[SSC23-P4-26]

Examination of the Application Support Layer of CCSDS SOIS using SpaceWire Communication Stack and Low-code User Application

Mitsutaka Takada, Hiroaki Takada
Center for Embedded Computing Systems, Graduate School of Informatics, Nagoya University
Furo-cho, Chikusa-ku, Nagoya, Aichi 464-8601, Japan; +81-52-789-4228
mtakada@nces.i.nagoya-u.ac.jp, hiro@ertl.jp

Takayuki Ishida
Research and Development Directorate Japan Aerospace Exploration Agency (JAXA)
3-1-1 Yoshinodai, Sagamihara, Kanagawa 252-5210, Japan
ishida.takayuki@jaxa.jp

Keiichi Matsuzaki
Institute of Space and Astronautical Science (ISAS) Japan Aerospace Exploration Agency (JAXA)
3-1-1 Yoshinodai, Sagamihara, Kanagawa 252-5210, Japan
matsuzaki.keiichi@jaxa.jp

ABSTRACT

As a SmallSat grows, there is an increasing need for high-speed and high-functioning sub-networks, such as SpaceWire, to connect equipment to the OBC and store large amounts of optical image data. SpaceWire has the ability for time-sensitive delivery, fast communication, and flexible network topology construction.

JAXA/ISAS has developed SIB2, a low-code user application that designs telemetry commands based on a database stored on the satellite. SIB2 has a code generation function called SIB2Generator, which outputs code that performs the behavior of telemetry commands during sender/receiver and is used as a template for user applications. On the other hand, the Application Support Layer of CCSDS SOIS is positioned between sub-networks and user applications. And then, service types are defined in the Application Support Layer, but no concrete implementation interface is specified.

Therefore, we have implemented an examination of the Command and Data Acquisition Services layer using SIB2 and the SpaceWire communication stack. In the executed study, we adopted the SpaceWire RMAP Library, a class library that makes SpaceWire communication. Using this library, we have designed and developed the Application Support Layer, integrating the SpaceWire application stack based on the SIB2 database design and the user application code generated by SIB2Generator.

INTRODUCTION

Nanosatellites have become the mainstream of launched satellites in the past few years due to significantly reduced development and launch costs and more simple functional configurations than mid-to-large-size satellites.

As a result, the range of mission applications has expanded, and nanosatellites are now used for single and multiple missions. Small satellites vary from 1U (10cm square) to 12 or 16U. However, sensors, actuators, and telemetry devices are often connected directly using the IO of the CPU installed in the OBC, and the bus wiring of the onboard equipment boards is reduced by using backplanes to connect the boards. The bus wiring of the onboard equipment boards is reduced by using backplanes to connect the boards.

With this trend toward larger satellites, IO control and management during equipment connection is becoming more complex with conventional connection methods, and sub-networks using standard buses are being considered.

A layered set of communication services for flight avionics is CCSDS Spacecraft Onboard Interface Services (SOIS).1 SOIS is intended to cover most onboard communication requirements and consists of a sub-network and Application Support Services.

Services are defined in the Application Support Layer but are not specified concerning specific implementation interfaces.

In terms of MBSE in satellite development, JAXA/ISAS has developed the Functional Model of Spacecrafts (FMS),2 which is a functional model of a satellite, and the Spacecraft Information Base version 2 (SIB2),3 which is a database that holds satellite information based on the FMS. SIB2 is a database that stores satellite information based on the FMS. The SIB2 is mainly intended to centralize the design and management of telemetry command information necessary for satellite operations and to be used as a data format for satellite and ground-based data. To facilitate the development of applications, the SIB2Generator has been developed and is used to output user applications that process satellite telemetry commands in auto code from SIB2 information.

Therefore, we extracted the part corresponding to the services of the application support layer of SOIS from the low-code user application output by SIB2Generator and studied the implementation interface corresponding to the application support layer. Then, we used the SpaceWire communication stack we have already developed as a subnetwork layer of SOIS and implemented the interface using RMAP and SpacePacket protocol as a use case. Finally, we implemented the interface as a use case and verified the communication using an actual board with the same application source code as in the PC simulation environment.
The stack configuration combining the validated user application implemented interface, and SpaceWire protocol is shown in Figure 1.

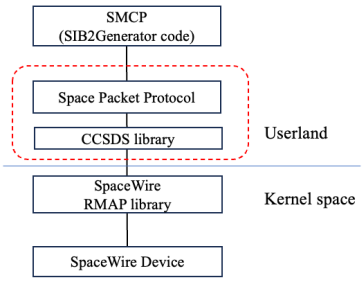

Figure 1: Protocol stacks examined.

