, then the *source ID* field can be used.
- The new **response band** field informs the receiver about **whether to send the TM response to this TC through the 400 MHz or the 2.4 GHz** band.

1.2 Redefinition of OBC role to boost efficiency and reliability

- The current design anticipates all TM packets to pass through the OBC, as the OBC will contain information related to the health of each subsytem, and will be able to perform the coordination of all the information.
- Received packets will be sent as **CCSDS packets** from COMMS to OBC via the CAN bus.
- The kinds of information sent via CAN (i.e. **all information moving between microcontrollers on AcubeSAT**) can be categorized into a few message types:
  - Heartbeat/ping messages
  - Space Packets
  - Parameter gets/sets (*related to ST[20]*)
  - Event occurences (*related to ST[05]*)
  - Functions/commands (*related to ST[08]*)
  - Error/success messages (*related to ST[01]*)

1.3 NAND Flash on SU

- A proposal was made by Stavros Filosidis & Dimitris Stoupis to **keep the NAND Flash memories that store all the SU photographs within the SU vessel**, instead

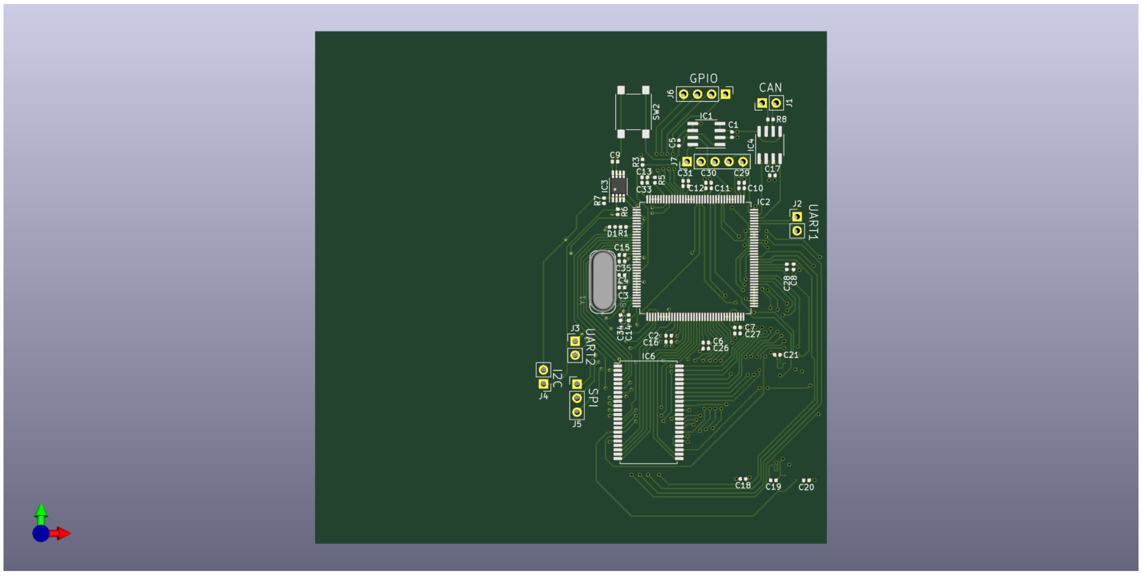

Figure 1*: The current OBC Engineering Model, if we removed one MCU and all NAND memories*

of on the OBC board.

- **Advantages** of this solution are:
  - **No need to communicate the photos to OBC via CAN or any other protocol**.
    - The camera interface requires photos to be stored immediatelly in an MCU memory.
    - A 2 MP photo requires **6 Mbits** of storage, which is less than what the internal RAM of any STM32 MCU offers.
    - The SU MCU will therefore have to use an external RAM, or send the data to OBC through a very fast communications protocol.
      - For a slow 1 MHz camera pixel clock, an 8 MHz serial connection is required.
      - Candidate protocols for this are USART, SPI, dual SPI, QUADSPI, or OCTOSPI. In other words, an extra heavy connection between the vessel and the OBC is required to transmit the data quickly enough.
    - Do note that JPEG compression might require the full data of the image to be available in the memory.
    - If the data is not required to be stored in a RAM memory, it can also be immediately persisted to the NAND flash instead.
    - The decision of whether to include MRAMs (with a maximum capacity of 16 Mbits) or regular static RAMs (with higher susceptibility to radiation) for the SU will depend on the amount of memory needed to store the photos, and their resolution.
      - The RAM will be used for very short bursts of time whenever a photo is taken, making radiation a minor concern.
  - Less usage of the CAN protocol, and of OBC resources.
  - NAND flash will be better protected from radiation, due to the vessel walls.
    - This protection is negligible. Aluminum shielding protects mostly from *ionizing dose* effects, not SEEs which are our main concern.
  - More space available on the OBC board.

- **Disadvantages** of this solution are:
  - Less available SU processing time (e.g. for focus correction)
  - Less available space on the SU board
    - This is negligible, as the SU will contain a separate PCB for the micro-controller.
- **Due to the reduced requirements of having the NAND Flash memories connected directly to the SU microcontroller, we opted for moving them within the SU vessel.**
- This creates a lot of empty space on the OBC board, which can be used for other components of the satellite (e.g. batteries or magnetorquers).
  - Alternatively, the OBC MCU and peripherals can be installed on the COMMs PCB, if space and Electromagnetic Compatibility allows.
  - A discussion was made to consolidate the OBC and COMMs functions within one microcontroller. However, since both MCUs will have different heavy functions, it was decided to not risk operations and separate the concerns.
- Data will be sent directly from the SU (in *space packet* format, see subsection 1.2) to COMMs. Files can be requested via the *ST[06] memory management* service, through the *object management* functions.

1.4 FYS!3 specifications overview

- Most points are already planned to be implemented.
- A configurable countdown timer had not been considered, but is planned to be added.
- Software patching was discussed in detail.
  - The presence of two NAND flash banks allows easy in-orbit patching. This immediately solves reliability & backup concerns, as the MCU can be running while being programmed.
  - In order to address the **uplink data rate demands**, we can:
    - Partially patch code, by specifying sections on the *ST[06] memory management* service.
    - Use *DEFLATE* compression that might reduce the size of the code to 1/4.
    - Use **binary diffing** methods that might reduce the size of the code to 1/10.

2 Incomplete discussions

- OBC subsystem requirements overhaul
- constexpr vs #defines
- Exception usage in C++
- Request success & error result