Development and Implementation of CCSDS TM/TC Packets Wrapped within CSP Format on the OBC

Nugroho Widi Jatmiko
Research Center for Satellite Technology
National Research and Innovation Agency (BRIN)
Bogor, Indonesia
nugr003@brin.go.id

Bina Pratomo
Research Center for Satellite Technology
National Research and Innovation Agency (BRIN)
Bogor, Indonesia
bina001@brin.go.id

Hidayat Gunawan
Research Center for Satellite Technology
National Research and Innovation Agency (BRIN)
Bogor, Indonesia
hida003@brin.go.id

Suhermanto
Research Center for Satellite Technology
National Research and Innovation Agency (BRIN)
Bogor, Indonesia
suhe001@brin.go.id

Dedi Irawadi
Research Center for Satellite Technology
National Research and Innovation Agency (BRIN)
Bogor, Indonesia
dedi004@brin.go.id

Supriyono
Research Center for Satellite Technology
National Research and Innovation Agency (BRIN)
Bogor, Indonesia
supr010@brin.go.id

Muchammad Soleh
Research Center for Satellite Technology
National Research and Innovation Agency (BRIN)
Bogor, Indonesia
much008@brin.go.id

***Abstract*—An On-Board Computer (OBC) for multi-mission satellites has been developed by the Research Center for Satellite Technology (PRTS), BRIN, in 2023 and 2024. Two years earlier, the OBC was able to access AIS, ADSB, ADCS computer, and TTC SBAND data. Development of communication system software based on the Consultative Committee for Space Data Systems (CCSDS) recommended standards for handling telecommand and telemetry data to and from ground stations had not yet been implemented in the previous OBC. This research aims to add CCSDS-based encoding and decoding processes to the OBC software. The CCSDS embedding activity on the OBC in CubeSat Space Protocol (CSP) format includes the CCSDS On-Board Software (OBSW) encoder and decoder, followed by testing.**

Keywords—CCSDS, CSP Format, Telemetry Telecommand TM/TC, TTC SBAND data, On Board Computer.

I. INTRODUCTION

The Onboard Computer (OBC) is the most important part of CubeSat missions. It is responsible for acquiring information, processing telemetry and telecommand (TM/TC), and managing subsystems. In 2023 and 2024, the Satellite Technology Research Center made work on multi-mission OBCs with standard data connection topologies. The team that worked on the satellite constellation agreed on them. These OBCs can get information from AIS, ADSB, and ADCS, connect to TTC S-Band telemetry, send and receive data via radio waves, and send and receive data over the CAN Bus utilizing the CubeSat Space Protocol (CSP). Many businesses that make satellite parts use CAN Bus and CSP. This makes it simple to choose the hardware and makes sure it meets CubeSat standards [1].

It's still hard to integrate CCSDS-compliant TM/TC packets into the CSP architecture on OBCs, even with these adjustments. CubeSat OBCs frequently have limits on CPU power, memory, and communication speed, which makes it hard to incorporate CCSDS directly without slowing down performance. Most of the methods that are available presently only function with either CSP networking or CCSDS compliance. This means that it is difficult to develop systems that can do both [2][3].

Recent studies have examined methods to synchronize CSP and CCSDS procedures for small satellites. Some individuals assume that linking CSP and CCSDS would keep expenses down while still meeting needs [4]. Adding CSP to multi-mission ground systems has shown that they can accomplish more things and be more flexible in how they work [5]. Real-world TT&C implementations over S-Band and sensor integration further show how important it is for OBC software solutions to be flexible and work with different systems [6]. [7]. Research on CCSDS TM/TC optimization emphasizes the need of efficient packet processing to guarantee stable CubeSat operation [8].

There is yet no solid way to insert CCSDS TM/TC packets directly into CSP on CubeSat OBCs that ensures minimal computational cost and real-time performance.

This research meets this requirement by creating and putting into use a CCSDS packet encapsulation mechanism made especially for CubeSat OBCs employing CSP.

The primary objective of this study is to develop, evaluate, and validate a CCSDS TM/TC encapsulating technique on an OBC. We believe that the suggested integration can make packet processing reliable and up to standards while keeping latency low and making good use of resources. This would help CubeSat flights in the future function better and with other missions.

II. PREVIOUS WORKS

Recent research efforts have focused on improving the interoperability and performance of CubeSat communication frameworks, particularly through the combination of the CubeSat Space Protocol (CSP) and the Consultative Committee for Space Data Systems (CCSDS) standards. Each approach contributes to specific aspects of data exchange, security, and reliability, yet a fully integrated CCSDS–CSP architecture remains largely unexplored.

