Design and Test of an Embedded Systems Controller for a BiModal CubeSat Propulsion System

Kaushik Manchikanti* and E. Glenn Lightsey†
Georgia Institute of Technology, Atlanta, GA, 30332

The Spectre Propulsion System is an effort to develop a bimodal propulsion system for CubeSats. Spectre aims to use green propellant to both drive a small chemical monopropellant thruster as well as an electrospray system. The project is a collaboration between NASA MSFC, Georgia Tech, and MIT. The propulsion unit hardware and controller are being developed within the GLRG at the SSDL at Georgia Tech, the systems engineering effort is being led by NASA MSFC, and the electrospray system development is being led by MIT SPL As a part of that effort, a low-cost system controller is being developed using commercial-of-the-shelf parts to drive costs down and still be able to withstand the harsh environments of space missions. This paper shows the development and test process of Spectre Propulsion System controller. From the high-level system architecture to the testing of specific hardware, the total controller system development effort will be outlined.

I. Acronyms

ADC	=	Analog to Digital Converter
EDU	=	Engineering Design Unit
FC	=	Flight Computer
$GLRG$	=	Glenn Lightsey Research Group
$GPIO$	=	General Purpose Input Output
IC	=	Integrated Circuit
PCB	=	Printed Circuit Board
PPU	=	Power Processing Unit
SEL	=	Single-Event Latch-up
SPI	=	Serial Peripheral Interface
SPL	=	Space Propulsion Lab
$SSDL$	=	Space Systems Design Lab

II. Introduction

CubeSats were introduced as a design standard for university labs in 1999. It aimed to allow university labs to design, build, test, and deploy spacecraft rapidly. Doing so allows for students to learn about spacecraft development in a controlled setting and carry that knowledge over to the workforce. Since 2003, when the first CubeSats were launched, there have been massive improvements to space technology that allow for missions that are more complex to be conceived for CubeSats and also allow for CubeSats to serve as "Technology Demonstrators". As the standard was released as a public reference, different institutions have developed unique areas of expertise in space technology development, both in advancing functionality and in

*Graduate Research Assistant, Guggenheim School of Aerospace Engineering
†Professor, Guggenheim School of Aerospace Engineering

miniaturizing exiting technology. This, in turn, leads to spacecraft missions that propose niche "Technology Demonstrations" and for missions that have different labs collaborate to bring their different technologies together to mesh their strengths together in one complex mission proposal.

The SSDL at the Georgia Institute of Technology is one such lab that takes part in developing CubeSat technologies. Over the years, the SSDL has developed an expertise in the areas of system integration and propulsion system development. The SSDL has proposed and delivered multiple systems to develop this expertise, including Bevo-2, Biosentinal, Prox-1, Ascent, SunRISE, VISORS, and LFPS. Each of these mission systems expanded the knowledge that is gained by the lab in CubeSat propulsion. The SSDL has worked on missions that track the progress of propulsion systems on CubeSats, from no on-board propulsion to cold-gas and monopropellant systems.

CubeSats have been able to be proposed for increasingly complex missions, due in part to the advancements in propulsion system technology. One such advancement is the SPECTRE BiModal propulsion system proposed to NASA-MSFC to develop a monopropellant system to drive both a chemical thruster and an array of electrospray thrusters from the same green propellant source. The SSDL would take on the propulsion structure, controller, and system integration efforts with NASA MSFC managing systems engineering, and the MIT SPL leading the electrospray system development effort.

III. SPECTRE System Overview

The following section details the system overview for the proposed SPECTRE propulsions system. The hardware layout and schematic as well as the proposed controller design and functionality will be described. The hardware design will be briefly described with a more in-depth look into the decisions for the electronics development.

A. Hardware

The SPECTRE system is responsible for operating a chemical thruster and an array of electrospray thrusters. A view of the fluid schematic and the physical structure of the system is shown below.

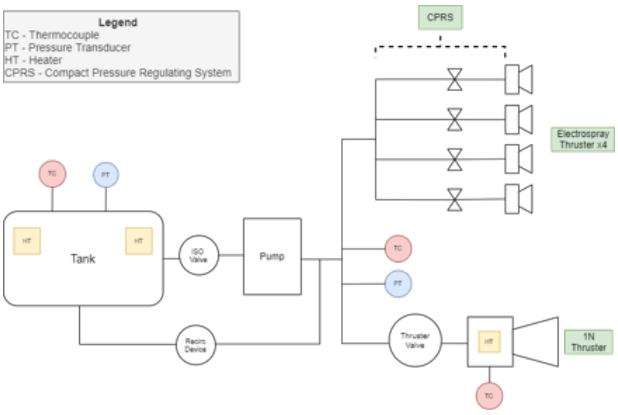


Fig. 1 Spectre System Schematic

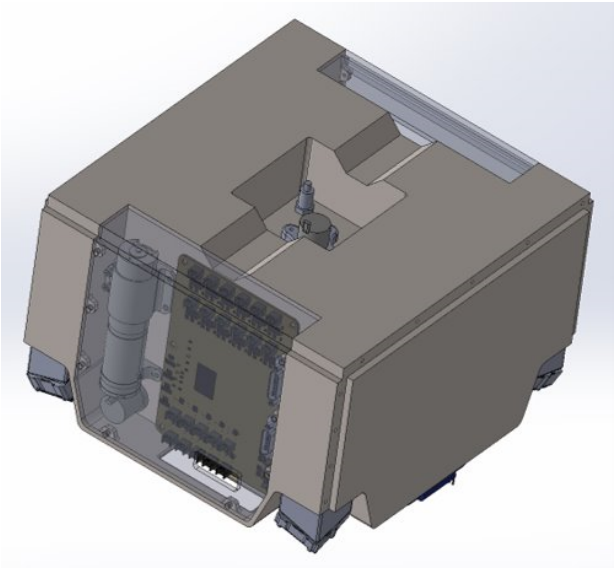


Fig. 2 Spectre System Physical Structure

Shown in Fig. 1 [**?**], the system operates valves to move fluid from the structure to a chemical thruster and

