Proximity Operations Nano-Satellite Flight Demonstration (PONSFD) Rendezvous Proximity Operations Design and Trade Study Results

Jacob D. Griesbach, Jason J. Westphal, Christopher W. T. Roscoe, Joseph A. Smugeresky, Dean R. Hawes, John P. Carrico, Jr.
Applied Defense Solutions, Columbia, Maryland

ABSTRACT

A pair of duplicate 3U CubeSats are planned for a Fall 2015 launch to demonstrate rendezvous, proximity operations, (RPO) and docking in low Earth orbit (LEO). Each CubeSat contains a processing subsystem dedicated to RPO guidance, navigation, and control (GNC). A mission overview is presented as well as a detailed functional explanation of the RPO GNC subsystem. The evolution and design of the flight specific software architecture, which addresses key mission challenges and permits flexibility in algorithm development as the program progresses, is described in detail. This includes the software implementation of real-time and non-real-time processing threads and inter-process messaging. A high level overview of primary subsystem capabilities of navigation and guidance is also presented.

1. INTRODUCTION

The Proximity Operations Nano-Satellite Flight Demonstration (PONSFD) program is designed to demonstrate rendezvous, proximity operations (RPO), formation flying, and docking with a pair of 3U CubeSats. The program is sponsored by the NASA Ames Research Center via the Office of the Chief Technologist in support of its Small Spacecraft Technology Program (SSTP).

The goal of the mission is to demonstrate complex RPO and docking operations with a pair of low-cost 3U CubeSat satellites using passive navigation sensors. These operations have been demonstrated by past missions, but they required much larger, more powerful, and more expensive spacecraft. Developing low-cost algorithms and technology to accomplish complex mission goals using miniaturized spacecraft and components is a principal goal of the NASA SSTP. The program encompasses the entire mission evolution including system design, acquisition, satellite construction, test, launch, mission operations, and final disposal. The satellites are scheduled for launch in Fall 2015 with a 1-year mission lifetime. This paper provides a brief mission overview but will then focus on the software architecture design and subsystem capabilities of the RPO guidance, navigation, and control (RPOGNC) mission specific processor.

The current overall design utilizes a distributed architecture with multiple on-board processors, each specifically tasked with providing mission critical capabilities. These capabilities range from attitude determination and control to image processing. The RPOGNC processor is responsible for absolute and relative navigation, maneuver planning, attitude commanding, and abort monitoring for mission safety. A low power processor running a Linux operating system has been selected for implementation.

The current overall design leverages Tyvak Nano-Satellite Systems LLC's [1] heritage Intrepid software for internal communication and messaging. Other published flight software designs were also reviewed [2–9]. Use of NASA/Goddard cFE/CFS flight software [2, 3] was briefly considered, but was eliminated from consideration due to concerns of availability, accessibility, and compatibility with the chosen hardware. Dwyer's design patterns for CanX [4] were studied for its use of multiple parallel threads. The SNOE architecture [5] was reviewed, which contains published designs for CubeSat-typical sensor acquisition and data handling with associated command and telemetry message communication. However, due to the PONSFD requirements for autonomy and support for non-real-time processing (as discussed in section 4), a more complex and capable architecture/framework is needed. Lastly, lessons learned from previous flight software design pattern principles [6–9] were applied by designing desired UML description and flow diagrams prior to code development.

The RPO GNC subsystem must provide two major capabilities: navigation and guidance. Navigation consists of keeping track of the satellite's own orbital state (position and velocity), as well as, when possible, the other satellite's state. Furthermore, the navigation component must also track the satellite's attitude to be able to react to and understand the information provided from on-board, optical sensor observations and accurately model atmospheric drag effects.

Guidance consists of providing the ability to autonomously and efficiently calculate maneuvers to achieve a desired future state or trajectory. For example, the satellite may be commanded to "initiate an in-plane circumnavigation relative orbit of 500m x 1000m at time X."

2. MISSION OVERVIEW

The PONSFD mission consists of several demonstration phases. After launch, the CubeSats will be deployed from the launch vehicle simultaneously, likely from a 6U P-POD [10] dispenser. Although the actual orbit for the mission is unknown at present, for analysis purposes, a reference 425km, circular, sun-synchronous orbit with a 10:30am descending node time is being used. Initially, the two satellites will be rigidly fixed to each other in a side-by-side configuration. The satellites will be deployed in a conjoined configuration to avoid large separations that would occur simply from natural orbital drift associated with separate deployment. After the satellites have passed initial checkout, the physical mechanism binding the satellites together will be broken, allowing the CubeSats to drift independently. A short final checkout period will follow after the release occurs.

