



Software Approach for the Regenerative Fuel Cell Project

*Mathew J. McCaskey
HX5, LLC, Brook Park, Ohio*

*Elizabeth E. Ash
Glenn Research Center, Cleveland, Ohio*

*Kevin R. Carmichael
Kevin Carmichael Consulting LLC, Westlake, Ohio*

*Phillip J. Smith and Ian J. Jakupca
Glenn Research Center, Cleveland, Ohio*

NASA STI Program . . . in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA Scientific and Technical Information (STI) Program plays a key part in helping NASA maintain this important role.

The NASA STI Program operates under the auspices of the Agency Chief Information Officer. It collects, organizes, provides for archiving, and disseminates NASA's STI. The NASA STI Program provides access to the NASA Technical Report Server—Registered (NTRS Reg) and NASA Technical Report Server—Public (NTRS) thus providing one of the largest collections of aeronautical and space science STI in the world. Results are published in both non-NASA channels and by NASA in the NASA STI Report Series, which includes the following report types:

- **TECHNICAL PUBLICATION.** Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counter-part of peer-reviewed formal professional papers, but has less stringent limitations on manuscript length and extent of graphic presentations.
- **TECHNICAL MEMORANDUM.** Scientific and technical findings that are preliminary or of specialized interest, e.g., “quick-release” reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- **CONTRACTOR REPORT.** Scientific and technical findings by NASA-sponsored contractors and grantees.
- **CONFERENCE PUBLICATION.** Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or co-sponsored by NASA.
- **SPECIAL PUBLICATION.** Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.
- **TECHNICAL TRANSLATION.** English-language translations of foreign scientific and technical material pertinent to NASA's mission.

For more information about the NASA STI program, see the following:

- Access the NASA STI program home page at <http://www.sti.nasa.gov>
- E-mail your question to help@sti.nasa.gov
- Fax your question to the NASA STI Information Desk at 757-864-6500
- Telephone the NASA STI Information Desk at 757-864-9658
- Write to:
NASA STI Program
Mail Stop 148
NASA Langley Research Center
Hampton, VA 23681-2199



Software Approach for the Regenerative Fuel Cell Project

*Mathew J. McCaskey
HX5, LLC, Brook Park, Ohio*

*Elizabeth E. Ash
Glenn Research Center, Cleveland, Ohio*

*Kevin R. Carmichael
Kevin Carmichael Consulting LLC, Westlake, Ohio*

*Phillip J. Smith and Ian J. Jakupca
Glenn Research Center, Cleveland, Ohio*

National Aeronautics and
Space Administration

Glenn Research Center
Cleveland, Ohio 44135

Acknowledgments

The authors would like to thank Joe Hickey and Lisa VanderAar for introducing the RFC software team to the core Flight System architecture and providing help to improve the overall RFC software design.

This report contains preliminary findings,
subject to revision as analysis proceeds.

Trade names and trademarks are used in this report for identification
only. Their usage does not constitute an official endorsement,
either expressed or implied, by the National Aeronautics and
Space Administration.

Level of Review: This material has been technically reviewed by technical management.

Available from

NASA STI Program
Mail Stop 148
NASA Langley Research Center
Hampton, VA 23681-2199

National Technical Information Service
5285 Port Royal Road
Springfield, VA 22161
703-605-6000

This report is available in electronic form at <http://www.sti.nasa.gov/> and <http://ntrs.nasa.gov/>

Software Approach for the Regenerative Fuel Cell Project

Mathew J. McCaskey
HX5, LLC
Brook Park, Ohio 44135

Elizabeth E. Ash
National Aeronautics and Space Administration
Glenn Research Center
Cleveland, Ohio 44135

Kevin R. Carmichael
Kevin Carmichael Consulting LLC
Westlake, Ohio 44145

Phillip J. Smith and Ian J. Jakupca
National Aeronautics and Space Administration
Glenn Research Center
Cleveland, Ohio 44135

Summary

The Regenerative Fuel Cell (RFC) project is developing a system that combines a fuel cell subsystem for energy production with an electrolysis subsystem for energy storage. The system is being developed primarily at the NASA Glenn Research Center with support from the NASA Johnson Space Center under the Space Technology Mission Directorate's Game Changing Development Program. Currently, the intent is to operate the RFC system in an environment equivalent to that of a lunar equator location for several lunar day/night cycles with minimal operator input. Demonstrating an automated RFC system in this environment is a step toward showing the extensibility to a flight system. This paper describes the RFC software team's approach for development of the monitoring and control software that allows for the automated operation of the RFC system. We describe the rationale for the choice of core Flight System architecture, as well as the plan for software and hardware integration.

1.0 Introduction

The primary objective of the Regenerative Fuel Cell (RFC) project is to advance RFC technology from technology readiness level (TRL) 3 to TRL 5 (Ref. 1) or greater by designing, developing, and demonstrating, via ground test, an automated RFC within a thermal vacuum (TVAC) facility that simulates the temperature and pressure environment of an equatorial lunar landing site. This takes RFC technology from proof of concept through operation in a relevant environment. To be judged minimally successful, the system must fully support an external load profile throughout at least one full lunar day/night cycle within the TVAC chamber, have a significantly higher system-level specific energy than the state-of-the-art (SOA) packaged Li-ion battery system, and provide round-trip efficiency greater than 50 percent. Furthermore, system reliability is greatly valued, so the RFC must also demonstrate operation over more than 13 total lunar cycles in a laboratory environment. At the time of project formulation, SOA

packaged Li-ion battery systems have a specific energy of 150 to 200 Wh/kg (540 to 720 kJ/kg) (Refs. 2 and 3). There currently exist no packaged RFCs to provide a comparable SOA value.

Alone, the electrochemical and fluidic balance of plant (BoP) cannot deliver the performance required to meet this success criteria. This paper describes the design of the software that enables the discrete components within the BoP to effectively operate as an integrated system. Following a brief description of the RFC system, operational modes, state machine, health monitoring, and fault detection, the software development approach is detailed from initial architectural considerations through operational qualification. Although the RFC project is not a flight program, the Space Technology Mission Directorate encourages advancing relevant technologies, components, and/or features toward flight certification as early in the development process as feasible. To that end, the benefits are evaluated for core Flight System (cFS) architecture with electronic data sheets (EDS) support. The resulting RFC-specific software architectures and application components are described. Lastly, an integration approach using a software breadboard system is outlined.

2.0 Regenerative Fuel Cell Overview

An RFC system integrates an electrochemical fuel cell and an electrochemical electrolytic cell into a closed-loop integrated system. Many such cells are often assembled in series and called stacks, becoming a fuel cell stack or an electrolyzer stack. This RFC system is based on conversion and storage of liquid water and gaseous hydrogen and oxygen. When electrical power is required, such as during a lunar night, the fuel cell converts the stored chemical energy of hydrogen and oxygen into heat, water, and electricity. The system recharges when an external power source, such as a solar array, supplies electricity to the electrolyzer to dissociate the fuel cell product water back into hydrogen and oxygen gases.

