IN-ORBIT IMPLEMENTATION OF ERROR PATCHING METHODS FOR LAPAN-A3/IPB OBDH FIRMWARE SYSTEM (IMPLEMENTASI METODE PENAMBALAN KESALAHAN DIORBIT PADA SISTEM PERANGKAT LUNAK OBDH SATELIT LAPAN-A3-IPB)

Muhammad Taufik[1], Wahyudi Hasbi[2], Abdul Karim[3]
[1,2,3]Pusat Teknologi Satelit
Lembaga Penerbangan dan Antariksa Nasional (LAPAN)
Jl. Cagak Satelit km. 0,4 Rancabungur, Bogor 16310
[1]e-mail: muhammad.taufik@lapan.go.id

Diterima: 7 September 2018; Direvisi : 7 Februari 2019; Disetujui : 5 Maret 2019

ABSTRACT

OBDH (On-board data handling) is a satellite subsystem that receives, processes, decides and executes commands from and to satellites. OBDH is built on two systems namely hardware and software integrated system (firmware system). In terms of hardware, OBDH uses a processor with 32bit RISC architecture, 128/256 Kbyte internal memory and a firmware system that is built using primitive programming. This programming uses the super loop architecture program and interrupt to manage the system to function properly. Problems occur when an error occurs in one of the functions in the interrupt routine resulting in failure of interpretation of commands or data from satellite sensors. This paper describes the implementation of the error patching methods on the LAPAN-A3/IPB Satellite OBDH firmware system in order to keep the system working well. Initial verification through testing on the ground have been successfully performed using engineering model of OBDH and hardware in the loop simulators (HWIL) module. Based on the test results, implementation on satellite has also been successfully done.

Keywords: *error patching methods, OBDH, LAPAN-A3/IPB Satellites*

ABSTRAK

OBDH (On-board data handling) merupakan salah satu subsistem satelit yang berfungsi menerima, mengolah, mengambil keputusan dan mengeksekusi perintah dari dan ke satelit. OBDH dibangun berdasarkan dua buah sistem yaitu sistem perangkat keras dan perangkat lunak yang terintegrasi (sistem firmware). Dari sisi perangkat keras, OBDH menggunakan prosesor dengan arsitektur 32bit RISC, 128/256 Kbyte memori internal, dan sistem firmware yang dibangun menggunakan pemrograman primitif. Pemrograman ini menggunakan arsitektur program super loop dan interrupt untuk mengelola sistem agar dapat berfungsi dengan baik. Permasalahan terjadi ketika terjadi kesalahan pada salah satu fungsi pada rutin interrupt sehingga mengakibatkan kegagalan interpretasi perintah atau data dari sensor satelit. Paper ini menjelaskan mengenai implementasi metode penambalan kesalahan pada sistem firmware OBDH satelit LAPAN-A3/IPB yang bertujuan untuk menjaga agar sistem tetap bekerja dengan baik. Verifikasi awal melalui pengujian telah berhasil dilakukan mengunakan engineering model OBDH dan modul hardware in the loop simulators (HWIL). Berdasarkan hasil pengujian, implementasi pada satelit juga telah sukses dilakukan.

Kata kunci: *metode penambalan kesalahan, OBDH, Satelit LAPAN-A3/IPB*

1 INTRODUCTION

LAPAN-A3 is the third-generation experimental satellite LAPAN after the LAPAN-A2/Orari and LAPAN-Tubsat satellites. This satellite has various missions including earth observation using 4 band line scanners and digital cameras, monitoring ships with AIS content and the scientific mission using a magnetometer (Hasbi & Suhermanto, 2013). LAPAN-A3 is equipped with PCDH (Power Control and Data Handling). PCDH is a satellite subsystem that combines functions to control power (Power Control Unit) and handling data from and to other subsystems (On-Board Data Handling Unit). OBDH has the function to receive, manage, make decisions and execute commands from and to the ground station as well as from the satellite subsystem.