Once nominal satellite operations have been verified, a walking safety ellipse [11–13] will be instantiated to maneuver the satellites to a nominal proximity distance. The walking safety ellipse provides operational collision safety during approach. Once the optical sensors and the inter-satellite communication link (ISL) have been tested and calibrated, precision formation flying and relative maneuvering will be demonstrated. Planned formations and maneuvers include a simple leader/follower formation, a natural motion circumnavigation formation, an active forced-motion waypoint traversing path, and active forced-motion non-passively safe R-Bar stationkeeping.

The flight software is intended to be robust to dynamic CONOPS. This is done by keeping the command set autonomous, generic, and agnostic to specific maneuver and mission ordering, while providing a rich selection of spacecraft modes to be able to perform and chain most conceivable maneuvers and formations. This is discussed in more detail in section 3.

Since both spacecraft are maneuverable, identically constructed, and run identical software, the roles of chief and deputy are designed and planned to be interchanged to normalize fuel usage between the two spacecraft. Furthermore, during RPO maneuvering, both spacecraft are planned to be actively tracking each other optically. This implies that both CubeSats will be actively controlling their attitude to maintain their sensor boresight on the opposing spacecraft. This allows for two benefits: 1) each spacecraft can collect relative sensing observational data that can assist with proper maintenance of RPO operations, and 2) both satellites maintain a nearly neutral differential drag due to their complimentary attitude angles relative to the velocity vector.

When RPO demonstration is complete and sensor accuracy has been verified, the CubeSats will approach along the V-Bar and physically dock using a custom docking mechanism. Once undocked, the spacecraft are available for additional formation flight and docking demonstrations. For example, once a successful V-Bar dock has been demonstrated, an R-Bar dock may be considered for demonstration.

Once the mission is complete, the spacecraft are designed to have additional propellant available for activities that are currently unplanned. After this period, both CubeSats will be actively and safely de-orbited.

3. FUNCTIONAL VIEW

The RPO-specific flight software is designed to be a modular component of the overall spacecraft. It is intended to be run on an independent System-On-Module (SOM) that is basically its own Linux computer, where data is shared between other SOMs with telemetry and command messages passed via a standard serial interface. The SOM related to RPO operations is referred to as the Rendezvous Proximity Operations Guidance, Navigation, and Control (RPOGNC) processor.

A key part of the RPO specific software is relative navigation, which relies on monitoring the relative position of the opposing spacecraft. While the RPO software is agnostic to how these observations are collected, another SOM shall process imagery collected from one or more optical sensors to provide such observations. Absolute measurements of

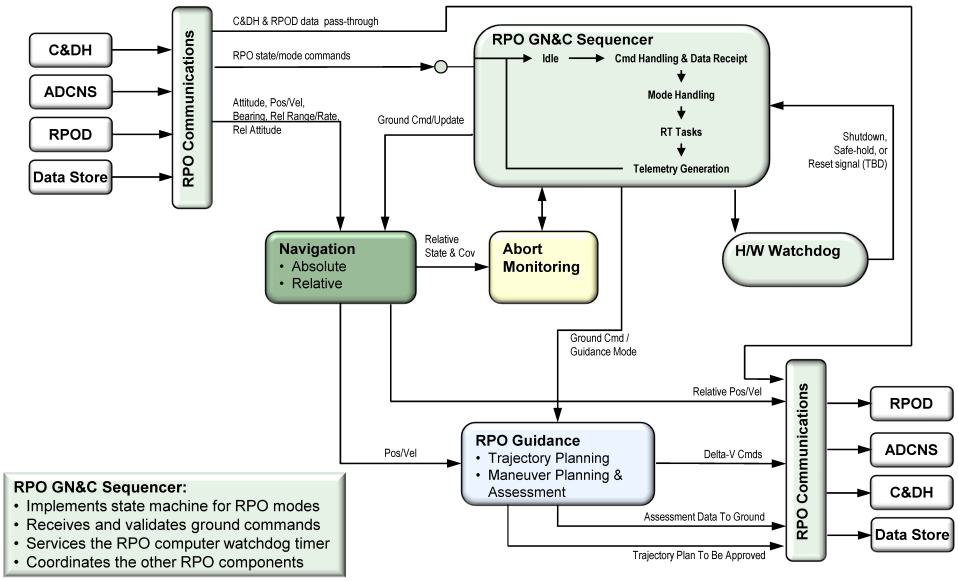

Fig. 1. PONSFD RPOGNC Processor Functional Diagram

the spacecraft's state (position and velocity) will be obtained from an onboard GPS unit. The RPOGNC processor is designed to communicate via standard ground communications and via an optional inter-satellite communication cross-link (ISL).

Fig. 1 illustrates the functional concepts provided by the RPOGNC processor/subsystem. At the heart of the diagram is the RPOGNC Sequencer. It is responsible for running the sequence of processing events that occur on each executive cycle. The two other main functional components are the Navigation and Guidance modules, which are described in section 5. The remaining elements of the functional diagram include the watchdog and abort monitoring components, as well as the underlying infrastructure software (discussed in section 4) that manages processes and enables internal and external messaging communication.

