DOCUMENT BASE PROGRAMMING SYSTEM 
TO REALIZE SEAMLESS LINKING BETWEEN ON-BOARD 
SOFTWARE AND GROUND OPERATING SYSTEM 

*Shinichi Kimura¹, Yuki Asakura², Hiroaki Doi³, 
Yasuhiro Nakamura⁴ 

¹Tokyo University of Science, 2641 Yamazaki, Noda, Chiba 278-8510, Japan, E-mail: kimura@rs.noda.tus.ac.jp 
²Tokyo University of Science, 2641 Yamazaki, Noda, Chiba 278-8510, Japan, E-mail: asakura@noda.tus.ac.jp 
³Tokyo University of Science, 2641 Yamazaki, Noda, Chiba 278-8510, Japan, E-mail: doi@noda.tus.ac.jp 
⁴Tokyo University of Science, 2641 Yamazaki, Noda, Chiba 278-8510, Japan, E-mail: nakamura@noda.tus.ac.jp 

ABSTRACT 
In satellites, on-board software is required to perform complicated mission sequences and autonomous scheduling, carry out preliminary data processing, and manage a variety of on-board equipment. The reliability of the on-board software strongly affects the reliability of the satellite itself. Therefore, to carry out complicated small-satellite missions, the on-board software needs to be both complex and reliable. To meet such requirements for the command and telemetry functions, we propose an automatic software generator that generates on-board software and a database for the ground operating system from the command and telemetry definition documents. Using the software generator, we can reduce the software-development load and avoid human error, even if commands and telemetry are modified in an ad-hoc manner during the development process. We can also easily cope with user preferences and the variation of software depth during a mission.

1 INTRODUCTION 
Due to the incremental complexity of small-satellite missions, the requirements for on-board software are becoming increasingly complicated. On-board software is needed to perform complicated mission sequences and autonomous scheduling, carry out preliminary data processing, and manage a variety of on-board equipment. The reliability of the on-board software strongly affects the reliability of the satellite itself. Therefore, to carry out complicated small-satellite missions, the on-board software needs to be both complex and reliable.

To meet such requirements, we propose and investigate a standardized software framework for small satellites. This software framework is expected to enhance software productivity and reliability by improving the reusability of the software resources in the standard software structure. We are currently applying it to two kinds of hardware platforms and the three satellites developed in the “Funding Program for World-Leading Innovative R&D on Science and Technology (FIRST Program).”

With the increasing variety of small-satellite applications, the telemetry and command functions are also becoming increasingly complicated. To realize such complicated applications, the on-board software must take into account the command and telemetry requirements of a mission [1]-[3]. In addition to the mission itself, commands and telemetries are also highly dependent on user preferences. Therefore, the software dealing with commands and telemetries needs to be customized for each particular mission, and is difficult to accommodate in a standardized framework.

Furthermore, the command and telemetry code is likely to be added to or modified during the satellite-development process. When these additions or modifications are made, the on-board software has to be updated in order to register them. Moreover, these changes affect the ground operation system. Therefore, the on-board software must correspond not only to the mission design but also to the operational database of the ground system. This is despite the fact that the operational database and the on-board software have not been developed in tandem in most traditional systems. Therefore, the command and telemetry functions of the on-board software must correspond properly not only to the user preferences but also to the ground operating system. If we expect to manage such consistency by manual operation alone, unexpected inconsistencies might arise from human error.

To meet such requirements for the command and telemetry functions, we propose an automatic software generator that generates on-board software and a database for the ground operating system from the command and telemetry definition documents. Using the software generator, we can reduce the software-development load and avoid human error, even if commands and telemetry are modified in an ad-hoc manner during the development process. We can easily cope with user preferences and the variation of software depth during a mission [1]-[3].
In this paper, the basic concept of an document-based programming system for on-board software is introduced. In this study, we propose a document-based software-code generator and a ground-operation-system database generator, which automatically generate an operational database and on-board software from satellite design documents. In addition, we propose a distributed and hierarchical method of managing the satellite design documents. This system is useful for the development of software for small satellites. The system has been used successfully in the development of the Hodoyoshi-3 and -4 satellites and the PROCYON small-size deep-space probe, as well as many software resources that can be utilized on further missions. We now plan to use this system for the development of IDEA-OSG-1, a space-debris monitoring satellite, and for ADRAS-1, a space-debris demonstration mission.