Compared with the previous LAPAN satellites, OBDH LAPAN-A3 has many enhancements from the satellite bus side, including OBDH. At LAPAN-Tubsat, OBDH uses a 32bit RISC processor, 4/16 kByte internal memory with some limitations on the system firmware (Hardhienata & Triharjanto, 2007). While at LAPAN-A2/Orari, in terms of hardware and system firmware OBDH is not much different from the LAPAN-A3/IPB satellite with several different functions on the command to satellite (Hasbi, & Karim, 2013). OBDH LAPAN-A3/IPB uses a 32bit RISC processor with 128/256 Kbyte internal memory, 1 Mbyte external static RAM, and 1 MB external flash memory. Optimization is also a lot done in terms of software, the OBDH contains the satellite operation system software (Firmware System), which allows direct control of all the functionalities via the ground station as well as a highly autonomous operation controlled by time or event triggered tasks (Hasbi, et al, 2016).

The system firmware is designed and built using primitive programming. This programming uses the superloop programming architecture and interrupt to manage the system to function properly. Superloop or main loop is a programming architecture that makes the system to run programs repeatedly and continuously, other tasks can be implemented separately that are triggered through an interrupt.

The OBDH LAPAN-A3 / IPB firmware system consists of two parts, namely the initial initialization of the BIOS (including hardware and software), the second is the main loop. The main loop does only three things, i.e. reset the watchdog counter, set the CPU to sleep mode, check errors in the boot program memory area, etc. Almost all functions are triggered by interrupt routines, one of them is ACS (Attitude control System) control loop, basic typical execution scheme of OBDH is shown in Figure 1.

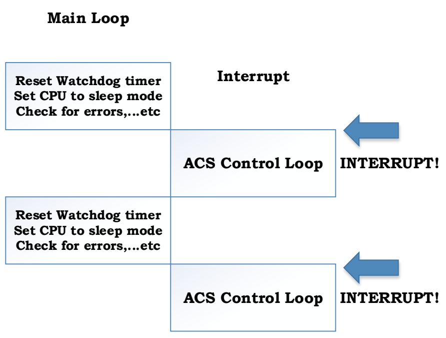


Figure 1: Basic typical execution scheme of OBDH Firmware System

ACS control loop is a function routine that functions to move the satellite in accordance with the selected satellite mode including manual mode, automatic nadir mode, automatic target mode, and others. Interpreting the value of the sensor, processing the data and moving the actuator is done in this routine. Errors in routine functions can cause data interpretation errors to control imperfect satellite attitudes.

Error patching is one way to maintain a device from a system failure due to an error that appears (Hayden, et al., 2012) (Kraft, 2011) (Trümper J, 2012). Some embedded system patching and updating methods have been successfully implemented (Ekman, & Thane, 2007) (Calder, & Kutt, 2009) (Savitha A, 2017) (Pingale P, 2016).

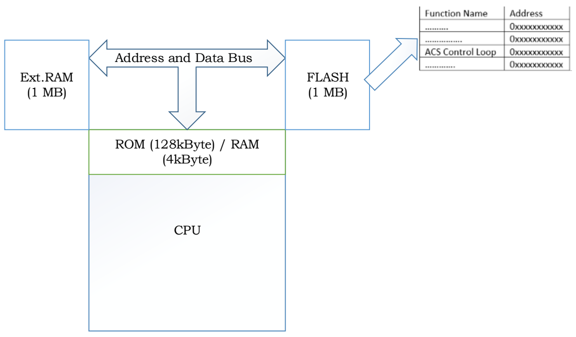
This paper describes the implementation of the method of patching the data interpreter function on the OBDH (case study is using the ACS control loop function) in order to improve the performance and keep the system working properly. The implementation of the patching process in the superloop architecture program is quite challenging because this process is performed when the satellite has been operating and is in orbit. Error sending commands to patch, delete, or overwritten certain memory addresses can cause OBDH malfunctions. Initial verification through testing on the ground have been successfully performed using engineering model of OBDH and hardware in the loop simulators (HWIL). Based on the test results, implementation on satellite has also been successfully done.