3.1 GNC Sequencer

The sequencer is responsible for interpreting command messages and managing the RPOGNC subsystem's modes and states that are shown in Fig. 2. The RPOGNC subsystem state is one of two options: proximity operations or hybrid. In the proximity operations state, the system is able to autonomously calculate and execute maneuvers. In the hybrid state, the system will still calculate maneuvers, but in this state, ground operations must approve all calculated maneuvers before they are executed.

The RPOGNC modes for each spacecraft are listed on the left hand side of Fig. 2, where the modes identify the type of maneuver the spacecraft is currently executing and/or targeting. It should be noted that, while all modes are available in the proximity operations state, only certain modes are available for use in the hybrid state. These are illustrated in the figure with a dark green background. An activity of particular interest is the abort activity, where a detected anomaly triggers either a hard or soft abort scenario. A soft abort consists of canceling all current and/or planned maneuvers and revert to standby mode, while a hard abort executes a simplified emergency escape maneuver to retreat to a safer orbital trajectory.

The sequence of events executed by the GNC Sequencer is shown in Fig. 3. Here, a continuous loop is executed that monitors processes and messages, where external inter-processor communication (IPC) messages are sent and received by a library that is part of Tyvak's heritage design. When a periodic timed event triggers, notionally every second, the GNC sequencer performs a list of processing events, as illustrated in the red block of Fig. 3. These activities include updating navigation estimates, performing stationkeeping (closed-loop maneuver control), maneuver scheduling, and

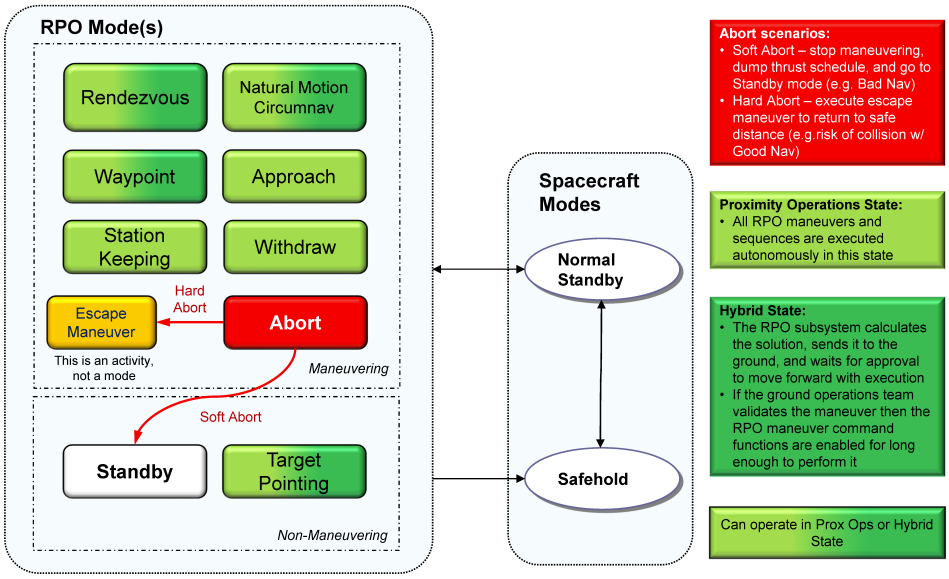

Fig. 2. PONSFD RPOGNC Modes & States

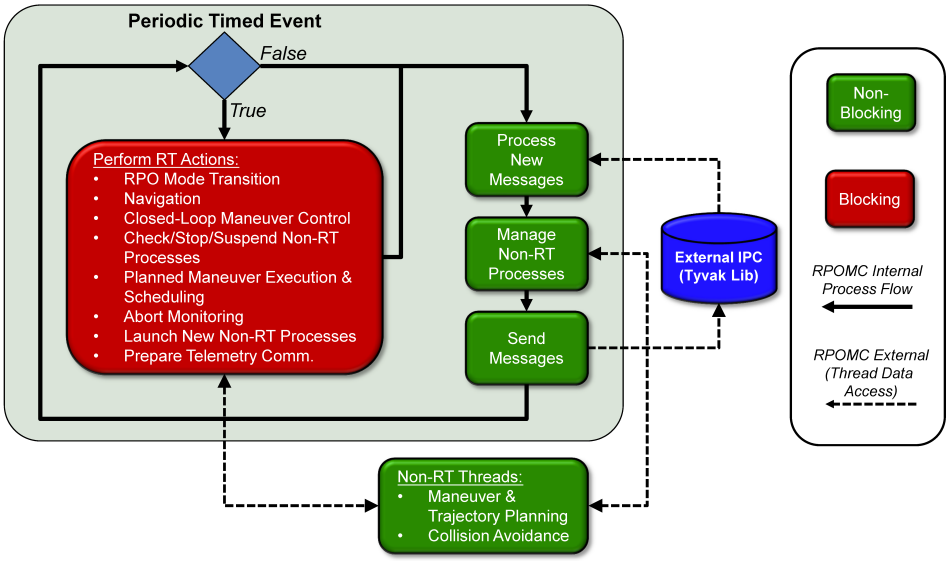

Fig. 3. PONSFD RPOGNC Sequencer

abort monitoring.