to an electrospray system. It also houses instruments to gather telemetry data in order to get information about the health and state of the system for operations. The structure is made to be compact and house all relevant functionality, which flows down to requiring any electronics to operate the system to also be compact. The layout of the physical structure also informs locations for peripheral connections for the system electronics.

B. Electronics

Given the requirements for the system and the physical layout of the propulsion unit structure, the required functionality and form factor of the electronics can be derived. The SPECTRE system must be capable of actuating a number of valves, a pressure pump, and heaters to induce propellant flow and keep the unit within an acceptable operating temperature. The system must also be capable of collecting housekeeping and health data pertinent to the operation of the system. The housekeeping data consists of values of interest to a larger spacecraft, including various temperatures and pressures across the system.

The interfaces required to satisfy these system level requirements is shown in Table 1 The allocation

Table 1 Controller Interfaces

Valves	6
Heaters	3
Pressure Transducers	2
Thermocouples	10
PPU	1

for the physical volume of the controller was constrained as shown in Fig. 2[**?**], so the layout chosen for the controller finalized as shown in Fig. 7. After iterating on the layout, it was chosen to have all of the electronics consolidated onto one board with strategically placed peripheral connectors to interface with an upstream satellite FC and the controller interfaces as shown in Table 1. While the controller does not conform to the CubeSat PCB standard, the elongated shape does not detract from the performance of the system as a whole, so the deviation in spec was accepted and designed towards. The benefits of this design, namely keeping it relatively simple as a single board and not as an interfacing board stack, outweigh the costs, being that the layout of the components will get cramped and that the design did not follow the standard CubeSat specification.

IV. Controller Development

A. Controller Design Process

The overall goal for the controller of the SPECTRE system is to be able to ensure the health of the system and condition and fire chemical thrusters and electrospray thrusters. As the system is being designed by a university lab, the goal is to keep costs down and use commercially accessible parts to design a controller that can accomplish its base design objectives. The main concern with using commercially available EEE parts for this controller is their response to radiation energy. The Earth protects against many harmful sources of radiation that are apparent in a space environment. Electronics can be especially susceptible to these sources of radiation, as their operation depends on the flow of electrons. Radiation interactions with the electrons inside these devices can cause harmful errors that require special attention to mitigate or fix. One way to account for these effects in the design process is to source parts that are better rated for these disturbances - they can handle more shocks than a normal commercial part. The design for the controller takes into account availability of automotive grade parts to take advantage of their superior performance.

