FalconSAT-3 Software Extension For Thruster Performance Analysis

Adrianna Eaton and Justin Landseadel
Cadets First Class, United States Air Force Academy, USAFA, CO, 80840

Steven Hart
Major, USAF Reserve, AFRL/RVE, Kirtland AFB, NM, 87117

This paper describes the USAFA design process for upgrading FalconSAT-3 (FS3) on-orbit software from concept through on-orbit data acquisition. FS3 was launched in March 2007 with the goal of meeting Department of Defense science objectives with three payloads: Plasma Local Anomalous Noise Environment (PLANE), Flat, Local, Ambient, Plasma, Spectrometer (FLAPS), and Micro Pulsed Plasma Attitude Control System (MPACS). The respective goals of these payloads were to investigate ambient plasma, plasma morphology, and generate plasma while establishing flight heritage on the MPACS thruster technology for the Air Force Research Lab (AFRL). The AFRL designed MPACS have been demonstrated through on-orbit tests which included short duration checkout, performance tests, and long duration firings. With initial objectives achieved, FS3 retains useful battery life and MPACS fuel. AFRL customers have considered these factors and redirected efforts toward analyzing long term spacecraft bus impacts from thruster firings. After initial investigation, it was determined that while the hardware capability exists, flight software did not provide adequate data capture rates, nor available storage, to allow the detailed thruster impact analysis to proceed. The software requiring update is the primary housekeeping task (PHT). PHT is responsible for managing all analog conversions and is also the primary software interface to ground operations. FS3 is currently operated by 3 geographically separated ground stations: USAFA, Air Force Institute of Technology, and the latest at the USMA at West Point. Multiple ground stations, users, processes, and documentation have required the flight software update create no impact to ground operation while providing the enhanced capability that AFRL customers have requested.

I. Introduction

The software extension for thruster performance analysis began as an Air Force Research Laboratory sponsored cadet research project in the summer of 2010. FalconSAT-3 (FS3) was launched in 8 March 2007 and has completed more than 4 years of spaceflight experimentation and USAFA cadet led operations. The Micro-PPTs Attitude Control System (MPACS) is an Air Force Research Laboratory (AFRL) sponsored experiment that includes 4 modules each consisting of 3 micro pulsed plasma thrusters (PPT). This system is shown in Fig. 1 below.

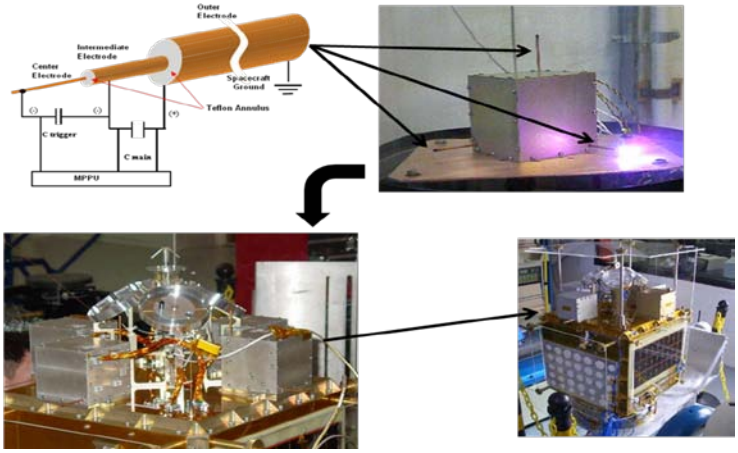

Figure 1. Micro PPT's integrated to MPACS modules and FS3

The summer research purpose was to investigate, collect, and quantify the PPT test firing results and also assess what potential spacecraft system impacts may have occurred due to the PPT firings. This consisted of three primary objectives:

1) Determine if MPACS PPT main and trigger capacitor voltages change over lifetime
2) Compare PPT firing data to each other, noting variation
3) Determine if MPACS operation affected spacecraft systems
    a. Solar Arrays (contamination)
    b. Battery (voltage degradation)
    c. Communication (degradation in capability)