There are many subsystems within the RFC to manage, including fluid, thermal, electrochemical, and power management and distribution (PMAD) functions. Hardware units requiring software control are primarily the fluidic and thermal components and the electrochemical stacks. The process control loop diagram shown in Figure 1 illustrates the major operations of the RFC fluidic system. These same processes typically exist for all closed-loop RFCs based on the hydrogen/oxygen/water cycle.

2.1 Fuel Cell Stack

An aerospace fuel cell is different in design than one intended for terrestrial use (Ref. 4). Whereas most hydrogen fuel cells obtain oxygen from air flowing through a stack, lack of atmosphere leads space designs toward utilization of pure oxygen. This is not a trivial modification, even though cell voltage performance is improved by use of pure, pressurized hydrogen and oxygen reactants. Without the ability to remove generated water from the reaction site by airflow, an alternative technology is necessary. Several NASA programs have coordinated to fund aerospace proton exchange membrane (PEM) (also known as polymer electrolyte membrane) stack development since the 2000s (Refs. 5 and 6). Non-flow-through stack designs emerged as the leading candidate for further research due to their successful performance and because there is no need for an active reactant or water recirculation. These are highly functional stacks when supplied with high-purity reactants, but not all BoP components are passive within the current system designs.

Along with the fuel cell stack, several other components require control, such as heaters, solenoid valves, and a coolant pump. Significant data inputs include individual fuel cell voltages, current load, and fluid temperatures and pressures. Based on these inputs, the avionics system must execute appropriate

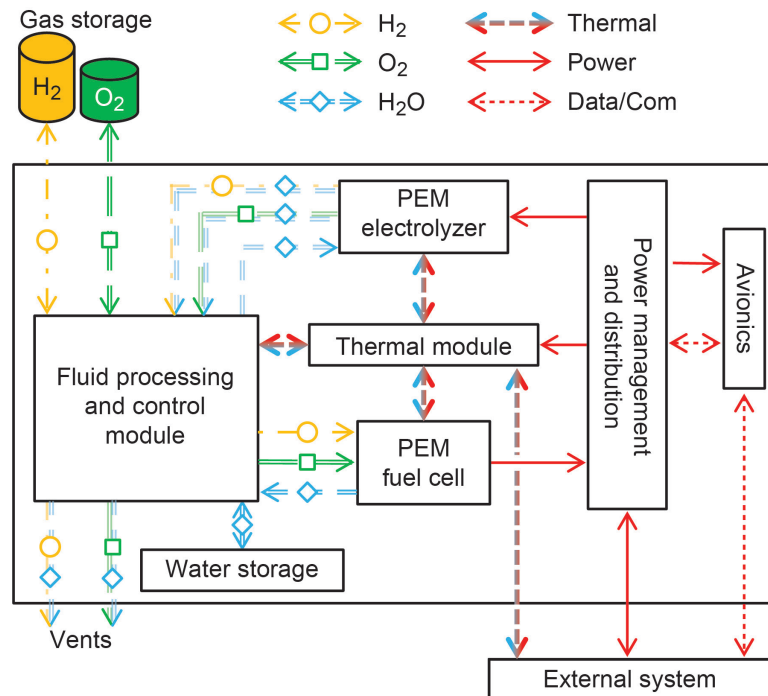


Figure 1.—Regenerative fuel cell system process control diagram within medium temperature thermal envelope. Proton exchange membrane, PEM.

commands to maintain the stack in a preferred operational condition. For example, coolant temperature or reactant differential pressure off-nominal deviations can severely and permanently impair stack performance and life. Therefore, the software must be structured to effectively and timely prioritize and execute all tasks. For this system design, these actions encompass pulse-width modulation of solenoid valves and feedback control of pump speed, among many others.

2.2 Electrolyzer Stack

Several different electrolyzer concepts exist; this RFC design is specific to a liquid anode feed stack. This type theoretically enables the highest electrolysis rate performance, but the required fluidic BoP is more complicated than that of other conceptual designs. Hydrogen and oxygen product gases must be fully separated from water upon exiting the electrolyzer. Water will make up a significant fraction of the fluid mixtures exiting the stack. It is imperative to capture and recirculate all water to avoid water loss, both to minimize initial total water storage quantity and to accommodate the cold temperatures to which the gas storage vessels will be exposed in the TVAC chamber and on the lunar surface. A lunar RFC system presents a unique thermal challenge, specifically at night during fuel cell operation, but it is the electrolyzer, and the associated BoP lunar day performance, that establishes overall system effectiveness.

Electrolyzer operation requires more complex BoP system control than fuel cell operation. In the electrolysis mode, water management becomes more challenging throughout the system. Electrolyzer feed water must be pressurized, thermally managed, separated from hydrogen and oxygen product gases, and recirculated in coordination with a simulated lunar solar array power profile. Therefore, in addition to controlling solenoid valves, heaters, and the coolant pump, the software is responsible for several other actions with respect to the electrolyzer system. The software determines when it is worthwhile to recharge while monitoring the system charge level via storage vessel pressures and available charge power. The

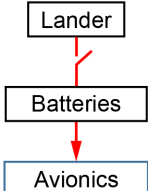
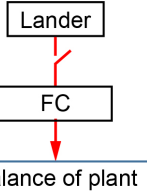
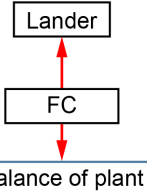
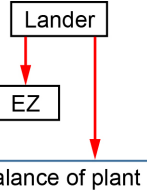
stack performance must be monitored via cell voltage values and gas purity metrics. The software must also manage reactant transfer through backpressure regulator piloting and pump cycling based on liquid-level-sensor feedback.

2.3 Operational Modes

The operational modes define functional steady-state conditions of the RFC system. Within each mode, system state tables define operational parameter inputs and the active state of actuators. Where required, the onboard avionics execute health monitoring and fault detection per the system fault tables to ensure system safety for both personnel and hardware. The operational modes, outlined in Table I, consist of the following:

- Unpowered mode. This is the default mode when the RFC system is nonoperational. The system is completely de-energized and inert, including all actuators. No electrical power is available for actuators or avionics such as instrumentation, processors, and so forth.
- Safe mode. This mode is defined as “safe” from a health and safety perspective. With all the actuators in the de-energized/fail-safe state, select portions of the avionics subsystem are electrically energized to enable passive monitoring of the system process state and/or basic software operations (downloading data, conducting system diagnostics, or uploading software updates).
- Standby mode. This mode is effectively an idled fully operational power system. The health monitoring and fault detection systems are fully operational and monitor all aspects of the system. The RFC utilizes the fuel cell subsystem to power all internal hotel loads, avionics, thermal control systems, and fluidic control systems.
- Normal: Fuel Cell mode. This mode differs from the Standby mode only in that PMAD enables the RFC to supply electrical and thermal power across the interface to the external system.
- Normal: Electrolyzer mode. This is the active charge mode for the RFC. The onboard avionics enable the electrolysis subsystem to receive electrical power through the RFC system interface. This power supports all internal electrical loads, including the electrolyzer.