2 METHODOLOGY

In general, an overview of the OBDH LAPAN-A3 / IPB memory allocation is shown in Figure 2-1.

Almost two thirds of the memory allocation in ROM is occupied by the BIOS program, one third is left blank. Whereas all data interpretation functions, whether sensors or commands, are placed on an external flash. Each memory, whether internal or external, shares access to the address and data they have. The patching process has several steps, as illustrated in Figure 2-2.

1. Error Identification
2. Create Patch File
3. Target Patch Preparation
4. Patching Process
5. Target Patch Activation



Function Name	Address
....	0xxxxxxxxxxx
....	0xxxxxxxxxxx
ACS Control Loop	0xxxxxxxxxxx
....	0xxxxxxxxxxx

Figure 2-1: OBDH memory allocation

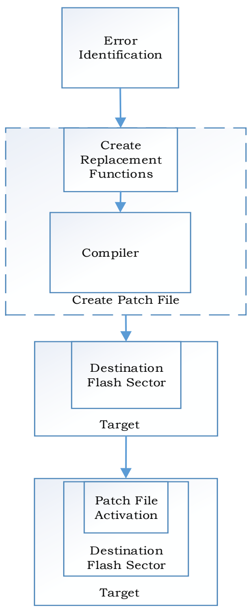


Figure 2-2: Process of patching the OBDH firmware system

2.1 Error Identification

In this step, identification and determination of errors that occur in the firmware system are accomplished. In this paper, the error that will be corrected is an error that comes from the ACS control loop command interpretation function.

2.2 Create Patch File

Code design, verification and program code compilation which aims to change the program code to the code that corresponds to the target binary file. The binary file is a patch file that will be uploaded to OBDH.

The program code design and compilation process are accomplished using the Renesas-High Performance Embedded Workshop (HEW) software.

2.3 Target Patch Preparation

The next step is check and deactivate the program functions that will be replaced. All program functions are stored at a specific address on the internal flash memory, for this purpose the process of checking and deactivating the program function is to be replaced at the function reference address. Delete the memory area on the intended flash address, to make space for new functions that have been created previously.

2.4 Patching Process

The upload process can be done in two ways, that is through commands when the satellite through the LAPAN Ground Station or can be done through scheduling. In this paper the upload process is done manually through commands to the satellite to minimize errors in the patch process. After the upload process is done then the patch file validation. At the function reference address, set a pointer for the new function that has been uploaded and then restart the OBDH system. A system restart is required to initialize all OBDH hardware and software functions**.**

2.5 Target Patch Activation

Check and verify again that the program that was uploaded was successfully executed by OBDH. Verification and initial testing are done on the ground using OBDH Model engineering and HWIL as satellite subsystem simulators.

3 RESULT AND DISCUSSION

3.1 On-Ground Testing

Before patching process is implemented on a satellite, the simulation and testing are done first on the ground. The test was conducted at the AIT (Assembly, Integration and Test) Laboratory of the LAPAN Satellite Technology Center. Testing instrumentation using engineering model of PCDH (EM PCDH) and HILS modules. The HILS module functions as a satellite subsystem simulator especially for attitude sensors and actuators including star sensors, reaction wheels, etc. This module uses data from each subsystem that has been stored in its internal memory to be used as feedback to EM PCDH and PCDH. Communication line between EM PCDH, HILS module, and PC using RS422 protocol. All action commands given by EM PCDH will then be followed up by the HILS module to produce a reaction. All data processing and settings for each module are done via a PC (Personal Computer). Figure 3-1 shows the test scheme on the ground.

