Contribution ID: **31** Type: **talk**

An Open Source Implementation of the ECSS PUS Services in Rust

Saturday, 12 December 2020 14:15 (20 minutes)

We will present an ongoing project of software development for ECSS PUS-C services in the new robust and fast programming language Rust[1]. Vision Space Technologies[2] started this open-source project called Prust[3]. The standard ECSS-E-70-41C[4] is a packet utilization standard (PUS), defining the packet structure and a set of standard services. Its recent iteration PUS-C was published in 2016 and it's relatively new. PUS is the standard of choice by ESA, and only one spacecraft operated by ESA doesn't use PUS (OPS-SAT). A space project can choose to implement only those standard PUS services which suit its specific mission characteristics and requirements. Also since there can be additional requirements depending on the mission PUS also includes some guidelines and rules for how these extensions should be implemented. It is clear that implementations of PUS needs to be robust and safe given the usage of these softwares, also it should be maintenance-cost effective since requirements are very dynamic. We will present how we tackled these requirements in Prust project.

Prust project aims to create software that is reliable, fast, and encapsulated in a way that only the desired functionalities are dynamically loaded to the spacecraft. This is why we chose to implement the software in Rust. C remained the only alternative for a long time because it was faster than other programming languages. The speed of C came from not checking memory safety at runtime. But with its features, Rust offered a cost-free way of ensuring memory safety. It also offered usage of High-Level programming features with System-Level performance. This paper[5] says with the adoption of Rust these mechanisms will become commoditized. Prust was used in VST104 device which is another project of Vision Space Technologies.

Prust offers potential ease of abstraction while being fast and robust. Two complete PUS services and some sub-services are implemented already. Implemented services are sufficient for controlling and monitoring spacecraft peripherals. With the convenience of Rust, well-written error propagation and error-handling features (Similar to Exceptions in higher-level languages) are implemented without cost. Which is handled poorly in C in terms of maintenance cost and safety because of the nature of the language itself.

We will present why and how Prust is an important project because it uses a new programming language which can be the future of embedded programming we will also show the following implemented PUS services;

1. **Function Management (Service 8)**; One request (TC) and it executes a function defined by the user by giving the name of the function. Which offers a potential to change the usage of the spacecraft dynamically.
2. **Request Verification (Service 1)**; 9 response (TM) types which every one of them indicates different states for the sub-services send (for example failure,success etc..)
3. **Housekeeping (Service 3)**; Plenty of TM and TC packets in it. It helps to have reports of the peripherals connected to the device. Implemented partially to monitor peripheral values from the spacecraft.

Prust project is still active and we would like to contribute by showing you the details of this project.

References

[1]: https://www.rust-lang.org/

[2]: https://www.visionspace.com/

[3]: https://github.com/visionspacetec/Prust

[4]: ECSS-E-ST-70-41C – "Telemetry and telecommand packet utilization"
European Cooperation for Space Standardization
http://www.ecss.nl/, 15 April 2016

[5]: Abhiram Balasubramanian, Marek S. Baranowski, Anton Burtsev, Aurojit Panda, Zvonimir Rakamarić, and Leonid Ryzhyk.
System Programming in Rust: Beyond Safety. In Proceedings of the 16th Workshop on Hot Topics in Operating Systems (HotOS'17).
Association for Computing Machinery.
doi:10.1145/3102980.3103006, 2017

[6]: https://github.com/visionspacetec/VST104

Primary author: ÖZLEYEN, Selman

Co-authors: FEITEIRINHA, Jose (VisionSpace Technologies GmbH); GEIB, Filip (VisionSpace Technologies GmbH)

Session Classification: Room #2