TABLE I.—REGENERATIVE FUEL CELL (RFC) OPERATIONAL MODES

Mode	Unpowered	Safe	Standby	Normal: Fuel Cell (FC)	Normal: Electrolyzer (EZ)
RFC status	Off	Processor only	Ready	Discharge	Charge
Status description	System de-energized	System dormant	FC powers RFC system only	FC powers lander and RFC system	Lander powers RFC system electrolysis
Avionics	Off	Processor access with limited data collection	Full data collection with active health monitoring and fault detection	Full data collection with active health monitoring and fault detection	Full data collection with active health monitoring and fault detection
Thermal control	Lander	Lander	RFC	RFC	RFC
Power source	Not applicable	Lander	FC	FC	Lander
Powered elements	Not applicable				

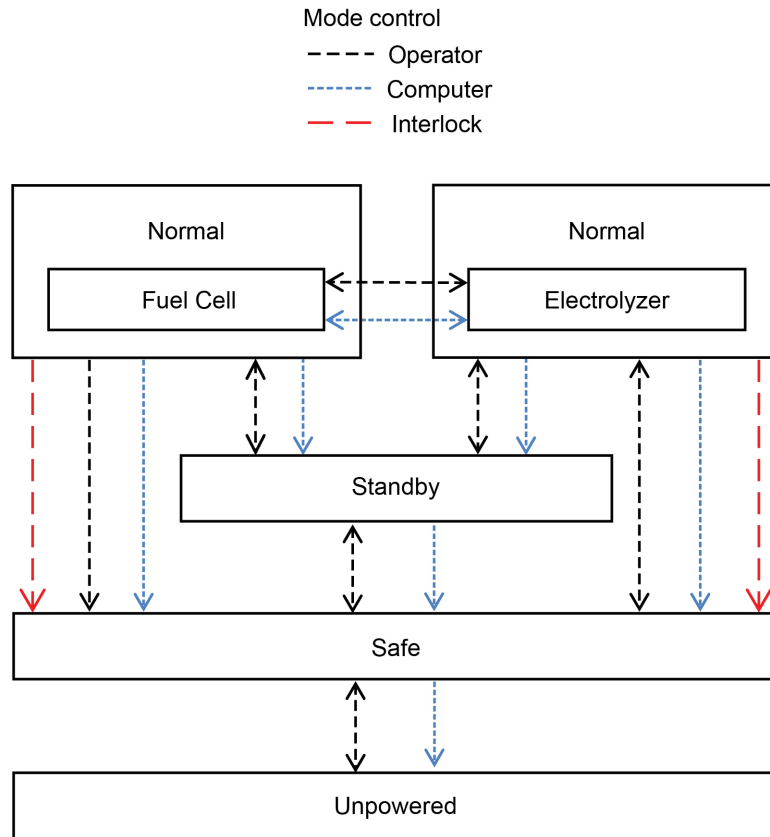


Figure 2.—Regenerative fuel cell operational mode transitions.

2.4 State Machine, Health Monitoring, and Fault Detection

A state machine control architecture is utilized to execute the processes of the active functional mode and the transitions between modes. When transitioning from one mode to another, the onboard avionics execute defined RFC system state changes, as conceptually depicted in Figure 2.

Only an operator may initiate mode transitions toward more complex modes (upward), whereas the avionics, operator, or safety interlock system may initiate mode transitions to less complex modes (downward). Automated transition between the Normal: Fuel Cell mode and Normal: Electrolyzer mode is based on the available power from the lander and the system state of charge, as calculated from the quantity of reactants stored within the reactant storage tanks.

Health monitoring entails calculating system health parameters and reporting the results to the operator. Actuator states are also checked against the state table of the current operational mode to verify appropriate system outputs.

Finally, fault detection entails regularly comparing current sensor values against the caution, warning, and alarm ranges defined in the system fault tables for the current operational mode. Operational data, such as individual cell voltages and stack pressures, verify that the system is operating as intended. Any anomalous event triggers a response in the software based on the severity of the fault, up to and including rapidly transitioning the system back to Safe mode.

3.0 Regenerative Fuel Cell Software Approach

The software enables automatic operation of the RFC without continuous human interaction. The RFC software consists of three computer software configuration items (CSCIs):

- Operational Program (OP)
- Operator Interface (OI)
- Hardware Simulator (HS)

The OP runs on the embedded processor in the RFC hardware and is based on a cFS architecture. The primary functionality of the OP is to directly monitor and control the RFC hardware. It manages the state machine that handles the RFC in each operational mode and in transitions between operational modes. Regular health monitoring and fault detection based on the system fault tables is performed, which may yield additional necessary fault actions. Operational data, incoming commands, and faults are stored in log files that can be downloaded by the operator for future analysis.

The OI runs as a standalone application on a remote computer that interfaces with the OP, allowing the operator to monitor and command the RFC system. The OI receives OP telemetry data that includes operational data and event messages to be presented in various formats (e.g., value fields, tables, and graphs) in the display. Depending on the authentication level of the operator, command requests to download log files, change operational modes, and turn on/off individual actuators are available to send to the OP.

The HS is a tool that simulates RFC hardware responses to informally test the OP and OI software. As such, the HS will be limited as a development tool only and will not be used for formal software verification. It facilitates software evaluation and fault testing when it is not feasible to use actual hardware. The HS is also based in a cFS architecture and can run on the same processor as the OP or as a standalone application with appropriate analog and digital interfaces to the OP.

3.1 Benefits of Using Core Flight System Architecture

After evaluating the requirements and the risk posture associated with developing all the necessary functionality of RFC software, cFS was selected as the software architecture. cFS was originally developed in-house by the NASA Goddard Space Flight Center for satellite applications to provide the core structure commonly required in spaceflight software (Refs. 7 and 8). After numerous successful flight software developments, cFS has become an open-source foundation for spaceflight software that continues to be improved by users across NASA. A group from Goddard is tasked with managing and maintaining the software to benefit all NASA centers (Ref. 9). Numerous NASA projects have contributed to the growth, verification, and validation of cFS. By using cFS architecture, the RFC software becomes more extensible to a potential flight system.

For the RFC project, the software team obtained access to NASA's Spacesuit Caution/Warning System (CWS) project software repositories. The CWS software was based on a cFS architecture and had requirements similar to those of the RFC software. CWS consists of an input/output interface with a set of sensors and actuators, a state machine, and a fault management system. This functional similarity makes the CWS codebase an excellent basis for developing the monitoring and control software for the RFC system.