In this paper, the software development system is introduced in outline, along with examples of its use.

2 HARDWARE PLATFORM

For the first hardware platform for the software development and verification system, we selected a small-sized SOI-SOC OBC (SOBC) for the small satellites of the “Funding Program for World-Leading Innovative R&D on Science and Technology (FIRST Program).” SOBC consists of three segments: a CPU board, an IF board, and a power board, as shown in Fig. 1. The CPU board is the main processing unit, consisting of RAM, a program flash memory, and an FPGA for the chipset and SOI-SOC processor. The SOI-SOIC processor was developed for dual use in space and consumer applications, and it has a high radiation tolerance. The IF board consists of FPGAs and various types of line drivers. It functions as a translator between SpaceWire and various types of interfaces. In contrast, the CPU board is dedicated to SpaceWire for its interface. The IF board is suitable for various interfaces depending on the application, and acts as a translator for SpaceWire. Since the CPU board obtains four SpaceWire ports, it can utilize a maximum of four IF boards and can easily expand its interface capability. The power board generates power for several CPU boards and IF board sources. It also performs a hardware function known as Watch-dog Box (WDB) (Fig.2), which is triggered by an external reset cue.

Fig. 1 SOBC

Fig. 2 Watch-dog Box (WDB)

3 SOFTWARE FRAMEWORK

The macroscopic structure of the on-board software that is constructed using the software development system is illustrated in Fig. 3. The software is constructed on four major levels: (1) the operating-system level, (2) the driver and middleware level, (3) the SDK level, and (4) the application level.

The operating system performs basic functions such as memory access and task management. The on-board computers of the FIRST Program adopt TOPPERS, which is a µTRON-based real-time operating system, whereas SH4-BoCCHAN-1 adopts Linux. The operating-system level compensates for the differences in the CPU structure.

The driver and middleware level functions as the physical-level interface between the CPU and peripheral components. As mentioned in the previous section, the SOBC interface can be customized using the IF board. Such variations in the interface structure are compensated for by the FPGA, which acts as a link between SpaceWire and various kinds of interface. For the software, the driver and middleware level performs the setting and interface functions of the FPGA.

The SDK level provides interconnections between applications and the driver and middleware level in an abstract manner. It accesses the sensors intermittently, the data of which are converted into a generalized structured memory map that applications can then access to obtain sensor information. They are free from the hardware interface process of the peripheral equipment. When the applications need to send commands to the peripheral equipment, an application request calls the generalized function. The SDK level translates the commands for transmission to the peripheral equipment with the correct timing, and...
generates system and connects science has developed a system that
4
points.

Several entry points are predefined, such as a 50
parameters are lower than a predefined threshold.

Event driven entry points are triggered by commands from
real at certain time intervals
entry point: time
time control, such as altitude control. Command
-driven entry points are triggered by commands from
the ground or the on-board computer networks. Event-driven entry points are triggered by predefined
conditions, such as situations in which certain
parameters are lower than a predefined threshold. Several entry points are predefined, such as a 50-ms
interval time-driven entry point and a UVC event-driven entry point; users can also add their own entry
time and command information in a satellite
design document such as the telemetry and command
definition tables. Figure 6 illustrates the flow of this
system. A user can transmit their preference directly
to both the on-board software and the operational
database of the ground system simply by editing the
telemetry and command definition tables. This system
has two pieces of conversion software, referred to as
the “Command and Telemetry Code Generator” and
the “Ground System Database Generator.” These
generators reference the telemetry and command
information in the definition tables, and automatically
generate on-board software and the ground-operation
system database. This therefore enhances the
productivity overwhelmingly by realizing these
processes automatically, which can prevent the
occurrence of human error and improve the reliability
of the system. Moreover, although this system was
developed originally for the three satellites of the
“Funding Program for World-Leading Innovative
R&D on Science and Technology (FIRST Program)”,
it can also be used for the on-board software and
ground operation system of other satellites. By
extracting and standardizing the differences in
satellite structure and by defining the conversion
process as a meta-database, this system is able to
absorb the differences in satellite systems despite the
different configurations of components and functions
of the on-board software of each individual satellite.