3.2 Blocking

In Fig. 3, modules that are colored green are "non-blocking" software blocks that do not halt or suspend execution if there is no action to be taken. For example, the Process New Messages task does not wait for new messages to arrive. If no messages are waiting in the message queue, process control is immediately passed to the next task (Manage Non-RT Processes).

The only task that blocks by design are the real-time actions that are performed in response to a periodic timed event trigger. Since these operations are critical for operation, and they must be processed in order, each action blocks execution of the following task until the current task completes. It should be stressed though that the real-time tasks

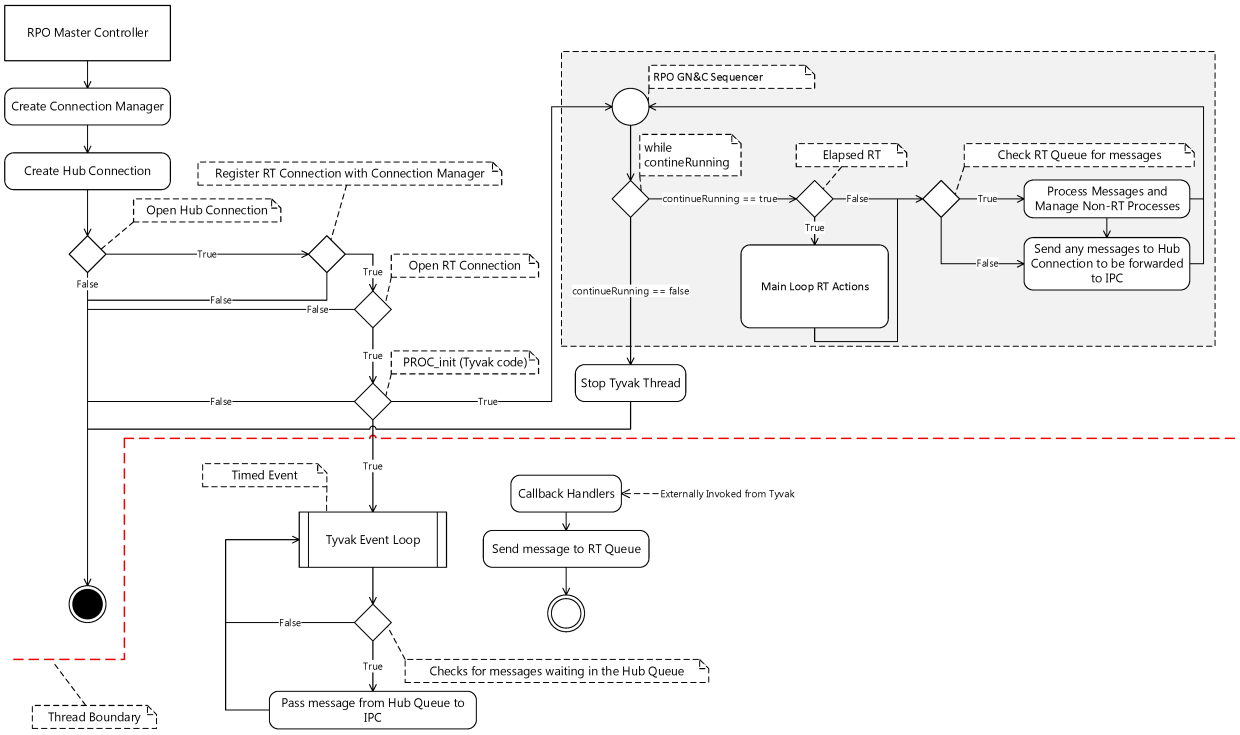

Fig. 4. PONSFD RPOGNC Sequencer UML Flow Diagram

are meant and intended to be quickly executed, such that they do not block operations for significant amounts of time.

3.3 Watchdog & Abort Monitoring

A watchdog exists to make sure the RPOGNC processor is executing as expected. Its role is to reply to status request messages sent by the Command & Data Handling (C&DH) processor. If status messages are not sent in a timely manner, then the C&DH processor may choose to power cycle the RPOGNC processor after a warning message is sent.

The role of abort monitoring is to ensure that the spacecraft is not on a potentially dangerous course that could result in a future collision. Its role is minimized when the spacecraft is in its walking safety ellipse pattern, since the orbit is designed to be passively safe. It is also minimized when the spacecraft is approaching the opposing CubeSat for docking. In other modes, especially when both spacecraft are in the same orbital plane, abort monitoring performs a simple collision avoidance analysis to determine a future probability of collision. If the probability exceeds a threshold, the module may issue a command message to the GNC Sequencer to trigger an abort condition.