The base cFS distribution comes with a series of core Flight Executive (cFE) applications that provide the framework used to build the RFC-specific CSCIs. One of the most utilized applications is the Software Bus (SB), which serves as an interapplication communication service. Applications can publish data to the SB, and then any other application can subscribe to receive and use that data. This simplifies the design by

providing a proven internal messaging system. Within this framework, applications can be added and removed in a modular fashion, making it easier to develop and test functionality in a piecewise manner.

cFS provides a layered architecture that streamlines development. The target operating system and drivers are abstracted out, meaning that application software does not need to be rewritten if the operating system or hardware changes. This allows a relatively straightforward transition between simulated and actual hardware when switching between various testing scenarios. A community of core flight developers is available to help if problems occur during development, and bug fixes can be quickly uploaded to the cFS codebase.

3.2 Electronic Data Sheets

The use of electronic data sheets (EDS), defined in the Consultative Committee for Space Data Systems (CCSDS) Recommended Standard 876 (Ref. 10), helps with the interface between software programs, as EDS files define all of the data structures needed for any type of communication message. The data structure definitions, metadata, and application program interface (API) functions are compiled in a linkable library called EdsLib (Ref. 11). In general, EDS is application agnostic and can be used within any program architecture. However, a distribution of cFS with EDS support creates a secondary library called CFE_MissionLib, which interfaces the data structures of EDS with the messaging system of cFS (Ref. 12).

In addition to these libraries, Python bindings have been developed so that both EDS and interface objects can be created and used within Python. The OI utilizes these objects to create the command messages sent to the OP and automatically decode telemetry messages from the OP. This is the primary method for conveying RFC operational data to the operator and allowing the operator to send commands to the RFC system.

3.3 Operational Program Architecture

Figure 3 diagrams the OP cFS architecture.

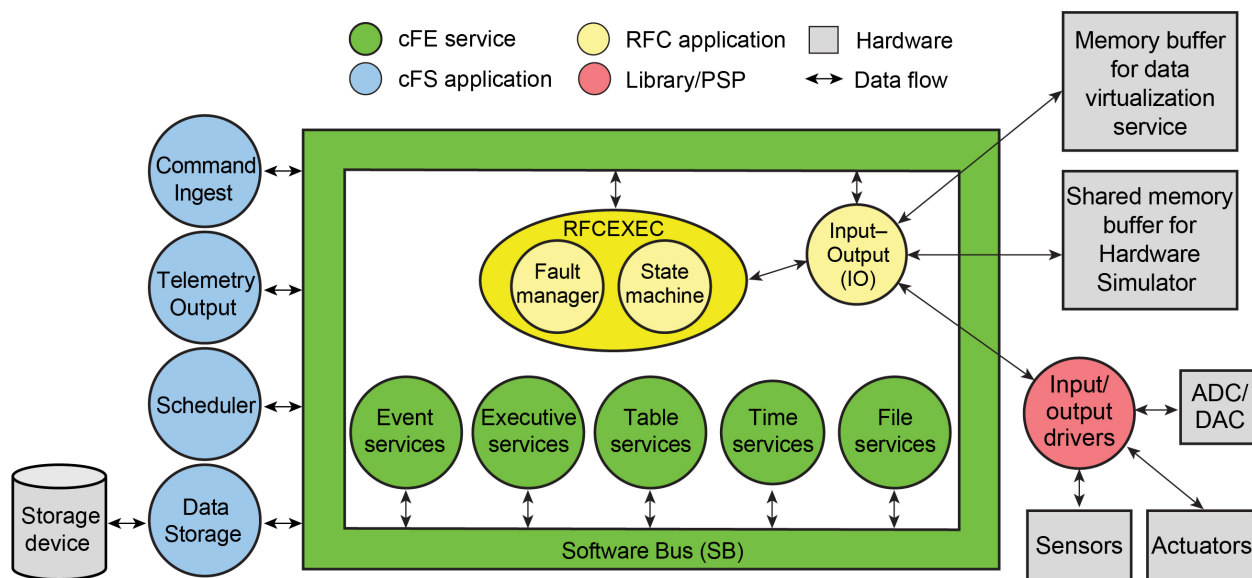


Figure 3.—Operational Program architecture. Regenerative fuel cell, RFC; core Flight Executive, cFE; core Flight System, cFS; platform support package, PSP; RFC Executive, RFCEXEC; analog/digital converter, ADC; digital/analog converter, DAC.

For any cFS-based CSCI, the primary structure is given by the cFE services. Not only do these services handle common objects such as files, tables, and events, but they also provide the message-passing service via the SB (Ref. 8). This allows all the other applications to consistently communicate with each other. Applications from the cFS community (Ref. 9) provide additional functionality common among spaceflight software applications but not explicitly needed to run a cFS-based CSCI. These applications cover sending and receiving SB messages with an external program, scheduling, and data storage.

Specific to the OP, the software team has developed applications to satisfy project needs. The RFC Executive (RFCEXEC) application manages the state machine, health monitoring, fault detection, and fault actions of the RFC system. The Input-Output (IO) application handles the interface between RFCEXEC and system hardware. A more detailed discussion of the application functionality is given in Section 3.5.

Finally, hardware-specific components are incorporated in the architecture. Applications such as IO and Data Storage (DS) interface with volatile memory for temporary data storage and nonvolatile memory for long-term data storage. Hardware drivers are managed in platform support package (PSP) modules that can be easily swapped in and out as hardware is changed.

3.4 Hardware Simulator Architecture

Figure 4 illustrates the cFS architecture diagram for the HS.

The HS uses many of the same applications as the OP. Most notable is the IO application, which can be easily configured to meet the needs of the HS while leaving the application itself unchanged. If the OP and HS are running in a processor-in-the-loop (PIL) testing scenario, the IO interfaces with a shared memory buffer to send information between both programs. Alternatively, in a hardware-in-the-loop (HIL) testing scenario, the IO instead interfaces with a series of analog-to-digital converters (ADCs) and digital-to-analog converters (DACs) to transmit physical signals between the OP and HS.