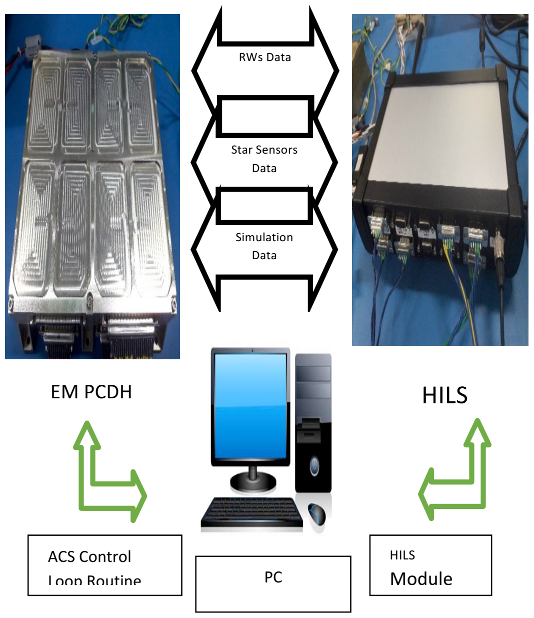


Figure 3-1: On-Ground testing scheme

The ACS Control Loop function is related to several sensors and actuators namely star sensor and reaction wheels that are simulated using the HILS module. The design and compilation of patch files is done on the PC and then uploaded and activated on the target. Observations on satellite attitude data generated by star sensors are carried out before and after patch activation on the target device. Table 3-1 shows the test results after and before the patch file is activated.

The first ten data in table 3-1 are the data from the quaternion obtained before the patch is activated, and the next ten data are then quaternion data obtained after the patch is activated. Table 3-1 shows that before the patch file is activated the quaternion value on the star sensor is the same for data sampling and the reference time for each sampling is the same which means that the quaternion value cannot be renewed due to a malfunction in the star sensor data interpretation obtained from the HILS module.

3.2 In-Orbit Implementation

Patch file implementation on satellites is done when the satellite is not in a specific operation mission, aiming to reduce the risk of failure in the system firmware. Observations were made to compare satellite attitudes data from star sensors before and after the patch file was activated as shown in table 3-2.

Tabel 3-1: ON-GROUND TESTING RESULT BEFORE AND AFTER PATCH ACTIVATION

Star Sensor attitude data in Quaternion format	
Q1; Q2; Q3; Q4	**Ref. time [s]**
0;0;1;0	0
0;0;1;0	0
0;0;1;0	0
0;0;1;0	0
0;0;1;0	0
0;0;1;0	0
0;0;1;0	0
0;0;1;0	0
0;0;1;0	0
0;0;1;0	0
-0.26571;0.64236;0.67296;0.25273	118.806
-0.26572;0.64226;0.67303;0.25280	118.814
-0.26577;0.64229;0.67300;0.25275	118.823
-0.26594;0.64246;0.67282;0.25263	118.832
-0.26559;0.64209;0.67322;0.25285	118.841
-0.26569;0.64206;0.67321;0.25865	118.849
-0.26582;0.64219;0.67307;0.25279	118.858
-0.26568;0.64206;0.67322;0.25285	118.866
-0.26568;0.64194;0.67335;0.25279	118.988
-0.26576;0.64228;0.67304;0.25268	119.005

Tabel 3-2: IN-ORBIT IMPLEMENTATION RESULT BEFORE AND AFTER PATCH ACTIVATION

Star Sensor attitude data in Quaternion format	
Q1; Q2; Q3; Q4	**Ref. time [s]**
-0.3257; -0.9447; -0.0142; 0.0332	32724
-0.3257; -0.9447; -0.0142; 0.0332	32724
-0.3257; -0.9447; -0.0142; 0.0332	32724
-0.3257; -0.9447; -0.0142; 0.0332	32724
-0.3257; -0.9447; -0.0142; 0.0332	32724
-0.3257; -0.9447; -0.0142; 0.0332	32724
-0.3257; -0.9447; -0.0142; 0.0332	32724
-0.3257; -0.9447; -0.0142; 0.0332	32724
-0.3257; -0.9447; -0.0142; 0.0332	32724
-0.3257; -0.9447; -0.0142; 0.0332	32724
-0.229001; -0.84204; -0.051492;0.48567	441383953
-0.2228986; -0.84197; -0.051523;0.485791	441384203
-0.228964; -0.841699; -0.051645;0.486263	441385203
-0.22915; -0.841455; -0.051613;0.486601	441385703
-0.229063; -0.841443; -0.051695;0.486654	441385953
-0.229071; -0.841257; -0.051738;0.486895	441386453
-0.228918; -0.841298; -0.051881;0.487169	441387203
-0.228955; -0.841173; -0.051956;0.48756	441387953
-0.229009; -0.840932; -0.051956;0.487715	441388203
-0.228867; -0.840827; -0.052115;0.487995	441388954