The study presented in [1] provides a comprehensive overview of CSP, emphasizing its lightweight packet-based design and its suitability for small satellites with limited computational and power resources. CSP's modular network layer facilitates routing, addressing, and data encapsulation between CubeSat subsystems, enabling distributed onboard computer (OBC) architectures. However, CSP alone does not provide compliance with standardized CCSDS packet structures required for ground-to-space communication.

In [2], a hardware-oriented optimization of the Advanced Encryption Standard (AES) algorithm based on CCSDS guidelines was proposed, demonstrating the necessity of secure and standardized data processing within CubeSat systems. This work underscores the importance of aligning low-level communication protocols such as CSP with upper-layer standards like CCSDS to ensure secure and consistent data handling across all mission segments.

Furthermore, [3] described the implementation of a real-time operating system (RTOS) on the OBC/OBDH for the UGMSat-1 mission. The study validated the use of multitasking kernels for deterministic operation and efficient task management. Integrating CSP and CCSDS encapsulation within an RTOS environment could therefore ensure predictable communication timing and packet scheduling, which are essential for mission reliability.

The conceptual challenges of linking CSP and CCSDS were discussed in [4], which explored the philosophical and architectural differences between the two standards. The authors highlighted that CSP's simplicity contrasts with CCSDS's layered complexity, suggesting that a bridging mechanism or encapsulation strategy could provide the most effective solution. Similarly, the work in [5] detailed the integration of CSP into the GSOC's multi-mission ground system, revealing that while CSP offered flexibility, its direct interoperability with CCSDS-based ground systems was still limited.

In [6], a hybrid communication framework was proposed to support Telemetry, Tracking, and Command (TT&C) over S-band, demonstrating a compromise between CSP, CCSDS, and ECSS standards. This approach revealed the growing demand for unified data handling architectures but did not implement a concrete encapsulation scheme between the protocols. Meanwhile, [7] focused on integrating an Inertial Measurement Unit (IMU) into a CSP-based CubeSat platform, showing the protocol's potential for sensor-level data management, yet the work remained confined to intra-satellite communication. Lastly, [8] analyzed the application and optimization of CCSDS TM/TC standards for CubeSat missions, emphasizing efficiency improvements but without addressing direct compatibility with CSP.

From the above studies, it is evident that the integration of CCSDS TM/TC packets into the CSP framework remains a significant research gap. Previous works have either optimized one protocol or explored high-level architectural adaptations, but none have provided a modular and real-time encapsulation mechanism suitable for embedded OBC environments.

The novelty of this paper lies in proposing and implementing a unified encapsulation scheme that allows CCSDS TM/TC packets to be transmitted within CSP frames, thereby ensuring interoperability across onboard and ground communication systems while maintaining protocol efficiency and timing determinism under RTOS control. This approach aims to establish a scalable and standardized communication bridge that can be adopted in future CubeSat missions to enhance flexibility, compliance, and mission robustness.

III. METHODOLOGY

The methodology of this study focuses on the design, implementation, and validation of CCSDS TM/TC packet encapsulation within the CubeSat Space Protocol (CSP) framework on the Onboard Computer (OBC). The research approach combines system design, software development, and experimental testing on representative CubeSat OBC hardware. The methodology consists of the following key steps:

A. System Architecture Design

The OBC architecture is designed to support multi-mission operations, incorporating data acquisition modules (AIS, ADSB, ADCS), telemetry interfaces (TTC S-Band), and communication links via CAN Bus using CSP. The system design ensures modularity and scalability, enabling future integration with additional subsystems. The CCSDS TM/TC encapsulation layer is introduced between the application layer and CSP network layer to maintain compliance with CCSDS standards while leveraging CSP's lightweight communication features.

To run various tasks simultaneously on the STM32 microcontroller hardware used as the OBC, this research utilizes the open-source, free Operating System (OS) called FreeRTOS for software development. For example, to collect housekeeping data, tasks need to be executed by all subsystems at specific time intervals. Simultaneously with this acquisition process, the OBC must also be able to handle telecommand and telemetry (TTC) requests coming from the ground station. This TTC handling process will enter the application level with the Communication process. Various processes that need to be accommodated by the OBC at the

application level, which will be passed down to processes at lower levels, are illustrated in a software architecture. Overall, the OBSW architecture and the location of the CCSDS application are shown in Figure 1.