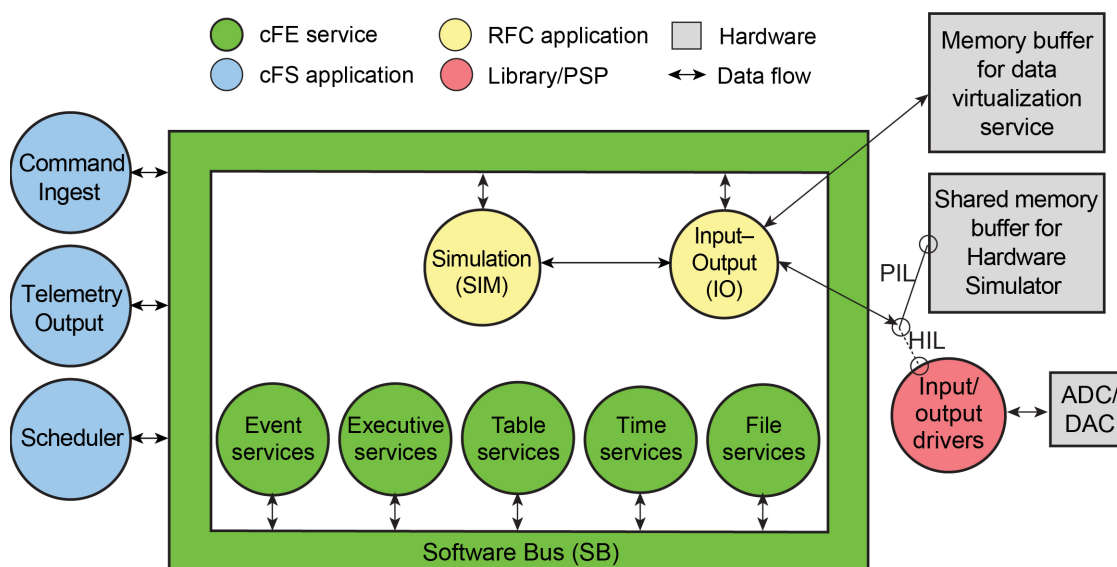


Figure 4.—Hardware Simulator architecture. Regenerative fuel cell, RFC; core Flight Executive, cFE; core Flight System, cFS; platform support package, PSP; processor in the loop (PIL); hardware in the loop, HIL; analog/digital converter, ADC; digital/analog converter, DAC.

The only application that is unique to the HS is the Simulation (SIM) application, which simulates sensor responses based on the current state of all the RFC actuators. When commanded, SIM also injects fault conditions to test that the OP performs appropriate fault responses.

3.5 Regenerative Fuel Cell Developed Core Flight Applications

Along with the base core flight software, such as the cFE applications, the RFC software team has developed several applications specific to the RFC project.

3.5.1 Regenerative Fuel Cell Executive Application

RFCEXEC is the heart of the OP. First, it manages the state machine, including the RFC operational modes and the transitions between modes. Within each operational mode, this application performs fault detection in appropriate sensors based on fault tables specific to that mode. If a sensor value is not within the appropriate bounds, the application creates an event using the Event Generator (EvGen) library to perform appropriate response actions. A set of function calls to the IO allows the use of up-to-date sensor values for the fault detection process.

In addition to fault detection, system health parameters are regularly calculated and reported in telemetry messages to the OI, such as RFC state of charge and overall efficiency. The reported parameters can be used by operators to determine the overall health of the system and whether any additional actions need to be taken by the system to return it to a healthier state.

3.5.2 Input-Output Application

The IO manages the interface between the hardware and the OP or HS. Using a configuration file that is generated with a Lua script (Ref. 13) during the build process, the IO sets up the configuration for all of the hardware channels (both actual hardware and simulated hardware). It also configures the output telemetry messages to be sent to the OI. During normal operation, the IO manages the data acquisition rate of all of the sensor values and has functions that allow other applications to obtain the latest sensor values for any channel.

3.5.3 Simulation Application

The primary function of the SIM application is to simulate sensor responses of the RFC system based on current actuator values. Interfacing with the IO in a manner similar to that of the RFCEXEC, SIM uses a set of function calls to get up-to-date actuator values for simulating sensor responses. If the HS is running in a PIL simulation, SIM calls a PSP module to create a block of memory that is shared with the OP in order to read/write actuator and sensor values. Further details on both PIL and HIL simulation scenarios are given in Section 3.7.

The SIM application can also be commanded to simulate fault responses for a particular sensor and operational mode. In these cases, the fault table values drive the sensor responses as opposed to the state-of-the-system actuators. The duration of each fault simulation is set to assure that a fault triggers in the OP. After the necessary fault persistence, the SIM returns to normal operation.

3.5.4 Configured Core Flight System Applications

To streamline development of common functions, the cFS community developed many applications for use with cFE in a plug-and-play manner (Ref. 9). For the RFC project, we have configured the following applications for use in both the OP and HS:

- Command Ingest (CI). The CI application listens for user datagram protocol (UDP) command messages on a specified port. When a message is received, it is decoded into an SB message and sent along the SB.
- Telemetry Output (TO). The TO application takes the telemetry messages from the SB and sends them out to a specified internet protocol (IP) address via UDP message.
- Scheduler (SCH). The SCH application sends a command message every second to all other applications, which causes them to send a housekeeping telemetry message. These messages contain application status updates that can be used to determine if the applications are working properly.
- Data Storage (DS). The DS application writes SB messages to files, allowing the OP to create and store log files for operational data, commands, faults, and so forth.

3.6 Operator Interface

The OI is a Python3-based CSCI using the PyQt5 framework (Ref. 14). It allows a user to send commands and receive telemetry to and from the OP. The OI consists of three modules: the viewer, the controller, and the data model. Figure 5 shows an overview of the three OI modules and how they interact with a user, the OP, and each other.

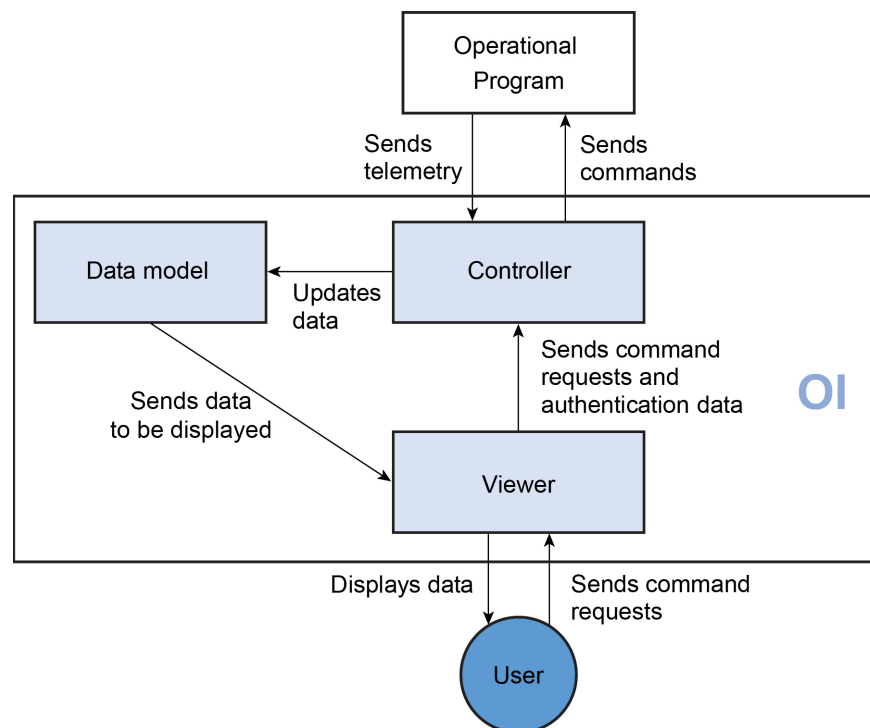


Figure 5.—Operator Interface (OI) architecture.