4. SOFTWARE ARCHITECTURE

A number of trade studies were performed to analyze and properly move forward with an appropriate design pattern for the RPOGNC software architecture. This section highlights some of the more prominent trade study results and analyses.

4.1 Non-Real-Time Processes

It was identified early on that the RPOGNC processor must be able to execute processes in both real-time and non-real-time. Real-time processes are necessary to routinely process messages, maintain navigational awareness, etc.

These processes are expected to be run synchronously at standard periodic intervals. Furthermore, they are expected to execute in a short amount of time compared to the GNC Sequencer execution period.

However, non-real-time processes with the potential to be computationally expensive were identified such that they could not be reliably included in the standard real-time cycle without severely affecting the cycle's run-time period. In particular, the software processes of concern were the maneuver planning guidance and the collision avoidance assessment tasks. Both tasks require iterative convergence, such that the number of iterative cycles (and hence overall runtime) is not known *a priori*. It is envisioned that these processes could entail forward-looking propagations that are fairly lengthy, and as a result, could require runtimes that would be prohibitive within the standard GNC Sequencer execution loop.

As such, an architecture was constructed where computationally intensive processes could be forked to run in the background with lower preemption precedence. With this type of framework, real-time processes would run as expected, and background non-real-time processes would primarily only run as processing time becomes available.

The UML diagram that evolved conceptually from hosting non-real-time processes in parallel to the real-time GNC execution loop is illustrated in Fig. 4. The upper-left side of the diagram shows process initialization. The main GNC Sequencer loop that was examined in Fig. 3 and section 3.1 is highlighted with the dashed box in the upper-right side of the diagram. Finally, the thread that handles the messaging interface to the other subsystems via the Tyvak software library is illustrated on the bottom portion of the figure.

4.2 Soft-Real-Time vs. Hard-Real-Time

As mentioned above, the RPOGNC software hosts a Linux kernel and distribution as its underlying operating environment. While the standard Linux kernel was not designed for real-time use, several open-source kernel extensions have been provided and are available to enable the principles of real-time execution. Extensions are available for both soft- and hard-real-time execution.

"Hard-real-time" execution states that processes *must* complete in a fixed or standard amount of time, and that missing a deadline is a total system failure [14]. This implies that real-time processes cannot be interrupted, even by standard low-level system interrupts. "Soft-real-time" execution allows processes to be interrupted, and thus relaxes the requirement that all real-time processes must complete in a pre-specified, known amount of time. However, it is realized that for soft-real-time execution, the usefulness of a result degrades after its deadline, and hence degrades the system's quality of service [14].

While hard-real-time execution seems advantageous at first, it comes at the expense of stability in the Linux operating environment. Because hard-real-time execution prevents kernel-level interrupts from occurring, standard kernel housekeeping activities can be delayed or dropped, which can cause disk and memory to inherently de-synchronize. This leads to unexpected kernel panics, kernel dumps, and requirements to reboot. Since stability must be placed at utmost importance in a space environment, a soft-real-time environment was planned for PONSFD use.

With the soft-real-time requirement, all developed software can run as standard C++ userland processes. No kernel level or embedded software development has been identified or is planned. Instead, the RPOGNC software can make use of the kernel's native ability to preempt and fork processes, with standard Linux priority assignments. As such, the RPOGNC software is not strictly hard-real-time, as it can be preempted. However, soft-real-time functionality for certain software modules is provided by assigning such tasks high-priority / high-preemption precedence, such that only important kernel-level interrupts shall preempt the real-time software modules. Herein, the phrase "real-time" refers to this "soft" real-time environment.

4.3 Process Management

A complicating factor of this design is that control and messaging between the various processes must now be managed. This is performed by integrating process management and message polling into the primary execution loop, as shown in Fig. 3. Process management is the control that is needed to spawn, monitor, harvest, and kill non-real-time processes.

In this software design, the "main" function is the GNC Sequencer as illustrated in Fig. 3, where as commanded, it will fork non-real-time processes to execute tasks to compute future maneuvers to achieve a desired orbital trajectory or compute a collision assessment. The GNC Sequencer monitors the non-real-time processes that it spawns to make sure they are executing as expected and to harvest their results when they complete. If an anomaly occurs, it can choose to kill and respawn a new process.

4.4 Internal & External Messaging

The GNC Sequencer also maintains another thread that routinely checks a message queue that is populated when IPC messages arrive. This message queue thread is referred to as the Message Hub in Fig. 4, since it provides central queues for incoming and outgoing inter-process messaging communication. Process that are spawned on the RPOGNC processor can access the hub to receive messages (commands/parameters/start/stop/etc.) or send messages (status/results/etc.).