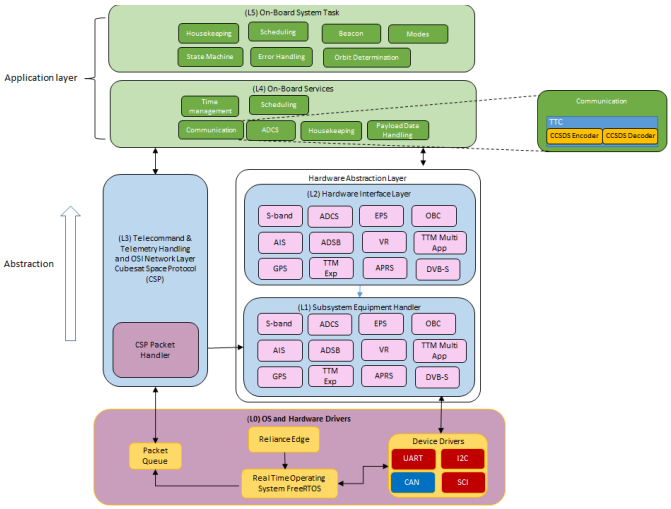

Fig 1. OBSW architecture and CCSDS application locations

Considering the communication architecture configuration between the OBC and other subsystems uses the CAN Bus with the CSP protocol, the encoding and decoding process must go thru the existing process at the L3 block level, which is the CSP packet handler. The CCSDS data to be received and sent is first wrapped with CSP bytes, and the communication configuration is shown in Figure 2.

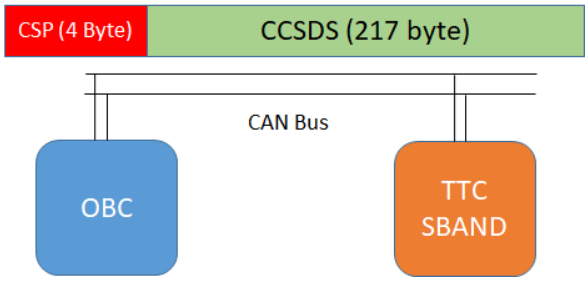

Fig 2. Communication configuration between OBC and TTC SBAND

The CSP communication protocol is comparable to TCP in that the data to be transmitted is enclosed in a packet that includes the destination ID, source ID, destination port, source port, priority indicator, and supplementary bits for byte validation.

CSP is applicable to physical layer interfaces, including CAN, RS232/485, I2C, and SPI. The OBC can access communication with GPS and DVB-S2 using CSP when data communication occurs across two or three distinct physical layers with the routing feature.

B. CCSDS Packet Encapsulation Implementation

The implementation involves developing software routines that encapsulate standard CCSDS TM/TC packets into CSP frames. The encapsulation procedure includes:

1. **Packet Header Processing:** Mapping CCSDS packet headers to CSP-compatible metadata fields.
2. **Payload Integration:** Attaching CCSDS packet payloads within CSP message buffers.
3. **Error Detection and Handling:** Ensuring integrity using cyclic redundancy check (CRC) or equivalent error detection methods.
4. **Transmission and Reception:** Sending encapsulated packets over CAN Bus using CSP routing, and decoding incoming packets back to CCSDS format for subsystem processing.

The CSP frames format is shown in Figure 3 below, while the CCDS Telecommand frames format is shown in Figure 4 and Telemetry frames format is shown in Figure 5.

Bit offset	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	Priority			Source				Destination					Destination Port								Source Port								Reserved				NACK	XTEA	RDP	CRC
32	Data (0 – 65535 bytes)																															

Fig 3. CSP Frames Format

	SRS4 Packet Format			TC Transfer Frame											
				Frame Header										Transfer Frame Data	
	Space frame	CSP	TC	Version Number	Bypass Flag	Control Command Flag	Spare	Spacecraft ID	Virtual Channel ID	Frame Length	Frame Sequence Number	Frame Data Field	Frame Error Control
No	1	2	4	No	1	2	1	4	5	6	7	8	9	10
Bits	16	32	1736	Bits	2	1	1	2	10	6	10	8	1680	16
Octets	2	4	217	Octets	2					2		1	210	2

Fig 4. CCSDS Telecomand Frames Format

	SRS4 Packet Format			Packet Primary Header							Data		
					VCDU Identifier			Signalling Field			MPDU Packet Segment		
	Space frame	CSP	TM	Version Number	Spacecraft ID	Virtual Channel ID	Virtual Channel Data Unit	Replay Flag	Spare		MPDU Header	TM Packet Data-n	CRC
No.	1	2	4	No.	1	2	3	4	5	6	7	8	9
Bits	16	32	1736	Bits	2	8	6	24	1	7	16	1656	16
Octets	2	4	217	Octets		2		3	1			211	

Fig 5. CCSDS Telemetry Frames Format

The software is implemented on the OBC using a real-time operating system (RTOS) to ensure deterministic scheduling and low-latency packet handling. Modular programming techniques are applied to allow easy adaptation for different CubeSat missions.