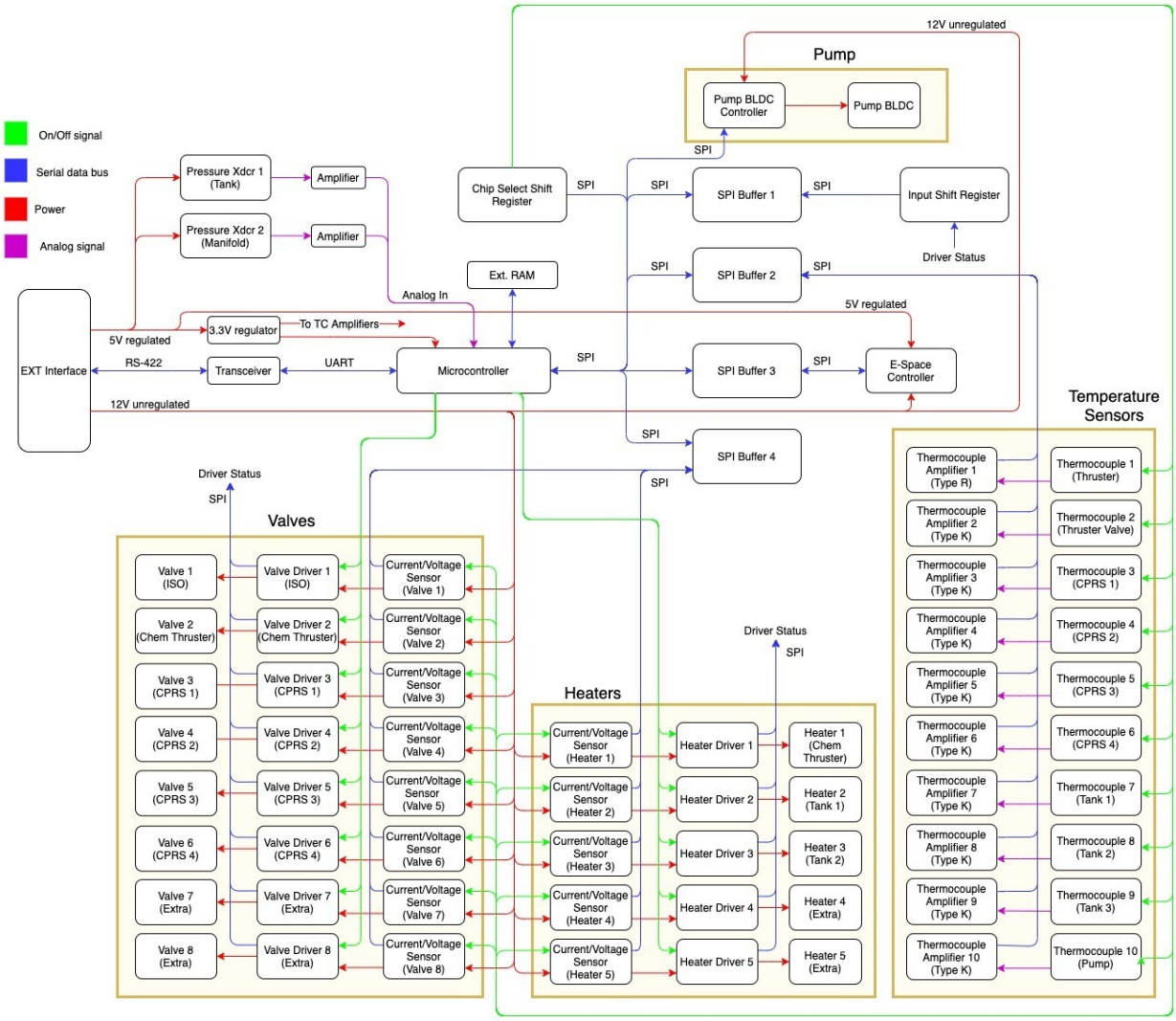


Fig. 3 SPECTRE system controller block diagram

A particularly important phenomenon that the controller needs to guard against is SEL. Particularly as SELs can cause ICs to suddenly short circuit, which is a danger as overcurrent can propagate and damage other ICs present on the PCBs that were not affected by the radiation event. The controller handles SELs by placing current limiting resistors on digital logic lines to offer protection to the ICs that may see an increased current due to a latch-up. Additionally, the digital logic power rails on the controller all have sensing elements to detect any faults and reset power to offer another layer of protection against the effects of harmful radiation.

The SPECTRE system consists of chemical and electrospray thrusters. The structure serves as a 3D printed tank interface and there will be propellant lines running from a central propellant tank area to the valves that serve as the gateways to the thrusters. The controller must be able to keep the propellant and itself at a favorable temperature, and it must be able to guide the propellant down these feed lines as needed. Additionally, as the electrospray system is an electric propulsion system, there is a PPU that it must interface with to operate the high voltage logic that actuates the electric thrusters.

Lab heritage is also a big factor in deciding the parts to use for the controller. The GLRG has proposed and worked on many satellite hardware projects, so its repository for tested parts is extensive. Some of the parts

for the controller design were sourced from this knowledge and others were sourced as being "automotive quality". Critical parts such as microcontrollers, crystal oscillators, and voltage regulators were chosen based on the flight heritage seen in the GLRG, and newer, interface specific parts were ensured to be of automotive quality. A full breakdown of the parts chosen for this controller is shown in Fig. 3. A large part of the organizational structure shown in the figure is due to the need for IO expansion out of the microcontroller. A big decision to make on the electronics structure is picking what communication interface will be used to interface with sensors and actuators. This decision also ripples out to inform the connectivity between those peripherals and a central processor.

For SPECTRE, SPI was chosen as the intra-board communication protocol for its flexibility and BUS architecture. Figs. 4 and 5 show the details of the SPI communication protocol. One of the advantages of the SPI protocol is that during one transaction, a read and a write occur rather than either a read or a write. This allows for a controller structure to be developed as seen in Fig. **??**. A disadvantage of the SPI protocol, however, is that it allows for multiple modes of data transmission, so a processor must be able to account for different peripherals using different SPI modes to transfer data.