4.2 Example of development
We use Excel workbooks to create the satellite
design documents, and we use C# on Microsoft
Visual Studio 2010 to create the conversion software.
An automatic code generator is essentially a piece
of conversion software. Figure 7 shows the software
window of the telemetry- and command-code
generators. The telemetry-code generator creates C-
language files that are entitled tlm_frame_aobc.c,
tlm_frame_mobc.c, tlm_frame_aobc.h and
tlm_frame_mobc.h. The command-code generator
also generates C-language files, which are named
cmd_analyze_aobc and cmd_analyze_mobc. These
files are modularized and classed by component,
function and sensor data. The sensor data are stored in
a predetermined memory in order to allow variables
to be referenced. Moreover, these files are intended
for generating a telemetry frame that packs the
telemetry data to be transmitted. We can choose
which workbook we want to use and in which place
to generate the software.
4.3 Coding on anomaly treatments from satellite design documents

The on-board computers have to detect and treat anomalies autonomously from the status information of the on-board components in order to protect the satellite. Such anomaly treatment is essential for the satellite’s survival, so the coding of such processes has to be both correct and comprehensive.

The anomaly-treatment process is coded in essence as a condition identification of the on-board parameters. In many cases, this identification process includes a number of parameters and complicated logical combinations of the parameter conditions.

We developed a GUI editing tool for such anomaly treatment in combination with an anomaly-treatment document (Fig. 8). The tool generates a logical flow chart automatically from the anomaly-treatment document, which is then easy for the designer to understand and is also effective in avoiding misunderstandings in the anomaly-treatment process. The edited result can be saved as an anomaly-treatment document, and is automatically convert into a program code of the on-board computer. As for the parameters, the GUI editing tool links automatically with the telemetry and command database, where all the parameters are defined. The calculation function between parameters can then be utilized. The tool enables the designer to design the treatment process in terms of an intuitive image.

4 HARDWARE IN THE LOOP VERIFICATION SYSTEM

Software verification is essential for achieving software reliability. In some cases, software verification based entirely on fundamental logic might not be sufficient. In some types of software, the verification process has to take account of sensitive performance parameters such as the response time. In the future, we hope to verify the implementation process for actual on-board computers, and we expect that it will be able to verify whether the implemented software can perform the same function as a logic-level simulation. To realize these requirements, the software should be verified under the conditions in which it will be implemented in a real OBC. Even though such hardware-based verification can be realized using an assembled satellite model, the simulation conditions are constrained by the hardware limitations.

Therefore, we developed a hardware-in-the-loop OBC software verification system in which the performance and interface of the peripheral equipment are simulated by a PC simulator. In this way, closed-loop simulation can be realized using a real OBC. Using this verification system, the software can be verified in a realistic situation on a real OBC, and the user can verify the implementation process. Because the software runs in a real OBC, the sensitive timing and hardware performances are included naturally in the verification process. Since the PC simulates the external conditions, nominal or non-nominal simulation conditions can easily be set without hardware limitations.

LabVIEW is used as the platform for the hardware interface and simulation. LabVIEW is a GUI-based hardware interface control language that was developed by National Instruments. The interface software can be constructed by interlinking software blocks in a GUI manner, and the software structure is easily understood. In the software verification system, we developed custom modules that correspond to well-known sensors and actuators, including their interface and performance parameters. Users can construct their own simulation systems easily by
dragging and dropping the icons of the components used in their own satellites.

Fig. 9. Hardware in the Loop Software Verification System

5 CONCLUSION

In this study, we introduced a document-based software-code generator and a ground-operation-system database generator, which automatically generate an operational database and on-board software from the satellite design documents. This system is being used successfully in UNIFORM-1, and HODOYOSHI-3 and -4. We plan to improve this system and apply it to other hardware platforms.

Acknowledgement

This research was supported by the Funding Program for World-Leading Innovative R&D on Science and Technology (FIRST Program).

References