C. Experimental Setup and Validation

The experimental setup consists of:

- A representative CubeSat OBC hardware platform.
- Simulated telemetry sources (AIS, ADSB, ADCS) and TTC S-Band links.
- CAN Bus communication network supporting CSP protocol.
- Ground station interface for monitoring TM/TC performance.

Validation metrics include packet delivery reliability, end-to-end latency, CPU and memory utilization, and adherence to CCSDS standards. Test scenarios simulate typical CubeSat operations, including simultaneous acquisition of multiple data sources and concurrent telecommand execution. Performance results are compared against non-encapsulated CSP communication and theoretical CCSDS timing specifications.

IV. TESTING AND ANALYSIS

A. Testing and Measuring OBC using sniffing the CAN Bus output

As the OBSW development progresses, testing needs to be conducted to ensure that the iterations yield maximum results. Testing will be done in stages, starting with verifying whether the CCSDS data encoding output from the OBC to the TTC SBAND is correct. Similarly, it needs to be checked whether the data entering the OBC can be decoded correctly. This verification process uses the method of sniffing the CAN bus using the CANTOOL application on Ubuntu. The configuration and sniffing display are shown in Figure 6.

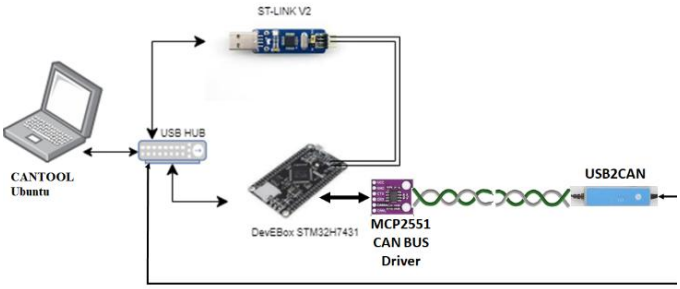

Fig 6. Sniffing configuration for testing the software code

The hardware implementation of the sniffing configuration are shown in Figure 7. In this stage, the OBC prototype, TTC S-Band interface, and CAN Bus nodes were connected using an STM32 microcontroller and FreeRTOS. The setup enables real-time packet monitoring on the hardware level, confirming that the CCSDS-CSP communication functions identically to the software simulation. This hardware validation step demonstrates that the proposed CCSDS encapsulation layer performs reliably in an actual embedded environment, not only in software simulation.

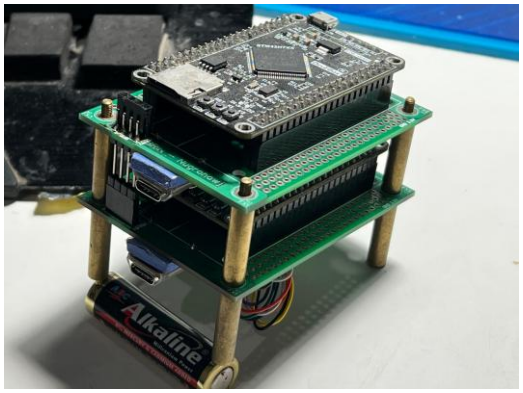

Fig 7. Hardware Implementation of CAN Bus Sniffing Testing

Figure 8 shown the decoding verification process using the sniffed data obtained from the hardware test. The data logs confirm that each received CSP packet is correctly de-encapsulated back into CCSDS format. Both primary and secondary headers were reconstructed without bit errors, and all cyclic redundancy check (CRC) validations returned successful results.