The Message Hub also links to the Tyvak Shared Message Library to send and receive messages to/from destinations external to the RPOGNC. These messages are defined in the PONSFD Interface Control Document (ICD), and contains messages such as the following:

- Receive relative observational data estimates from a sensor
- Receive absolute state data estimates from a GPS unit
- Receive commands messages to start/stop RPO operations
- Receive command messages to perform a new RPO operation
- Send RPOGNC status
- Send current estimated absolute or relative states
- Send/Receive ground communications

In this way, the Message Hub provides a single, central point for all RPOGNC processes to exchange data. Each message is stamped with a header that provides the message type, message destination, and message pedigree (history).

4.5 Power Management

CubeSats are generally power constrained, so it is intended that the RPOGNC module only be utilized during RPO operations. With this intent, the RPOGNC module is capable of receiving command messages to enter low-power and sleep processor states. A low-power state may be realized, where the processor is only used to maintain relative navigation, i.e. no maneuver planning. In this processor state the processor can be forced to idle in between GNC Sequence loops.

The processor can also be commanded to sleep during periods where no RPO operations are occurring. In this mode, the RPOGNC system memory is still powered to allow for a quick wake, similar to an ACPI S3 state [15]. When the processor is sleeping, it may be awakened by either a predefined time (alarm) or a wake-message.

Lastly, of course, the RPOGNC module may be powered off completely, similar to an ACPI G3 state [15], where the physical power to the SOM module is cut. For each of the above states, messages are planned to command and/or warn the RPOGNC processor before the state is meant to occur. With these messages, the RPOGNC SOM can prepare itself for a clean shutdown.

5. KEY CAPABILITIES

5.1 Navigation

The Navigation component is responsible for tracking the absolute state (position & velocity) solutions generated by the GPS component during RPO maneuvers, and as such information becomes available (via the ground or the ISL),

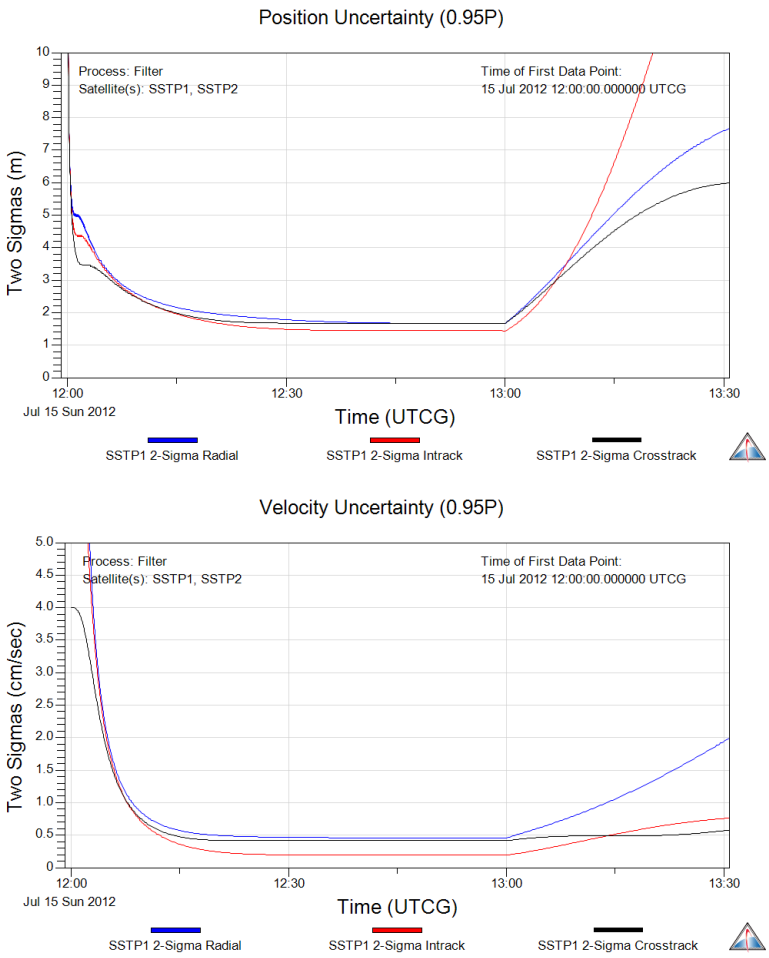

Fig. 5. PONSFD Simulated GPS Position and Velocity Uncertainties

the absolute state of the opposing spacecraft as well. Navigation is also responsible for interpreting the relative observations calculated by the sensor image processor and maintaining a relative state estimate of the opposing spacecraft. The navigation component is thus responsible for estimating both the absolute and relative position and velocity states of the two spacecraft. The navigation system to be hosted on the RPOGNC processor will be implemented as an extended Kalman filter (EKF) which will propagate the states and associated covariances utilizing on-board dynamic models, and will process measurement inputs from the various sensors to perform state and covariance updates.