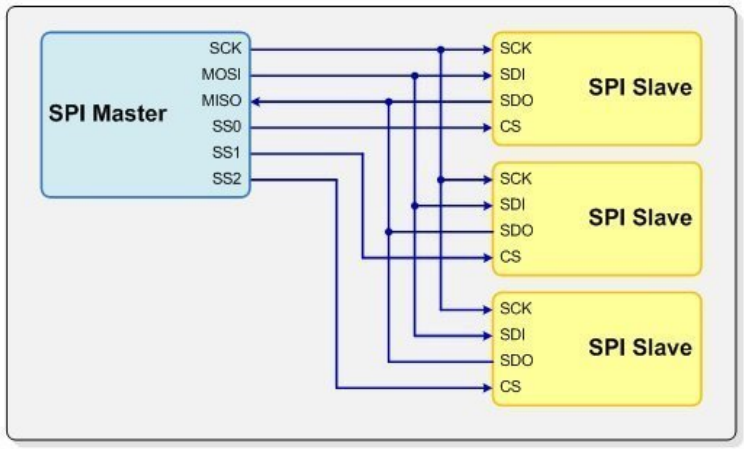


Fig. 4 SPI BUS Architecture

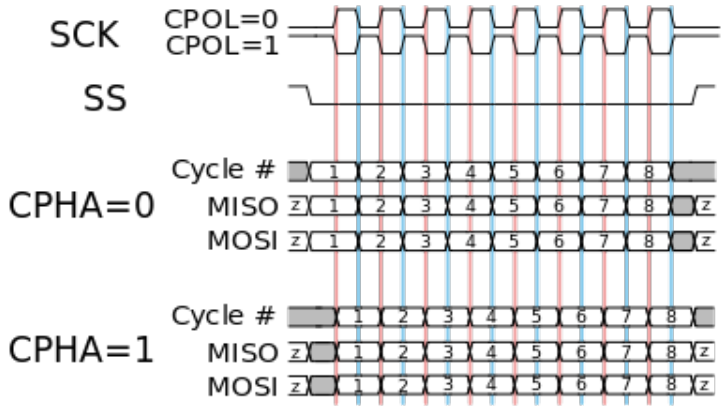


Fig. 5 SPI Protocol

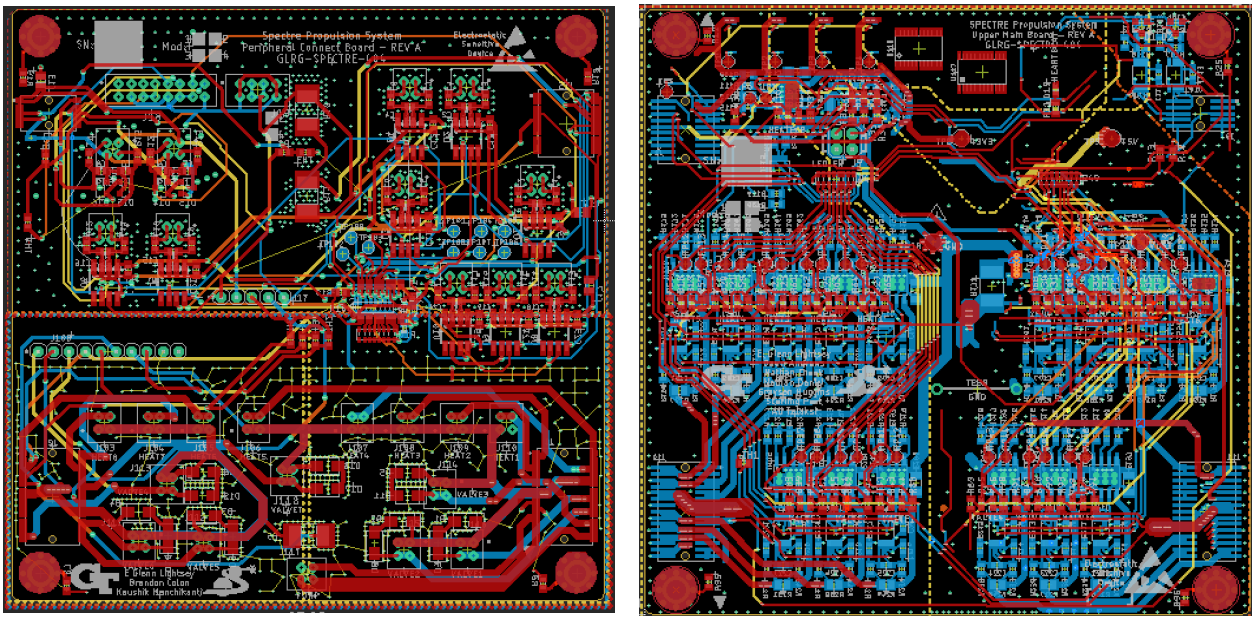
The different components in the controller design are as follows. Firstly, the SPECTRE controller interfaces with a PPU for the electrospray thrusters by providing power and communication. Shown in Fig. 3,

the PPU receives 5V, 12V, and SPI lines from the controller. The controller also interfaces with pressure transducers, valves, heaters, thermocouples, and a pump to drive propellant flow, heat the chemical thruster catalyst bed, keep the system at a favorable temperature, and measure pressures and temperatures for telemetry. The microcontroller for the controller will perform all of the complex operations needed to run the propulsion system, so it must be carefully selected. The microcontroller must be able to interface with all of the peripheral connections and handle the operations required to run any firmware. Above all of that, the microcontroller must be able to handle the harsh space environments that SPECTRE will experience. Taking all of this into consideration, the ATmega128 was chosen due to the familiarity that the GLRG has with this processor and the heritage it has seen there. One thing to consider about the ATmega128 is that it does not have enough IO lines to satisfy all of the needs of this system, so a system of IO expansion must be considered.

The valves and heaters on the system are actuated through the use of PWM enabled low-side drivers. The currents passing through those drivers is collected as housekeeping data to determine which drivers are being operated. The pressure transducers are connected directly to the ATmega128 ADC. The valve driver ICs are directly operated by GPIO ports on the ATmega128, while the heaters, thermocouples, and current sensing devices are interfaced via a shift register on the ATmega128 SPI bus. The pump is connected directly to the ATmega128 SPI bus as is the PPU for the electrospray thrusters. Shift registers were chosen to interface the heaters, thermocouples and current sensing devices as they mimic the SPI architecture well. Reading and writing to a shift register is done in a way similar to how a SPI transaction is handled, so the already existing device firmware can be leveraged for use with the shift registers.