In attempting to satisfy these objectives, it was determined that the data available from prior MPACS firings was insufficient due to limited analog sampling rates. This finding led not only to new experiments being planned for the 2010-2011 academic year to satisfy AFRL customer requirements, but a need for upgraded spacecraft software to successfully execute the experiments. The primary experiment underway is called the MPACS high data rate-short duration test campaign. This new MPACS experiment will involve 5 minute PPT firings while in view to the USAFA ground station and collecting the PPT firing telemetry channels at a faster sampling rate than previously used. FalconSAT-3 flight software manages the MPACS payload via the Primary Housekeeping Task Executable (PHTX) software. In order to accomplish the higher data collection during the PPT test firing, the PHTX software needed to be modified to enable the faster sampling rate. Previous MPACS tests were conducted with a once per 5-second data sampling rate (0.2 Hz). However, MPACS PPT firings occur twice every second, a 2 Hz sampling rate, leading to under-sampled data sets to best capture the actual internal details of the PPT firing event. With the AFRL customer desiring great insight into the PPT firings, the FS3 flight software needed to be modified to collect the data at a customer threshold minimum 1 Hz sampling rate. AFRL scientists and engineers sought to gain detailed insight into the life cycle of the rapid PPT firings and also be able to assess effects of the PPT operations on the spacecraft bus systems. Emphasis was placed on assessing the PPTs affects on solar array effectiveness, battery life and the downlink communications link. This paper details the processes involved in the software upgrade and extension to support the additional spaceflight experimentation for the thruster firings and analysis. The going in assumption regarding this software extension was that there are no changes in any interfaces as well as to crew force operational procedures. Thus, the effort was to be seamless and transparent in the eyes of the FS3 operators and engineers maintaining the ground systems multiple ground stations.

II. Software Development Process

The design philosophy behind the FalconSAT program at USAFA is to procure and maintain a non-flight set of hardware that is nearly identical to the on-orbit hardware. This Flat-Sat hardware was used pre-launch as the sole mechanism for testing flight software. The Flat-Sat hardware is flight-like in every way, including RF data link, peripherals, and operating system. The pre-launch software design and integration flow included only the flat-sat and flight model hardware shown in Fig. 2. These hardware platforms imposed limitations that inhibited rapid prototyping of flight software due to limited debugging capability and slow turn time from compilation through test.

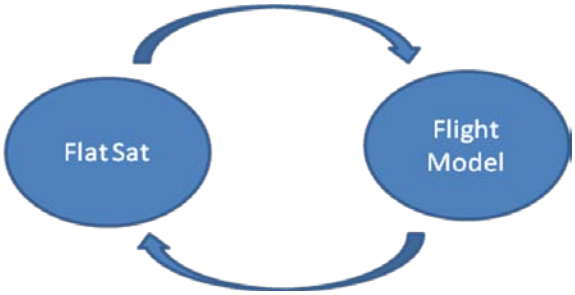

Figure 2. Pre-launch software design and integration flow.

The time and debug limitations led the cadet team to investigate alternate means of software development to expedite the integration of enhanced software. The desired end state was to place a simulation capability (Fig. 3) in the design loop to facilitate "quick turn" development and test, as well as a more full-featured debugging capability to include Statement Testing[1].

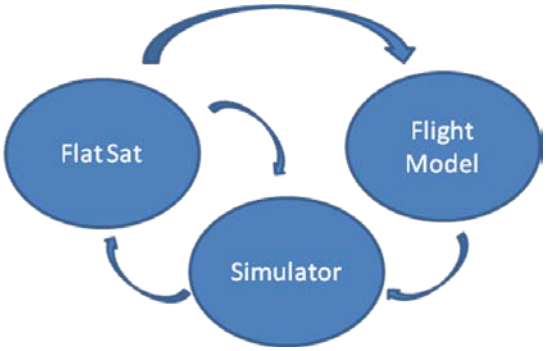

Figure 3. Updated software design and integration flow.