CCSDS SOIS

The Spacecraft Onboard Interface Services (SOIS) concept is intended to allow applications to interact with and provide solutions for defined standard onboard data services.

Furthermore, considering the differences in functionality supported by the different data link layers, the SOIS subnetwork services provide a standard interface for communication over a single data link and a common aggregation of services for higher-layer applications. An overview of the architecture defined in SOIS is shown in Figure 2.

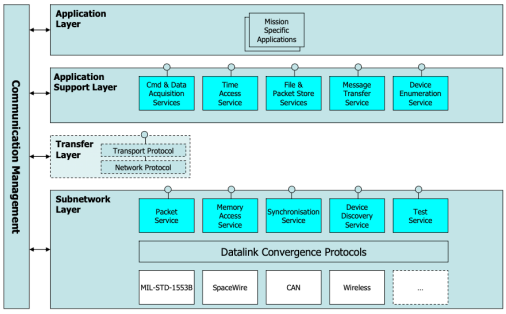

Figure 2: SOIS architecture overview,1

Application Support Layer

The Application Support Layer aims to provide common services required for applications on any processing data system on a spacecraft. These services focus on separating the application from the underlying topology and communication architecture on the spacecraft and transmitting service protocols.

This paper investigates the similarities of Command and Data Acquisition Services (CDAS), which may be relevant to telecommand processing and is the target of SIB2.

CDAS is intended to provide a method of accessing hardware devices such as sensors and actuators with minimal overhead,

1) Device Dependent Diver (DAS) provides drivers for basic reading and writing from devices regardless of their location, 2) Standard Device Driver (DVS) provides virtual driver references for physical drivers, 3) Device Data Pooling Services (DDPS) maintains the value state of many devices.

SIB2 is intended to maintain data necessary for satellite operation (including sensors and actuators) and encompasses some of the functions of DDPS.

While DAS and DVS have abstract device access interfaces provided by the Flight Software platform, we dared not to consider virtual drivers in this hierarchy but to replace them as a mapping between Telecommand data and device actuator data. The following is a brief overview of the SIB2.

In SIB2, the code output by SIB2Generator is based on access to sensor actuators by memory map. In the case of access other than by memory map, a driver is written for each target device.

SIB2 AND SIB2GENERTOR

As a database for storing information about satellites, JAXA/ISAS designed and developed SIB2, which is used for satellite development in JAXA/ISAS scientific satellite missions.

Furthermore, to shorten the development period of onboard software spacecraft, we have developed the SIB2Generator, which generates source code for command telemetry processing from SIB2 and provides a low-code user application environment.

The SIB2Generator automatically generates command processing and telemetry processing as user applications, while the OS for running onboard applications and the middleware for processing SpacePacket are prepared separately. Figure 3 shows the development process of a user application using SIB2Generator.

The user only needs to encode the packet sending/receiving part of the application, which is assumed to be implemented in low code as an onboard application. The SIB2Generator has no specific API for packet transmission/reception processing, which is written according to the subnetwork protocol and OS processing.

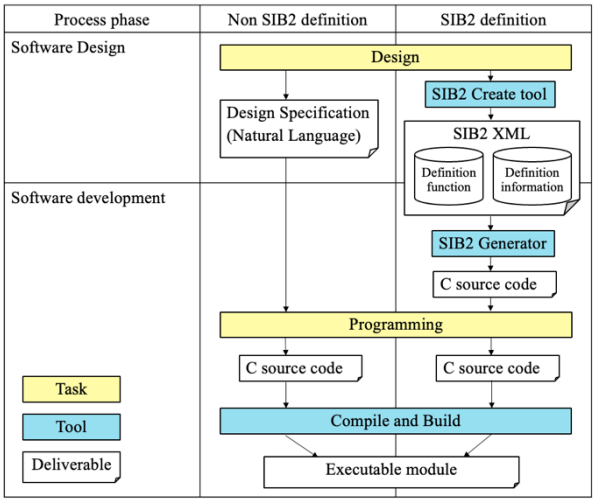

Figure 3: Development process using SIB2.

IMPLEMENTATION OF SOIS APPLICATION SUPPORT LAYER

In examining the support layer, we positioned the packet transmission and reception processing as the application support layer of SOIS and the middleware as the subnetwork support layer so that the packet processing would conform to the CCSDS protocol.

First, it was found that when a user application generated by SIB2Generator first receives a telecommand, decoding is performed assuming CCSDS SpacePacket.