B. Controller Layout

Controller layout is just as important as the component selection. Two main design iterations were considered in the layout process to accommodate all of the elements required in building out the controller. The first consideration was to create a 3 4-layer board stack up with each board conforming to the CubeSat PCB standard of being a square with 96.5 mm^2 surface area. The second and final layout considered was to use one six-layer board that measured 125 mm $\times$ 84 mm.



(a) Peripheral Connections Board **(b) High Voltage Logic Board**

Fig. 6 Initial Board Stack-up PCB Layouts

Pictures of some of the boards comprising the first layout are shown in Fig. 6. The boards would interface with each other using spring loaded connectors that would pass through power and data signals. The three boards would separate out the laying out of interfacing connectors with the external peripherals, the higher voltage circuitry of the low-side PWM driver ICs and the pump, as well as the logic pertaining to operating the ATmega128 microcontroller. The boards were separated out this way to allow for common elements to be housed together and to best consolidate and cross-board signals to mitigate complications during the layout process. Four layer boards were chosen so that the power and ground planes could be separated from planes used for signal routing. That way, any signal interference could be moved away from affecting the common source of power and ground for the whole board.

The uppermost board in the 3 board stack-up contained the connector to interface with an upstream satellite FC, the ATmega128, memory expansion, pump control circuitry, and the power distribution logic. This subset of the functionality allows for most of the components that connect directly to the ATmega128 to live on the same board. The middle board of the stack housed the low-side PWM driver ICs to operate the valves and heaters and the pressure transducers. All of the sensors and actuators related to fluid operation were housed on the same board, and the IO expansion of those components also lived on that board. This way, a few signals could be sent from the ATmega128 to this board and they would get broken out to the specific target on this board alone. Finally, on the lower board were the connectors to the actual heaters and valves. All of the cabling would be restricted to that board and there would be enough space to comfortably connect all of the harnesses. As the routing is very compact, the four layer design choice was bumped up to six layers to get the same benefits of isolated ground and power planes with four additional planes for wire routing to allow for such compact component placement.

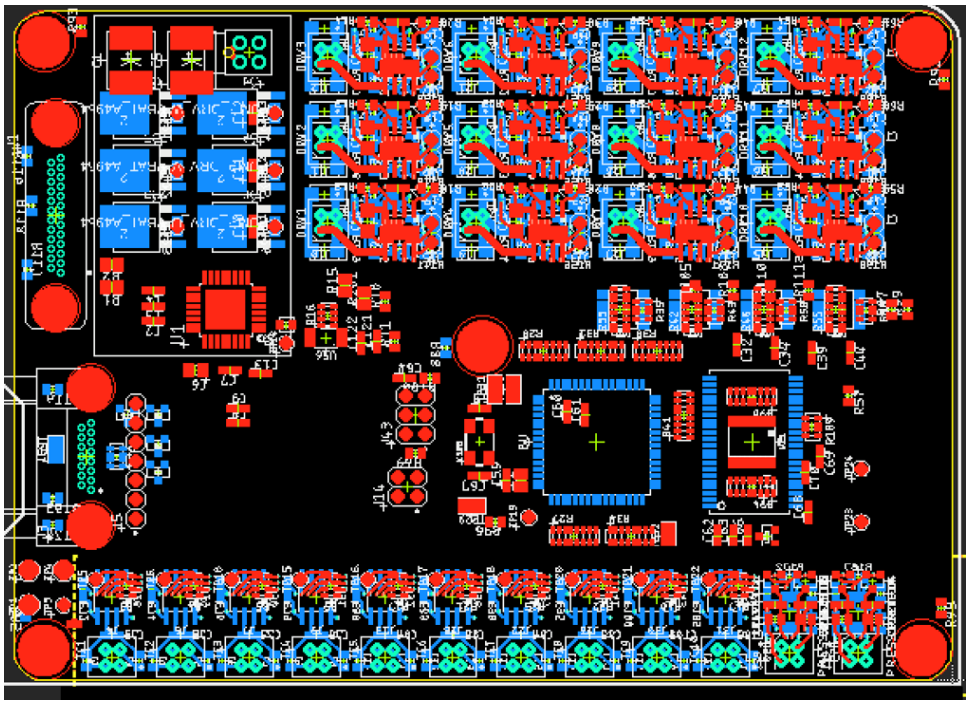


Fig. 7 Final SPECTRE board layout

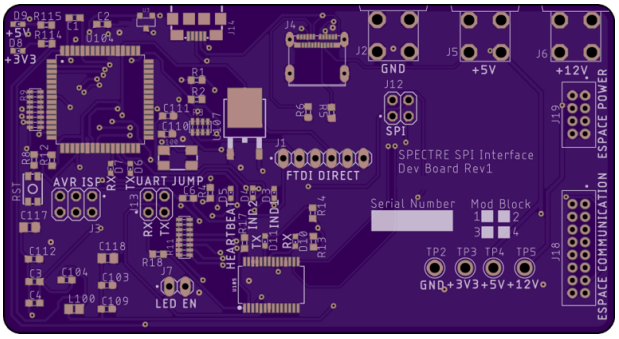
Above, in Fig. 7, the final decision to layout all of the components on a single board is shown. With this layout, the components would be a lot more compact, but any failures that may result due to faulty connector contact would be mitigated. The connectors to an upstream satellite FC and the external peripherals are all around the edge of the board, and the controller operation circuitry is located in the center of the board. The

testing of these individual boards also unearthed bugs that may have been present in the schematic or PCB layout of those components and feedback on those errors helped iterate on the controller design and layout.