Same as in table 3-1, the first ten data are the data from the quaternion obtained before the patch is activated, and the next ten data are then quaternion data obtained after the patch is activated. The default system firmware on the OBDH before the patch file is activated shows the same quaternion value and reference for the star sensor. After the patch is activated, changes occur for each quaternion value and reference time for each data sampling.

4 CONCLUSION

Testing result and implementation in orbit indicate that the patching method used is able to overcome errors that occur in the ACS control loop function on OBDH. This method can also be used to update the system firmware of OBDH. The use of this method requires caution due to errors in function changes, certain memory deletions, and overwriting of functions with other functions can cause PCDH may have unpredictable consequences and may result in a malfunctioning device.

ACKNOWLEDGMENT

The authors would like to thank Director of Satellite Technology Center (LAPAN) Mr. Ir. Mujtahid, MT and all members of LAPAN-A3/IPB AIT and Ground Station team for their support so that this research can be successfully completed. Also a regards for redaction team of Jurnal Teknologi Dirgantara LAPAN.

REFERENCES

Hasbi W., & Suhermanto., 2013. *Development of LAPAN-A3/IPB Satellite an Experimental Remote Sensing*

Microsatellite 34th Asian Conf. Remote Sens. 2013, ACRS 2013, p. 1508-1515.

Hardhienata S., & Triharjanto RH., 2007. *LAPAN-TUBSAT: From Concept to Early Operation*. Lembaga Penerbangan dan Antariksa Nasional, 2007.

Hasbi W., & Karim A., 2013. *Lapan-A2 System Design for Equatorial Survaillance Missions*. In 9th International Symposium of The International Academy of Astronautics (IAA) Berlin, (pp. 8-12).

Hasbi W., et al., 2016. *LAPAN-A3/IPB Microsatellite for Remote Sensing Experiment Detail Design*. Laporan Teknis Kegitan Penelitian. Pusat Teknologi Satelit. Bogor

Ekman M., & Thane H., 2007. *Dynamic patching of embedded software*. In Real Time and Embedded Technology and Applications Symposium, IEEE. pp. 337-346

Calder A., & Kutt P., 2009. *Flight Software Design Guidelines for Enhancing On-Orbit Maintenance*. InAIAA Infotech@ Aerospace Conference and AIAA Unmanned... Unlimited Conference. p. 1818

Hayden CM., Smith EK., Denchev M., Hicks M., Foster JS., 2012. *Kitsune: Efficient, general-purpose dynamic software updating for C*. InACM SIGPLAN., Vol. 47, No. 10, pp. 249-264.

Kraft J., Kienle HM., Nolte T., Crnkovic I., Hansson H., 2011. *Software maintenance research in the PROGRESS project for predictable embedded software systems*. InSoftware Maintenance and Reengineering (CSMR), 2011 15th European Conference on 2011 Mar 1 (pp. 335-338). IEEE.

Savitha A., Chetwani RR., Bhanumathy YR., Ravindra M., 2017. *Model based system for software change analysis for embedded systems on spacecraft*. InAdvance Computing Conference (IACC) (pp. 418-422). IEEE.

Trümper J., Voigt S., Döllner J., 2012. *Maintenance of embedded systems: Supporting program comprehension using dynamic analysis*. InSoftware Engineering for Embedded Systems (SEES), 2012 2nd International Workshop (pp. 58-64). IEEE.

Pingale P., Amrutkar K., Kulkarni S., 2016. *Design aspects for upgrading firmware of a resource constrained device in the field*. InRecent Trends in Electronics, Information & Communication Technology (RTEICT), IEEE International Conference (pp. 903-907). IEEE.