Technical Challenges in Liquid Rocket Engine Control System Design & Implementation

Christian Dove*, Barrett Twining†, Francis Clark‡, Henry Holmberg§, Katheryn Barger¶, Sam Bradley‖, and Ronald Rowe**
University of Alabama in Huntsville, Huntsville, AL, 35805, United States

Tartarus, the undergraduate student liquid rocket engine development project within the Space Hardware Club at UAH, has been working on a completely new, in-house built controls system for upcoming tests and firings. This new design, named Juno, far exceeded the previous system, but, due to a multitude of unforeseen challenges, including budgetary constraints, integration incompatibilities, and missed paragraphical needles in datasheet haystacks, the two have had to be hybridized for a time. The major impetus for switching the Juno system was to get away from the National Instruments data acquisition units (DAQs) which were integral to the old setup. These were run in LabVIEW, which limited the sensor throughput and shoehorned the project into an unreliable and confusing interface. Direct memory addressing (DMA) was the linchpin to getting the new system to work, as it sped up the data throughput immensely, though many difficulties were encountered as even single mistyped letters caused hours of debugging. Even ignoring the software difficulties, there are many challenges related to the hardware design. Thousands of connections need to be made, on the circuit board and in the system it connects to, and just one of them done wrong can set a design back months. When designing custom hardware, every component must be carefully chosen to ensure maximum compatibility and functionality. One mistake here could lead to problems ranging from a sensor not reading properly to the entire board being rendered inoperable as soon as it is plugged in. For a time the controls subteam was limited to four active members. While there are many more now, onboarding them in the middle of system development and implementation has been challenging, with differing levels of programming, electronics, rocketry, and controls knowledge, and limited time to teach members one-on-one. The challenges of liquid rocket engine controls systems are legion, but the solutions are out there.

I. Nomenclature

ADC	=	Analog to Digital Converter
CCI	=	Command Control and Instrumentation
DAQ	=	Data Acquisition unit
ETS	=	Engine Thrust Stand
GPIO	=	General Purpose Input and Output
GUI	=	Graphical User Interface
I2C	=	Inter-Integrated Circuit (Communication Protocol)
IC	=	Integrated Circuit
LC	=	Load Cell
MTS	=	Mobile Test Stand
MWE	=	Minimum Working Example
NI	=	National Instruments

*Student, Electrical & Computer Engineering Department, cd1816@uah.edu Student Member, 1823296.
†Student, Electrical & Computer Engineering Department, barrett.twining@uah.edu Student Member, 1811037.
‡Student, Electrical & Computer Engineering Department, francis.clark@uah.edu Student Member, 1823553.
§Student, Mechanical & Aerospace Engineering Department, hjh0022@uah.edu Student Member, 1811014.
¶Student, Mechanical & Aerospace Engineering Department, kcb0025@uah.edu Student Member, 1930867.
‖Student, Mechanical & Aerospace Engineering Department, sam.bradley@uah.edu Student Member, 1930858.
**Student, Mechanical & Aerospace Engineering Department, sam.bradley@uah.edu Student Member, 1930848.

P&ID = Piping and Instrumentation Diagram
PCB = Printed Circuit Board
PT = Pressure Transducer
QSPI = Quad-Serial Peripheral Interface (High-Speed Communication Protocol)
RAM = Random Access Memory
RPi4 = Raspberry Pi 4
SDIO = Secure Digital Input Output (High-Speed SD Card communication Protocol)
SPI = Serial Peripheral Interface (Communication Protocol)
SV = Solenoid Valve
TC = Thermocouple
UART = Universal Asynchronous Receiver/ Transmitter

II. Introduction

Tartarus is a student-led rocketry team focused on designing, manufacturing, and testing a bipropellant liquid rocket engine. The project is based at the University of Alabama in Huntsville and exists as a part of the school's Space Hardware Club. The team is made up of four subteams: Controls, Fluids, Propulsion, and Systems Engineering & Operations (SE&O). The Controls team deliberates the system's design for sensor data acquisition storage, valve control, and user interfacing with the rocket engine fluids handles the design, construction, and fluid management of the engine's test stand; propulsion handles the creation of the igniter and injector; SE&O deals with documentation, safety simulations, and media related activities. The overarching goal of this project is to teach students the necessary requirements and skills for designing and constructing a full scale liquid rocket engine.