The sensor data consists of the absolute GPS solutions coming from the spacecraft system processor and the relative observation data coming from the image processor. For relative observations, the image processor analyzes images obtained from the optical sensors to calculate estimates for bearing, range, and pose, with associated rates and uncertainties. The GPS data will be the primary input in the far field range and for the initial rendezvous, but near field approach will require the relative observation data.

While each satellite carries a GPS unit, the GPS may be used sparsely to conserve power. As such, absolute navigation will at times, consist solely of propagating past known states. Similarly, the relative observations will only be available once the spacecraft have maneuvered to within close range of each other and the quality of the measurements will be highly dependent on lighting conditions. To accommodate this inconsistent data availability, the EKF design will be capable of processing measurements as they become available asynchronously from their various sources.

5.1.1 Preliminary GPS Uncertainty Analysis An analysis was completed to compute the expected GPS uncertainty given the nominal planned spacecraft orbit and the known parameters of the currently selected GPS module. The

analysis was performed with the Orbit Determination ToolKit (ODTK) produced by Analytical Graphics, Inc. (AGI) and was conducted by simulating GPS observations given actual GPS constellation orbits and clock biases measured from this year.

Results were computed given the GPS unit's navigation solution (NavSol) output, which is appropriate for on-board use. These results are shown in Fig. 5. Here, the simulation was run for one hour with the GPS unit actively processing observations every second. After an hour into the simulation, the GPS unit is modeled as being turned off, at which point the uncertainty grows. Thus, the graphs provide three realizations: the time to steady-state once the GPS receives its first fix, the steady-state positional uncertainty achieved while the GPS unit is active, and the uncertainty growth rate that occurs once the GPS unit is made inactive.

Simulations were also conducted utilizing the GPS' pseudorange output data, for which data will be stored, but not processed on-board the spacecraft. When in communication with the ground station(s), the CubeSats will download both the GPS NavSol and pseudorange data to the ground station for orbit refinement and tracking.

5.2 Guidance

Another key task for the RPOGNC processor is guidance. This includes maneuver planning automation to achieve a desired formation configuration or trajectory (including docking), as well as automation to safely react to potentially dangerous situations. This will allow each spacecraft to autonomously plan fuel-efficient maneuvers to achieve a desired trajectory as well as compute adjustment maneuvers to correct for thrusting or navigation errors. Ground software will also work in conjunction with the on-board software (when the RPO GNC subsystem is in hybrid state) to validate and approve maneuvers as necessary.

The guidance component will take in the current spacecraft state (and uncertainty) and return a sequence of impulsive thrusts and application times required to accomplish a commanded maneuver. In hybrid state, the thrust sequence will be relayed to the ground for validation before being executed on-board. Commanded maneuvers are designed to guide the spacecraft to a specified waypoint or to follow a desired trajectory with respect to the other spacecraft. Specific trajectories used to accomplish these goals depend on the phase of the mission and the navigation accuracy at the time of the maneuver. For example, for initial rendezvous, when confidence in relative navigation sensors is not strong, a walking safety ellipse [11–13], a passively safe trajectory, will be used to mitigate the risk of uncontrolled along-track drift. Later in the mission, once sensor performance has been validated, non-passively safe trajectories with closed-loop control will be used for final approach and docking along the V-Bar.

In typical operational formation flying environments, maneuvers are targeted using Cartesian coordinates with HCW-based or numerical algorithms. This is due to the conceptual ease of formulating the guidance problem using Cartesian coordinates; however, such methods are numerically intensive, suffer from non-smooth dynamics, and have design spaces replete with local minima. Differential orbital elements, on the other hand, are a natural choice for designing spacecraft formations since they are constants of the unperturbed motion and employing mean elements allows for explicit inclusion of secular J_2 effects [16]. The PONSFD guidance system will solve for fuel-optimal impulsive maneuvers using differential mean elements with a formulation similar to the ones suggested by Breger and How [17] and Hamel and de Lafontaine [18], in which the analytical solution to the unforced dynamics is given by the Gim-Alfriend state transition matrix [19] and the control influence is given by Gauss's Variational Equations [20].

6. SUMMARY

This paper illustrates the current design philosophy for the RPOGNC processor associated with the NASA PONSFD CubeSat program. A functional overview was provided that described the major RPO software components, as well as the necessary underlying infrastructure. Special care was taken to provide a framework that allowed a soft-real-time capability for important periodic task execution, while also providing a mechanism to execute long duration, computationally complex tasks. The most important functional capabilities, navigation and guidance, were explored in additional detail. The Navigation software component strives to maintain an accurate representation of the current proximity operations state by processing absolute information from the GPS module with relative observational data obtained from the image processor's optical sensors. The Guidance component must autonomously calculate

thrust maneuvers to be able to achieve a desired orbital state or trajectory at a given time. The result is a complex, but achievable design that provides ground-breaking autonomous proximity operations capability in a small CubeSat package.