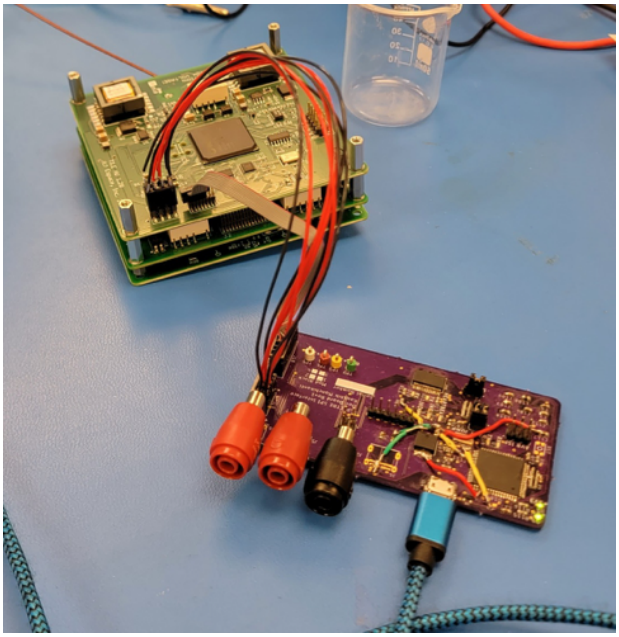
V. Controller Testing

A bottom-up approach was used to test the various components of the controller. Small interface and breakout PCBs were designed and manufactured to modularize the testing process. These smaller boards allow for interchangeable and local testing of individual components rather than having to initially implement the full functional software to test a few isolated components or systems[1]. Each new interface got a breakout board or interface board designed for testing, and they were designed to be plugged into breadboards for ease of use with an external microcontroller, such as a Teensy. This allowed for simple software development in a well-known and easy-to-use environment such as Arduino, while still being able to mirror the structure that would be used in the flight firmware for the controller. As there has been testing heritage for some of the features on the controller, not every interface got a test module PCB for functional validation.

A. PPU Interface Testing



(a) PPU Interface Board



(b) PPU Interface Testing Setup

The first big subsystem that was tested in this campaign was the connection between the controller and the PPU for the electrosprays. A variety of functionalities were promised by the developers of the PPU boards to be able to monitor and operate the electrospray system. This testing accomplished the validation of those command and data handling interfaces. The interface board used to mimic the connection from the Georgia Tech SPECTRE controller to the PPU is shown in Fig. 8a. The setup used to test the connection to the PPU is shown in Fig. 8b. A variety of the capabilities of the PPU were tested in an effort to verify this interface. The main features of interest that are needed to command the PPU in the flight firmware include CRC code generation, operation of thruster relays, and reading of relay currents. These operations correspond to ensuring a secure channel of communication, firing a thruster, and reading back the state of thruster operation. All of these features were validated during testing.

B. IC Unit Testing

Next in the testing campaign for the controller, the individual IC breakout boards were fabricated and populated. Once the PCBs were assembled, breadboards were prepared for each test. For each sensor, an Arduino Library was prepared to act as an intermediary software development to assist in the firmware development for the overall controller.

1. Shift Registers

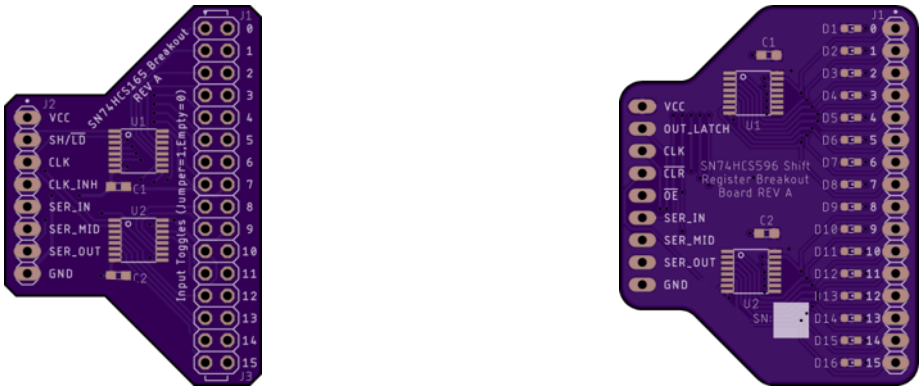


Fig. 9 SN74HCS165 Breakout Board (*l*) and SN74HCS596 Breakout Board (*r*)

The shift registers are especially important as they are the backbone of the extended IO SPI interface that connects the majority of the sensors and actuators to the ATmega128. The controller uses SN74HCS596 Parallel Output Shift Registers and SN74HCS165 Parallel Input Shift Register to manage the chip selection and sensor fault input connections to the ATmega128. The breakout boards for the shift registers are shown in Fig. 9. The functionality of the shift registers were first tested using a bit-bang approach to manually shift the data in and out of the shift registers and to the ATmega128. Once the chips were functionally verified, a SPI adaptation to the library was added to be able to leverage the structure of the SPI protocol in dealing with those shift registers. The functionality of the shift registers were verified and will be used in the validation of the SPI interface system on the controller.