Cadets assigned to the FalconSAT-3 capstone project undertook a trade study to examine options for compliers, development environments, and best-practices. The trade study led cadets to a GNU based open-source solution as a test platform. The Eclipse based Integrated Development Environment (IDE) and the CygWin gcc compiler running under Windows offered the ideal platform for achieving the quick turn time and software debug capability required. The study yielded a framework to perform best practices including reviews and inspection[2] with static analysis tools such as LINT[3].

The operating system running on FalconSAT-3 is the Commercially-Off-The-Shelf (COTS) "SpaceCraft Operating System" (SCOS), written by BekTek Inc. The Primary Housekeeping Task (PHTX) is one of several running tasks on the processor, with interfaces to multiple other pieces of software. Creating a new, simulated, version of PHTX (called PHTX2) was the next challenge to be overcome. After identification of the development environment, the next challenge was to effectively simulate the spacecraft operating system in an which does not support the majority of the SCOS functionality that is targeted for embedded 8086 processors.

With the simulation tools defined, initial development for PHTX2 occurred on a PC. The C code for the original PHTX was modified with conditional compilation and compiled for the Eclipse IDE. Because PHTX was targeted to run on FalconSAT-3, which has far different hardware and software resources, much of the functionality of PHTX would not work on PC. To work around this limitation, multiple stub functions were made that matched the prototypes of functions available to PHTX on FalconSAT-3. These functions could not perform the same actions as the original functions, but they were modified to produce an output to standard output or stimulus back into the PHTX under test. Using these stub functions, the original version of PHTX was compiled for a desktop computer and used as a baseline configuration for simulation and a starting point for PHTX2.

At this point in the project, two versions of the PHTX software had been created and configuration managed, PHTX and PHTX2. PHTX, when cross compiled for the PC is referred to as PHTXSIM. PHTXSIM, when run, was able to accept commands and produce output to the standard output. PHTXSIM thereby formed the basis for 'black box' testing of the PHTX software on a PC. With a varied stimulus, through command and time, PHTXSIM would produce a variety of output to the standard output, which was redirected to a file for data logging. Achieving

identical performance from PHTX2 was then a matter of modifying PHTX2 until the software produced identical output from the initial test stimuli.

Through several weeks of iteration and design in simulation, PHTX2 reached a point where the simulated output files matched PHTXSIM, which indicated it was ready for deployment in the FlatSat hardware.

At this phase of the program, PHTX2 passed from engineering to operations, and cadet operators took lead on Flat Sat testing. PHTX2 was compared to the original PHTX to make sure that they were outputting the same data under the same conditions. This was accomplished using the cadet operator test procedure used with the original flight model. The Flat-Sat model was exercised with both PHTX and PHTX2 to create files, manipulate hardware, and collect data. Both versions of the software were subjected to a battery of tests on the Flat Sat to fully test and compare their functionality. One hardware change to the Flat Sat allowed the injection of a 0.1Hz sinusoidal wave into an MPACS Analog to Digital Converter (ADC) channel to perform verification of the sampling rate upgrades. Several incompatibilities were discovered and resolved when migrating from the simulated PHTX2 to the Flat Sat. These minor changes included accessing far memory and other compiler/hardware specific changes that were repaired with conditional compilation. After it was established that PHTX2 was producing identical outputs as the original PHTX, the new high data sampling rate capability was tested using the injected MPACs signal. After this testing is completed, PHTX2 will be ready to be uploaded to FalconSAT-3 on orbit..

III. FlatSat Test Results

The testing for PHTX2 was accomplished on the Flat-Sat shown below.

Figure 4. Flat-Sat hardware

The original PHTX along with all the flight software currently on FS3 was uploaded onto the Flat-Sat. Then, the signal shown in Fig. 5 was injected into MPACS module 1 (on the –Y face of the spacecraft). This injected signal has amplitude of 700 mV peak to peak and a frequency of 0.1 Hz.