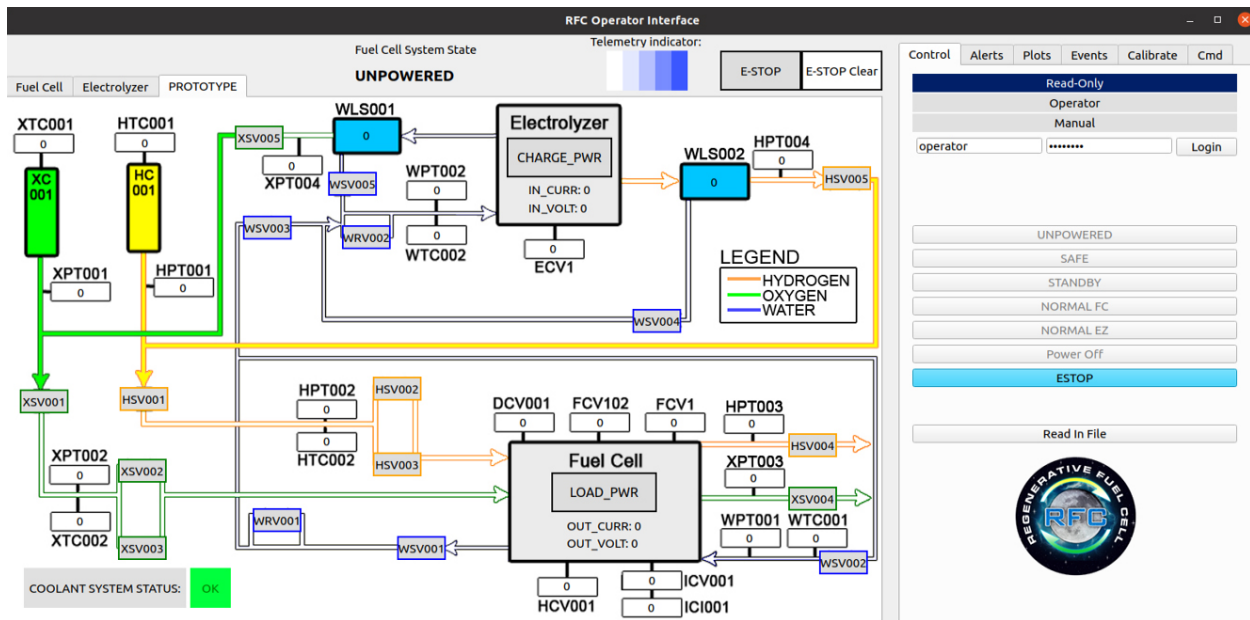


Figure 6.—Operator Interface viewer.

3.6.1 Operator Interface: Viewer

The viewer is the main interface between the operator and the rest of the OI. It displays the operational data stored in the data model to relay meaningful information to the user. Furthermore, it provides a way to send commands to the OP by allowing the operator to send command requests to the controller. Figure 6 shows a sample screenshot of the OI viewer currently in development.

Drawing from lessons learned in previous fuel cell activities (Refs. 15 and 16), the OI contains various images that help the operator navigate the RFC system. These are representations for the fuel cell subsystem, electrolyzer subsystem, and full system, which are shown within a collection of three tabs in the upper left-hand corner of the display. Each subsystem contains associated valves, tubing, sensors, and stack readings.

A labeled telemetry indicator (i.e., a set of colored lights) acts as a heartbeat from the OP. It changes at a rate of approximately 1 Hz and indicates that telemetry has been received by the OI from the OP. If no telemetry is received within 1 second, the indicator no longer changes and an alert is displayed to the operator. This provides a quick visual indicator for the operator to determine if there is a problem with the OP.

The right side of the OI viewer contains six tabs: Control, Alerts, Plots, Events, Calibrate, and Command. The Control tab allows the operator to log in, control the state machine, and download log files. The Alerts tab displays caution, warning, alarm, and error messages captured in incoming telemetry from the OP. The Plots tab shows updated sensor values as time elapses. The Events tab displays telemetry related to event messages. The Calibrate tab displays sensor and actuator data. The Command tab allows the operator to send command requests to the OP, given the appropriate authentication level. These commands could change operational modes, turn actuators off and on, or download telemetry files.

There are three authentication levels available: read-only, operator, and manual. The read-only authentication level is limited to two available command requests: downloading log files and sending an emergency stop (E-STOP) signal. The operator authentication level allows the operator to set sensor calibrations and send command requests for the state machine. The manual authentication level allows the operator to send command requests for all aspects of the RFC hardware.

Additional external computers running the OI in read-only mode would potentially need to receive telemetry data in parallel with the OI that is directly connected to the RFC system. The additional OIs would need to connect to the primary OI via transmission control protocol/internet protocol (TCP/IP). Once a connection is established, the primary OI can forward telemetry messages to the other remotely connected OIs. Operators can only send commands to the RFC system through the primary OI. Any command sent by a remote OI will fail the authentication process handled by the controller.

3.6.2 Operator Interface: Controller

The controller handles the logic behind the telecommand and telemetry systems of the OI as follows:

- **Telecommand system:** With the use of EDS, the controller has the ability to create command messages to be sent to the OP. Using these commands, the controller can determine if it is logged in to the OP by sending a “No Operation” command and receiving a response from the OP in the form of an event message. When the operator sends a command request by clicking on a button in the display, authentication data is sent to the controller based on the login password the operator entered as well as the IP address of the sender. The controller uses this authentication data to determine if the command request is valid. Upon verification, an appropriate command message is created and sent to the OP. This authentication process serves multiple purposes: it assures that the operator is experienced enough to know the correct login password, and it prevents remote OIs from sending commands to the OP.
- **Telemetry system:** When telemetry messages arrive from the OP, or are forwarded from the primary OI, the controller can automatically decode the telemetry packet with the use of EDS. Depending on the message type, the relevant data structures in the data model can be updated with the telemetry information, whether it be operational data, an event message, or an alert. When the viewer refreshes, it is updated with the latest data contained in the data model structures.