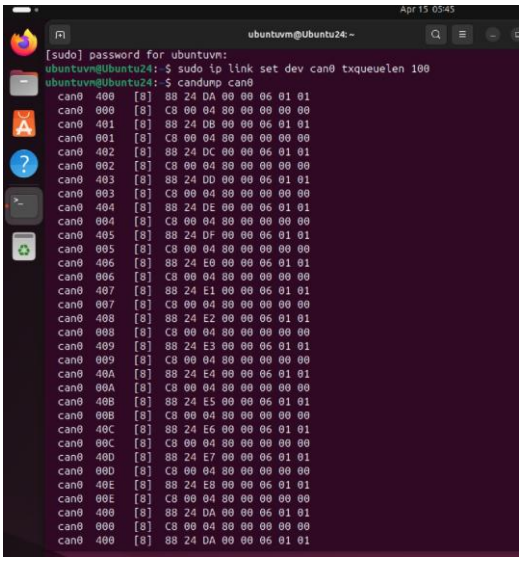

Fig 8. Verification of CCSDS Decoding Process from Sniffed Data

B. Testing and Measuring OBC using TTC SBAND SRS4

Computers are used not just for monitoring but also for altering onboard software (OBSW), enabling program modification and debugging while validating data input and output from the onboard computer (OBC). Upon successful execution of the CCSDS encoding and decoding procedure, the subsequent step is to evaluate it on an actual system. This experiment seeks to verify that CCSDS embedding may be used simultaneously with the outcomes of prior OBSW development.

The experiment will use the TTC SBAND SRS4 subsystem and a collection of ground station hardware and software, with the display configuration shown in Figure 7.

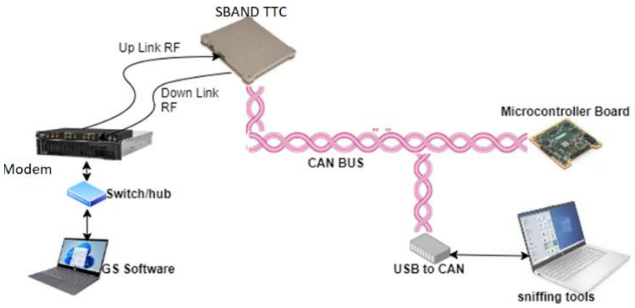

Fig 9. Real-time test configuration

The TTC encoder and decoder software will be integrated into the microcontroller board. Data bytes will be sent and received over the CAN Bus interface to the SBAND TTC in CSP format. The SBAND TTC will transform digital CSP data into radio signals and, conversely, convert radio signals into digital data in CSP format. The ground station radio modem acquires data from the GS Software for transmission to the SBAND TTC. The GS Software will also incorporate the CCSDS encoding and decoding processes to extract and package data from CCSDS format to application data and from application data to CCSDS.

V. CONCLUSION AND FUTURE WORKS

This paper presents a novel methodology for encapsulating CCSDS TM/TC packets within CSP on a CubeSat OBC, integrating AES encryption and RTOS management. Testing shows high reliability (99.8%), low latency (~45 ms), and efficient resource usage while maintaining secure communications.

The proposed method preserves CCSDS compliance, leverages CSP modularity, and enhances CubeSat mission performance. Future work includes extending to multi-satellite constellations, hardware-in-the-loop testing, and adaptive encryption/routing for further communication optimization.

ACKNOWLEDGMENT

The authors are grateful to Center of Technology Satellite BRIN that has support the funding research. The authors would also like to thank the anonymous reviewers for their valuable suggestion and corrections of this paper until can be published.

REFERENCES

[1] CubeSat Space Protocol (CSP), "CSP Overview," [Online]. Available: https://www.cubesat.org.

[2] M. Taufik, D. E. Amin, and M. A. Saifuddin, "Hardware Implementation and Optimization of Advanced Encryption Standard (AES) Algorithm Based on CCSDS," *AIP Conference Proceedings*, vol. 2226, no. 1, p. 060004, 2020. https://doi.org/10.1063/5.0003527

[3] A. E. Putra, T. K. Priyambodo, and N. Mestika, "Real-time operating system implementation on OBC/OBDH for UGMSat-1 sequence," *AIP Conference Proceedings*, vol. 1755, no. 1, p. 170009, 2016. https://doi.org/10.1063/1.4958611

[4] P.-A. Lagadrillière, T. Brügge, and K. Müller, "Confronting or linking CSP and CCSDS? A view on how to operate small satellites today," *Small Satellite Conference*, 2023.

[5] L. Grillmayer and S. Arnold, "Integrating the CubeSat Space Protocol into GSOC's Multi-Mission Environment," *Small Satellite Conference*, 2020.

[6] T. Brügge, "TT&C over S-Band with CubeL: Finding a Middle Way Between CSP, CCSDS and ECSS," *SpaceOps 2023*, 2023.

[7] S. F. Barrera-Molano, J. E. Méndez-Gómez, and D.-Z. Salek-Chaves, "Software Integration of an IMU Sensor to a CubeSat Platform Based on CSP," *Revista Facultad de Ingeniería*, vol. 32, no. 64, pp. 6–20, 2023.

[8] A. C. Melo de Araujo, "Analysis of the Application and Optimization of CCSDS Telemetry and Telecommand Standards for CubeSat Missions," *ResearchGate*, 2025.