Figure 5. Injected signal for testing on Flat-Sat

The WOD collecting was enabled and the MPACS module 1 was enabled and fired. The data from this firing was plotted and is shown below.

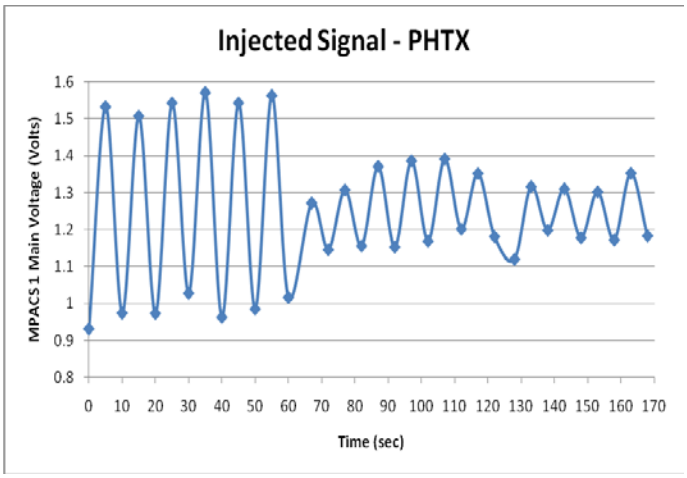

Figure 5. Recreated signal from PHTX data.

Figure 5 clearly shows the need for the updated software that will provide a faster sampling rate. The sinusoid does not resemble the injected signal after 70 seconds into the test because the sampling frequency is higher than the Nyquist frequency. This causes the injected to signal to appear to have different characteristics (different amplitude). The signal cannot be properly reconstructed given the available data because there are not enough data points. To solve this problem, PHTX2 was developed to sample at lower frequencies, giving more data points to reconstruct the signal. Sampling at a lower frequency, 1 Hz, gives the results shown in Fig. 6.

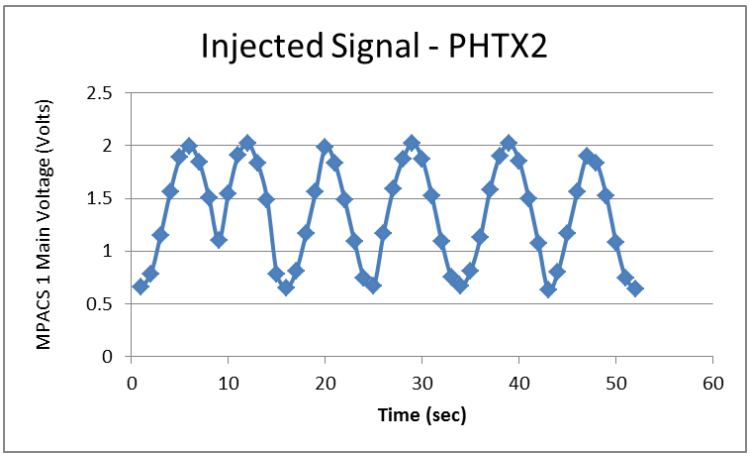

Figure 6. Recreated signal using PHTX2 data.

IV. Further Development Plans

Throughout this development process, we have learned the importance of interfaces in software engineering. The PHTX software controls FS3 commands and telemetry. Within the spacecraft, PHTX interfaces with other software and hardware on the satellite such as the power distribution board, sun sensors, ADCS, the payloads, file system, and the receiver and transmitter. External to the FS3 spacecraft, PHTX interfaces with the ground station software through a program called COMFS3 that receives all the telemetry and executes commands to the satellite. PHTX also writes a spacecraft telemetry file that captures more than 128 data points throughout the spacecraft. These are whole orbit data (WOD) files that must be executed by software on the ground. The ground station software and hardware then determines the procedures that are used for operating the satellite, the training manuals for the operators, and all of the data analysis tools used to process the collected data. This architecture can be seen in the Fig. 6 below.