3.6.3 Operator Interface: Data Model

The data model is a local memory database that stores information contained in telemetry messages, including operational data, event messages, and alerts. Data structures based on EDS can be used to store information for different purposes in the viewer. For example, a data structure containing all of the current operational values can be used for updating the main display. Arrays are used to temporarily store received channel data, allowing the viewer to create rolling plots showing channel values over time. Throughout operation the data model writes all data to log files stored on the computer running the OI, freeing up buffer space for incoming data.

3.7 Software Breadboard System

As the RFC design develops and the software evolves, a software breadboard system is necessary to integrate the software with prototype hardware. System hardware such as valves and pressure transducers can be wired into the necessary electrical interfaces to connect with the OP. Hardware not represented can be simulated using the HS running on a separate processor with input and output signals running through appropriate ADC and DAC devices. This HIL testing scenario is used both to develop the initial software and to debug the software and procedures as issues arise during hardware integration. Because the HS is sending physical signals via the connected DACs, replacing the HS with real system hardware should be transparent to the OP and require no additional code changes.

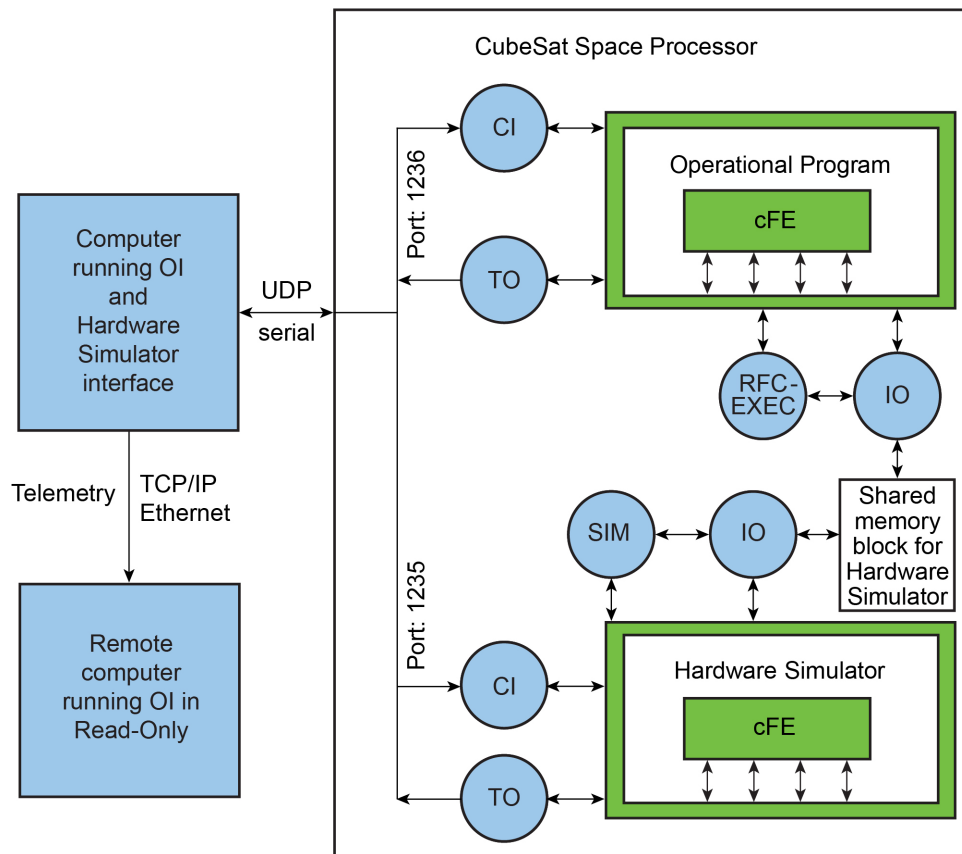


Figure 7.—Interface between computer software configuration items (CSCIs) in processor-in-the-loop (PIL) scenario. Operator Interface, OI; regenerative fuel cell, RFC; transmission control protocol/internet protocol, TCP/IP; user datagram protocol, UDP; Command Ingest, CI; Telemetry Output, TO; core Flight Executive, cFE; RFC Executive, RFCEXEC; Input-Output, IO; Simulation (SIM).

Figure 7 shows an interface diagram between the three CSCIs, including the CubeSat Space Processor (CSP) board, in a PIL scenario. In this scenario, the OP and HS only interface with each other indirectly through the shared memory block that is accessed via the IO application. The OP and HS use this shared memory block to read and write all the sensor and actuator values, effectively creating a virtual hardware system.

The OI and HS interface run on a separate computer that is connected to the hardware via a serial communication link. They can send messages to the corresponding port where the desired CI application is listening. They can also listen on the ports where the TO applications are sending messages to receive telemetry data. If the computer running the OI goes offline during testing, that computer can be rebooted while the OP continues to monitor the RFC system. Once restarted, the OI can verify communication by sending a “No Operation” command and receiving an event response from the OP.

Figure 8 shows an interface diagram between the three CSCIs, including the CSP board in a HIL scenario. In this scenario, the OP and HS now run on separate processors. The IO applications are connected to each other through a series of ADC and DAC devices. The OP connects to both the OI and HS through a custom HIL board currently in development by the RFC avionics group. This HIL board contains the appropriate ADCs and DACs to send and receive external hardware signals.

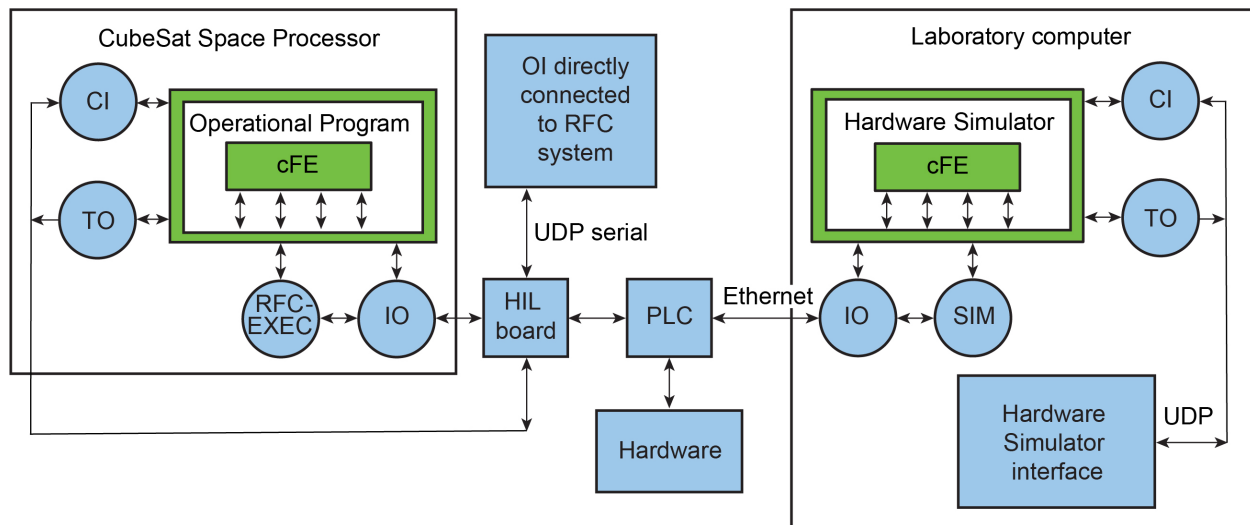


Figure 8.—Interface between computer software configuration items (CSCIs) in hardware-in-the-loop (HIL) scenario. Core Flight Executive, cFE; Command Ingest, CI; Telemetry Output, TO; regenerative fuel cell, RFC; RFC Executive, RFCEXEC; Input-Output, IO; Operator Interface, OI; user datagram protocol, UDP; programmable logic controller, PLC; Simulation (SIM).