2. INA239

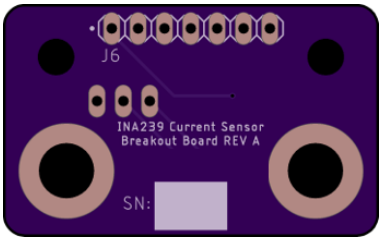


Fig. 10 INA239 Breakout Board

The INA239 is a current sensor IC which will allow for the controller to monitor the usage of the low-side PWM drivers for the valves and heaters. They will monitor the currents across the traces leading to the connectors for the valves and heaters themselves, and they will notify the controller whether the drivers are driving any valve or heater operation. This will allow for the controller to "double check" whether or not

a valve or heater is on or off after it commands a certain channel to activate or deactivate. Another useful feature that comes with the INA239 sensor is the detection of a current or voltage fault. This will provide another layer of protection against radiation effects.

The breakout board for the INA239 sensor is shown in Fig. 10. This breakout board allows for the INA239 to be easily connected to an external micro-controller on an interface such as a breadboard. An Arduino software library was developed to interface with the sensor and display current and voltage readings coming from the sensor. The testing of the sensor verified its working properly and that the data coming out of it is valid.

3. MAX31855/6

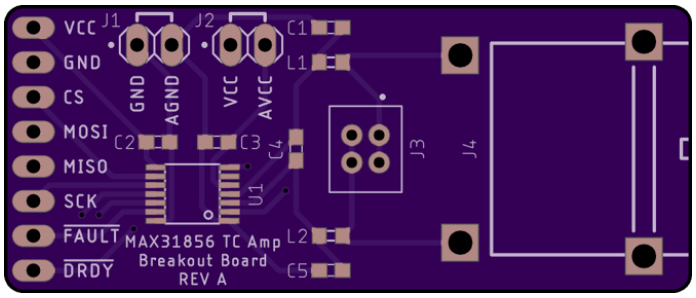
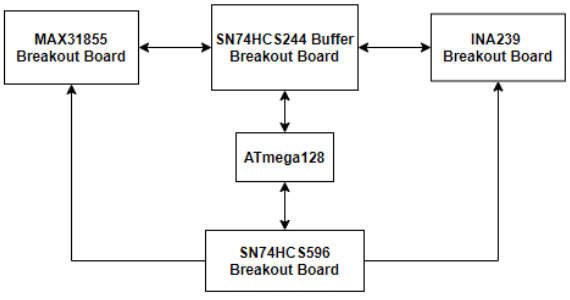


Fig. 11 MAX31855/6 Breakout Board

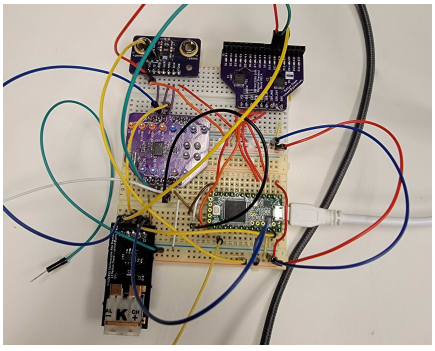
The MAX31855/6 families of thermocouple readers provide the interface between the thermocouples on SPECTRE and the controller. They allow for the reading of temperatures on the chemical thruster catalyst bed and various locations to get tank and propellant temperatures. These temperatures are valuable telemetry, so it is essential that the MAX31855 family of readers be validated and proven functional.

The breakout board to test the sensors is shown in Fig. 11. The breakout board lets the MAX chips be easily connected to an external microcontroller on a breadboard. Again, a software Arduino library was developed to get the readings from the chip once a thermocouple is connected. The testing verified that the MAX31855 family of thermocouple readers are functional and that the telemetry from those readers can be trusted during flight operation of the SPECTRE system.

C. SPI Interface Testing



(a) SPI Interface Testing Sensor Connections



(b) SPI Interface Testing Breadboard

Fig. 12 SPI Interface Testing Setup

The testing for the SPI interface is now the next step up from individual sensor unit testing. To test the forward propagation of selecting and reading from a sensor using the SPI Bus as present on the controller, multiple IC breakout boards were connected to the same Teensy on a breakout board. In particular, an SN74HCS596 Shift Register, an SN74HCS244 Octal Buffer, and both an INA239 current sensor and MAX31855KASA+ thermocouple reader were connected as shown in Fig. 12a on a breadboard to a Teensy to test the interface, the breadboard setup for this testing is shown in 12b.

During this testing, a bug with how the SPI communication was conducted for these ICs was found. The integration of multiple ICs did not mesh well with the software used on each individual IC and needed to be modified to perform the integrated functional test. This is good, as the purpose of this testing is to find and fix as many bugs as possible in software before performing a full controller-level functional test.

This integration testing works to iron out issues in software and hardware once more than one IC is being evaluated at the same time. The usage of the SPI hardware setup with shift registers and buffers allows for the formation of best practices to get smooth operations with the fully formed controller. The software can be fine-tuned to remove any unnecessary bloating or add any required additional elements. The hardware can also be verified to make sure that everything is being connected properly and assess any effects that may not have been foreseen during the initial design. These changes can then be propagated to the controller firmware and electrical design.

The results of the SPI interface testing confirm that software has been made to successfully read sensor data using the hardware setup present on the controller PCB. It also helps clear any issues that may have been present in the hardware layout of the controller.