After decoding the received telecommand, the SIB2Generator checks whether the command data matches the SIB2 database and performs the processing registered in SIB2.

SMCP, SCDHA2

In considering the application support layer of SOIS, the equivalent protocols in Japan are the Spacecraft Monitor and Control Protocol (SMCP)4 and the Standard of Communications and Data-Handling Architecture (SCHDA)5.

The overall concept of SCHDA is a large part of the CCSDS Recommendation. Still, it introduces several concepts beyond the Recommendation and presents constraints on the scope of the parametric options specified by the CCSDS Recommendation.

The SMCP provides the following services as a protocol to realize FMS and SIB2. (Table 1)

Table 1: SMCP Telecommand / Telemetry services

Telecommand	Telemetry
ACTION	VALUE
GET	NOTIFICATION
SET	ACK
MEMORY LOAD	---
MEMORY DUMP	MEMORY DUMP

As a use case, we developed a processing code compatible with ACTION Telecommand / ACK Telemetry based on the code generated from SIB2 information.

The main issue to be considered is where to copy the data handled in this paper between the kernel mode managed by the OS and the userland.

The processing code generated by SIB2Generator is based on the principle of first come, first served, and did not assume that the processing context is switched during processing.

Therefore, by copying data packets once between this hierarchy and the subnetwork, an OS with memory protection can treat the communication stack below the subnetwork in kernel mode and the upper levels of this hierarchy as data processing in userland.

SpaceWire RMAP Library and HAL

It was developed as a common software library that handles functions related to SpaceWire / RMAP Library.6 The library is written in the C++ language. It is implemented with few dependencies on external libraries, making it usable in general-purpose OS (Linux, macOS) and RTOS environments that can handle POSIX. The SpaceWireIF class is positioned as an abstract wrapper class for the SpaceWireIF hardware and OS-loaded driver. In addition, this verification will be performed in two environments: a PC simulation and a SpaceWire device. We have studied and implemented a reference implementation of the SpaceWire HAL,7 which enables SpaceWire communication with the same interface even if different SpaceWire devices are used.

CONCLUSION

We studied the implementation interface of the SOIS application support layer. As a use case, we used a user application that transmits and receives data using SIB2Generator based on data designed in SIB2, a satellite operation database. As a lower-layer subnetwork, we assumed SpaceWire communications and used the SpaceWire RMAP Library, which is capable of RMAP and CCSDS Packet transmissions.

Using these software modules, we extracted the necessary functions for the application support layer and created an implementation interface. Using the software modules studied in the use case, we confirmed the operation in a PC simulation environment and communication using a SpaceWire IF board and confirmed the abstraction of devices in the application support layer (Figure 4).

In the future, we would like to study implementation interfaces for SOIS application support layer services other than CDAS, such as Time Access Service and File & Packet Store Service.

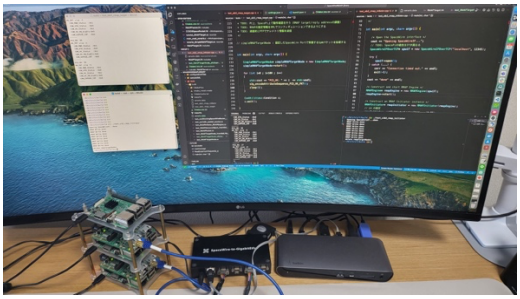

Figure 4: SpacePi and PC Simulator

References

1. "Spacecraft Onboard Interface Services, " CCSDS Green book, CCSDS 850.0-G-2, December 2013.
2. "Functional Model of Spacecrafts," JERG-2-700-TP001, March 2020.
3. Matsuzaki, Keiichi, et al, "Automatic generation of on-board software from the model–Spacecraft Information Base Version 2", TRANSACTIONS OF THE JAPAN SOCIETY FOR AERONAUTICAL AND SPACE SCIENCES, AEROSPACE TECHNOLOGY JAPAN.
4. "Spacecraft Monitor and Control Protocol," JERG-2-700-TP002, December 2019.
5. "End-to-End Protocol Architecture," JERG-2-400-TP102, December 2019.
6. Yuasa, Takayuki, et al, "SpaceWire/RMAP-based data acquisition framework for scientific instruments: overview, application and recent updates," International SpaceWire Conference 2010, 2010.
7. Takada, Mitsutaka, et al, "SpaceWire hardware abstraction layer Considerations," 2022 International SpaceWire & SpaceFibre Conference (ISC), IEEE, 2022.