The hardware side of the HIL scenario is driven by a programmable logic controller (PLC), which is also being put together by the RFC avionics group. With a PLC system, hardware signals can be driven either by messages sent from the HS or directly connected hardware. This way, real and simulated hardware can easily be changed by swapping components directly connected to the PLC system and making appropriate adjustments in the IO configuration files. An operator can also control the HS (e.g., set simulation parameters or inject fault scenarios) using the HS interface running on the laboratory computer.

4.0 Concluding Remarks

With the Regenerative Fuel Cell (RFC) project goal of demonstrating an automated RFC in a simulated lunar environment, this paper examined the software approach to the monitoring and control software needed for such a system. Using core Flight System (cFS) architecture provides a foundation that helps with the extensibility of the software to a flight system. The RFC software team has developed several prototype cFS applications to meet many of the RFC system requirements. The RFC Executive application handles the majority of the RFC functionality needs, including a state machine, health monitoring, and fault detection. The Input-Output application provides the necessary software interface between the system hardware and the Operational Program (OP) or Hardware Simulator (HS). Finally, the Simulation application provides the development tool needed during hardware/software integration and testing. A prototype operator interface has also been developed that allows an operator to send commands to the RFC system and view telemetry data in a useful format.

The next steps in development include compiling and running the OP and HS on the CubeSat Space Processor board to first test the computer software configuration items in a processor-in-the-loop testing scenario. Then, using the hardware-in-the-loop board and programmable logic controller system, the OP and HS can interact with each other via physical voltage signals. As RFC hardware is developed, it can be integrated into the system and tested with the OP.

Appendix—Acronym List

ADC	analog/digital converter
API	application program interface
BoP	balance of plant
cFE	core Flight Executive
cFS	core Flight System
CCSDS	Consultative Committee for Space Data Systems
CI	Command Ingest ¹
CSCI	computer software configuration item
CSP	CubeSat Space Processor
CWS	Caution/Warning System
DAC	digital/analog converter
DS	Data Storage ¹
EDS	electronic data sheets
E-STOP	emergency stop
EvGen	Event Generator ¹
HIL	hardware in the loop
HS	Hardware Simulator
IO	Input-Output ¹
IP	internet protocol
OI	Operator Interface
OP	Operational Program
PEM	proton exchange membrane
PIL	processor in the loop
PLC	programmable logic controller
PMAD	power management and distribution
PSP	platform support package
RFC	regenerative fuel cell
RFCEXEC	RFC Executive ¹
SB	Software Bus
SCH	Scheduler ¹
SIM	Simulation ¹
SOA	state of the art
TCP	transmission control protocol
TO	Telemetry Output ¹
TRL	technology readiness level
TVAC	thermal vacuum
UDP	user datagram protocol

¹Application.

References

1. Office of the Chief Engineer: NASA Systems Engineering Processes and Requirements. NPR 7123.1B, 2020.
2. Ratnakumar, V. Bugga; and Brandon, Erik J.: Energy Storage for the Next Generation of Robotic Space Exploration. *Electrochem. Soc. Interface*, vol. 29, no. 1, 2020, pp. 59–63.
3. Mehta, Mohit; and Lawson, John: Physics-Based Modeling and Simulation of Emerging Battery Technologies for Aerospace. Presented at the AIAA Propulsion and Energy Forum and Exposition, Indianapolis, IN, 2019. <https://ntrs.nasa.gov>
4. Jakupca, Ian: Fuel Cell Research and Development for Earth and Space Applications. 2018. <https://ntrs.nasa.gov>
5. Hoberecht, Mark A.; and Jakupca, Ian J.: Development Status of PEM Non-Flow-Through Fuel Cell System Technology for NASA Applications. NASA/TM—2011-217107, 2011. <https://ntrs.nasa.gov>
6. Jakupca, Ian, et al.: Non-Flow Through Fuel Cell Power Module Demonstration on the Scarab Rover. Annual AIAA Propulsion and Energy Forum 2017, Atlanta, GA, 2017. <https://ntrs.nasa.gov>
7. Wilmot, J.: A Core Plug and Play Architecture for Reusable Flight Software Systems. Presented at the 2nd IEEE International Conference on Space Mission Challenges for Information Technology, Pasadena, CA, 2006. doi: 10.1109/SMC-IT.2006.7 Accessed Feb. 2, 2021.
8. National Aeronautics and Space Administration: Core Flight Executive: OSAL User’s Guide. https://github.com/nasa/cFS/blob/gh-pages/cFE_Users_Guide.pdf Accessed Feb. 2, 2021.
9. National Aeronautics and Space Administration: core Flight System. 2020. <https://cfs.gsfc.nasa.gov/> Accessed Feb. 2, 2021.
10. The Consultative Committee for Space Data Systems: Spacecraft Onboard Interface Services—XML Specification for Electronic Data Sheets. Recommended Standard CCSDS 876.0–B–1, 2019.
11. National Aeronautics and Space Administration: CCSDS SOIS Electronic Data Sheet Implementation for Core Flight System. LEW–19710–1, 2021.
12. The Consultative Committee for Space Data Systems: Core Flight System With Electronic Data Sheets. <https://github.com/jphickey/cfe-eds-framework> Accessed Feb. 1, 2021.
13. Lua: Lua 5.3 Reference Manual. 2020. <https://www.lua.org/manual/5.3/> Accessed March 30, 2021.
14. Python Software Foundation: The Python Language Reference. 2021. <https://docs.python.org/3/reference/> Accessed March 31, 2021.
15. Bei-jiann, Chang, et al.: Regenerative Fuel Cell Test Rig at Glenn Research Center. NASA/TM—2003-212375 (AIAA 2003–5942), 2003. <https://ntrs.nasa.gov>
16. David Bents, et al.: Hydrogen-Oxygen PEM Regenerative Fuel Cell Development at NASA Glenn Research Center. Presented at the 2006 Fuel Cell Seminar, Honolulu, HI, 2006. <https://ntrs.nasa.gov>