7. ACKNOWLEDGMENTS

The authors wish to thank Scott MacGillivray, Al Tsuda, and Sean Fitzsimmons at Tyvak Nano-Satellite Systems LLC for their support and technical input to this work.

REFERENCES

[1] Tyvak Nano-Satellite Systems LLC. http://tyvak.com.

[2] J. Wilmot, "Use of CCSDS File Delivery Protocol (CFDP) in NASA/GSFC's Flight Software Architecture: core Flight Executive (cFE) and Core Flight System (CFS)," tech. rep., NASA/Goddard Space Flight Center.

[3] J. Wilmot and M. Bartholomew, "A Core Plug and Play Architecture for Component-Based Reusable Flight Software Systems," tech. rep., NASA/Goddard Space Flight Center, 2006.

[4] M. M. P. C. Dwyer, "Embedded Software Design for the Canadian Advanced Nanospace eXperiment Generic Nanosatellite Bus," Master's thesis, University of Toronto, 2009. Graduate Department of Aerospace Engineering.

[5] M. A. Salada, S. Ryan, J. Donnelly, and G. Tate, "Software Lessons from the University of Colorado's Student Nitric Oxide Explorer," 12^{th} *AIAA/USU Conference on Small Satellites*, 1998.

[6] J. S. Fant, H. Gomaa, and R. G. Pettit, "Architectural Design Patterns for Flight Software," 14^{th} *IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops*, 2011.

[7] J. S. Fant, "Building Domain Specific Software Architectures from Software Architectural Design Patterns," 33^{rd} *International Conference on Software Engineering (ICSE)*, 2011.

[8] J. Fant, H. Gomaa, and R. Pettit, "DesDesign Command and Data Handling (C&DH) Subsystems from Software Architectural Design Patters," tech. rep., The Aerospace Corporation and George Mason University, December 7 2010.

[9] U. V. R. Sarma, N. Pavani, and P. Premchand, "Building Software Architecture using Architectural Design Patterns," *International Journal of Science and Engineering Applications (IJSEA)*, Vol. 2, No. 4, 2013, pp. 71–77.

[10] W. Lan, "Poly Picosatellite Orbital Deployer (P-POD) Mk III ICD," tech. rep., California Polytechnic State University, San Luis Obispo, CA 93407, August 2 2007. Preliminary Version.

[11] D. E. Gaylor and B. W. Barbee, "Algorithms for Safe Spacecraft Proximity Operations," *Advances in the Astronautical Sciences*, Vol. 127, 2007. (Proceedings of the 17th AAS/AIAA Space Flight Mechanics Meeting, Sedona, AZ, January 28–February 1 2007).

[12] B. W. Barbee, J. R. Carpenter, S. Heatwole, F. L. Markley, M. Moreau, B. J. Naasz, and J. VanEepoel, "Guidance and Navigation for Rendezvous and Proximity Operations with a Non-Cooperative Spacecraft at Geosynchronous Orbit," *George H. Born Symposium*, Boulder, CO, May 13-14 2010.

[13] S. D'Amico and O. Montenbruck, "Proximity Operations of Formation-Flying Spacecraft Using an Eccentricity/Inclination Vector Separation," *Journal of Guidance, Control, and Dynamics*, Vol. 29, May-June 2006, pp. 554–563.

[14] Wikipedia, "Real-Time Computing," http://en.wikipedia.org/wiki/Real-time_computing.

[15] Wikipedia, "Advanced Configuration and Power Interface (ACPI)," http://en.wikipedia.org/wiki/Advanced_Configuration_and_Power_Interface.

[16] K. T. Alfriend and H. Yan, "An Orbital Elements Approach to the Nonlinear Formation Flying Problem," *International Formation Flying Symposium*, Toulouse, France, October 29–31 2002.

[17] L. S. Breger and J. P. How, "Formation Flying Control for the MMS Mission Using GVE-Based MPC," *IEEE Conference on Control Applications*, Toronto, Canada, August 28–31 2005.

[18] J.-F. Hamel and J. d. Lafontaine, "Linearized Dynamics of Formation Flying Spacecraft on a J_2-Perturbed Elliptical Orbit," *Journal of Guidance, Control, and Dynamics*, Vol. 30, November–December 2007, pp. 1649–1658.

[19] D.-W. Gim and K. T. Alfriend, "State Transition Matrix of Relative Motion for the Perturbed Noncircular Reference Orbit," *Journal of Guidance, Control, and Dynamics*, Vol. 26, November–December 2003, pp. 956–971.

[20] R. H. Battin, *An Introduction to the Mathematics and Methods of Astrodynamics*. Reston, VA: American Institute of Aeronautics and Astronautics, Inc., Rev. ed., 1999.