D. Controller Functional Testing

```
Choose an operation to perform on the Controller:
------------------------------------------------------------
1.  Read values from an INA239 sensor
2.  Actuate a valve
3.  Fire an electrospray thruster
4.  Stop the firing of an electrospray thruster
5.  Read from the pressure transducers
6.  Read from the Thermocouples
7.  Read out Voltage BUS telemetry
8.  Perform an XMEM test
9.  Perform an RS-422 test
10. Read out from all of the Thermocouples and thermistors
11. Read out currents from all the valve drivers
12. Read out currents from all the heater drivers
13. Temperature control loop operations
14. Specify a valve/heater mask to read current data from
```

Fig. 13 Actions to test in Controller Functional Testing

The final segment of testing for the controller is a full functional test that tests the functionality of everything built together. This testing takes the software built for all of the unit and system testing and uses them to create a program that is run on the ATmega128 on the controller PCB to recreate the test results on the sensors as they are on the PCB and not on individual breakout boards. The controller functional testing will take place on a fabricated and assembled EDU of the controller, and will use an Arduino sketch that includes all of the software written thus far for controller testing. The functionality being tested on the EDU is shown in Fig. 13. As the testing for the controller progresses, any final bugs that appear within the fully integrated system will surface and be taken care of. After iterating through this process of testing and

fixing, the functional test software will be in a state to transform into the flight firmware that will be used on SPECTRE.

VI. Future Work

Some testing still remains to fully validate the use of the controller on SPECTRE for any missions that would use the propulsion system. So far, functional unit testing has been conducted using software that is more familiar to the testing team and not the flight code that will be flown on the system. So, the flight firmware still needs to be developed and tested on the system.

SPECTRE plans to use a JPL developed flight software framework called F'. This framework is based on an interrupt-driven architecture for running processes. The main advantage of F' is that it comes packaged with software to already handle command and data processing[2]. The infrastructure to parse and route commands, and collect and send out telemetry is baked into the framework and does not have to be developer implemented. The framework also handles the execution loop logic for all of the main firmware processes that will be defined in the software, so the developers will only be concerned with defining these processes and the rest of the boilerplate code is provided.

F' can be split into three main parts: ports, components, and topologies[2]. An F' software deployment consists of a combination of all three. Components are the processes that will run in the flight software and represent the basic separation of tasks within the firmware. In the use case of SPECTRE, this is how the different drivers for the components on the PCB will be defined. Additionally, there are built-in components to handle calling each process to run, so the data handling and transfer that is triggered by the running of each process happens in a set time-interval[2]. A process for each IC will be encapsulated by each "component". Next, the way each process communicates with other processes is defined by "ports". Ports define the behavior of calls made between components[2]. Ports let components only define small chunks of functionality that can be linked together, instead of large chunks of functionality that must be able to function independently. Lastly, topologies define the connections between components and ports that make a software deployment[2]. As this architecture is very different from how the software libraries that were used in the functional testing of the components so far, the flight firmware needs to be developed and tested to make sure that the software is ported correctly and can run pseudo-synchronously.

Outside of further software testing, environmental testing also needs to be performed on the controller and controller components.

As not all of the components on the controller PCB have mission heritage with the GLRG, radiation testing must be conducted to assess their performance during a space mission lifetime and verify that the electronics are space-grade[3]. Specifically, tests to assess the effects of SEL and TID need to be conducted[3]. The performance of the new ICs on this controller in regards to SEL are assessed by hitting the ICs with a beam of certain radiative intensity and then measuring their performance. The performance of the controller as a whole then needs to be assessed - a fully assembled PCB will be exposed to certain TID levels of radiation to simulate total space mission lifetimes and the performance of the controller will be compared to a baseline of no radiation exposed to see how resilient the controller will be in space.

Furthermore, mechanical environmental tests need to be performed on the controller, such as thermal vacuum and random vibration testing. The controller will be put into a thermal vacuum chamber and its performance at different temperatures in vacuum will be base-lined against performance at Earth temperatures and pressures to assess how the electronics of the board fare in extreme environments. Random vibration testing is needed to understand if the controller will withstand the environments of a launch vehicle as the satellite containing the SPECTRE system is being brought into Earth orbit. Overall, the environmental testing is done to ensure that the controller will perform in the face of the environments and disturbances it would be exposed to during the course of a space mission.

VII. Conclusion

The SPECTRE project is aimed toward advancing the capabilities of CubeSat propulsion. As a part of the proposed propulsion system, a controller is needed to operate the electronics and inteface the unit with a parent satellite. This paper summarizes the efforts taken to design, develop, and start testing such a controller. First, the design process was detailed from being given performance and system requirements to selecting the components necessary. Next, the proposed layouts with the rationale behind each choice and the final pick for the physical form factor of the controller electronics were shown. Finally, the testing plan and results are shown to highlight the steps taken and proposed to validate the controller performance during space mission operations. The next steps that should be taken to finish the validation of the controller performance are also talked about in this paper.

References

[1] Cheek, N., "Small Satellites SSC 21-IX-6 Development of a COTS-Based Propulsion System Controller for NASA's Lunar Flashlight CubeSat Mission," 2021.

[2] NASA, "F' Flight Software Embedded Systems Framework," `https://nasa.github.io/fprime/UsersGuide/guide.html`, 2020. Accessed: 2021-11-30.

[3] Sinclair, D., "Small Satellites SSC 13-IV-3 Radiation Effects and COTS Parts in SmallSats," 2013.