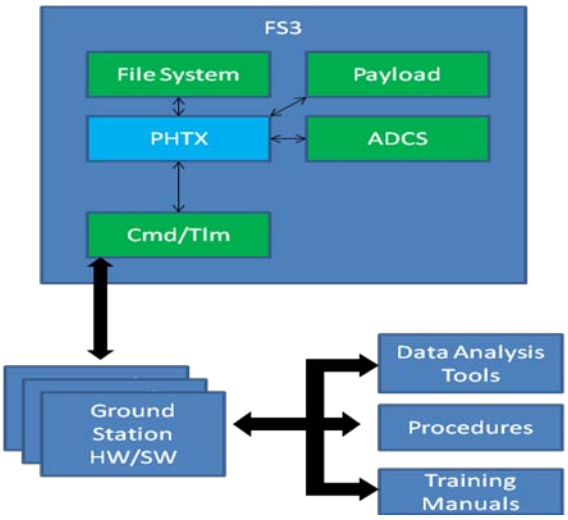

Figure 7. Software/hardware architecture for FS3.

Based on this software architecture, PHTX is just one piece of a highly integrated system. Because of this, PHTX2 is required to interface with the other satellite systems the same so nothing else beyond PHTX needs to be modified. If an interface with any of the other systems of FalconSAT-3 is changed, that system would need to be changed as well. Changing any interface to PHTX will drastically change the current operations of the satellite and would require changes to other systems as well as retraining for all of the operators at all three ground stations and modified hardware/software at those ground stations. PHTX2 was designed so that, to an operator, PHTX2 will look the exact same as PHTX. The only differences between the original PHTX and PHTX2 is the improved code structure for ease of modification and a higher data sampling rate capability to enable the high data rate MPACS experiment to meet the AFRL requirements and obtain the MPACS data.

With PHTX2 tested and validated, the next step is to upload and replace the existing PHTX with PHTX2. After a trial period of normal operations to further ensure no adverse effect on the spacecraft or ground system, a PPT will be fired and the improved PHTX2 capabilities verified. Once that is accomplished, up to 20 5-minute PPT test firings will be completed for the MPACS high data rate experiment. These test firings occur within a 10-minute pass over the USAFA ground station. The sister ground stations at AFIT and West Point will take part in later PPT test campaigns and joint the USAFA cadet satellite operators in executing agile, responsive and very high tempo spaceflight experimentation operations.

V. Conclusion

The FalconSAT-3 software extension was a necessary development to provide the AFRL customer an internal view of the MPACS payload. The software had to be completely rewritten as opposed to just modified because the original PHTX software was fragile. It had been operating on the satellite for three years and had been modified several times with little documentation. As a result, making more modifications would have done more harm and taken longer than rewriting the software.

The most important lesson learned from this software extension is the importance of interfaces. It is relatively easy to rewrite software to do what is required but extremely difficult to rewrite software to do what is required and look and operate the same to all of the satellite systems, ground stations, and operators. Because of this, the software extension has been an integral part of learning the true, iterative nature of engineering work through this development process.

As a result of the software extension, the cadets were able to place a simulation capability in the design loop to facilitate quicker development and testing for future undertakings with FS3's software development, giving future cadets a more full-featured debugging capability. Cadets are also now able to perform the experiments they envisioned for FS3 at the beginning of the academic year.

Acknowledgments

The authors thank Col. Jack Anthony for all of his help, contributions, support, and for being awesome.

References

Books

[1]Roper, Marc, *Software Testing*, McGraw Hill Book Company Europe, Berkshire, England, UK, 1994, Chap. 3.

Electronic Publications

[2]Chillarege, Ram., "Software Testing Best Practices," *IBM Research – Technical Report RC 21457 Log 96856 4/26/99*

Computer Software

[3]LINT, PC Lint for C/C++, Gimpel Software, Ver 8.0, Norristown, PA, 19403, USA