Hot fire is the ultimate purpose for each individual rocket engine, as well as developing various designs for engine stands, injectors, and control system setups. Tartarus has undergone a series of overhauls since its first implementation, where the controls team has essentially resorted to the creation of an utterly from scratch system to replace the previously inefficient electronic setup. The implementation of these new ideas has been challenging, from both a design standpoint and a team standpoint. The team itself has seen the rapid loss and gain of members, so there have been setbacks in manufacturing output. Similar issues have arisen with the implementation and ordering of new, custom embedded systems that effectively replaces the preceding system. The team has done its best to adapt and overcome these challenges whilst retrofitting the design and development to exceed the capabilities of the former network.

III. Previous Design - SULLY

SULLY was the original controls system built for testing the engine, based around three National Instruments (NI) 6211 Data Acquisition units (DAQ), connected to a computer running a custom LabVIEW program. The digital outputs of the DAQs connected to a group of relay boards, powered by an external variable power supply, which actuated the solenoid valves (SV). Pressure was data was collected through pressure transducers (PT), connected to that same external supply and analog inputs on the DAQ. Temperature was read by T and K-type thermocouples (TC) whose signal was amplified and digitized by thermocouple amplifiers, connected over SPI to an Arduino Mega microcontroller. The microcontroller and DAQs communicated with the computer over USB. All other connectors were either screw terminals, as on the DAQs and TC amps, or 4-pin circular connectors [1] referred to from this point as "4-pin connectors". It is those 4-pin connectors where the problems begin. While their keyed design makes plugging them in the wrong way nigh impossible, the housings are fragile and difficult to solder. Pins falling out after repeated connections was a common occurrence. The DAQs, being controlled via LabVIEW, could not reach their full data logging potential. The enclosures, while sturdy, were unlabeled, with internal wiring often tangled and loose. These reasons, alongside wiring complexity, insufficient institutional knowledge, and a lack of expandability, led the team to a complete redesign.

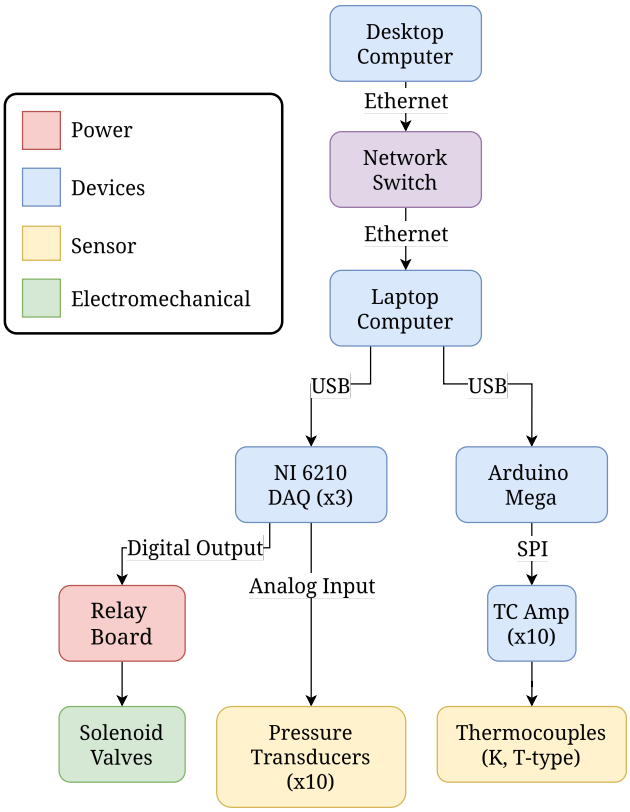

Fig. 1 SULLY Hardware Block Diagram.

IV. Improved Design - Juno

Requirements were laid out for the new design, named Juno, focused on reliability, expandability, all at a reasonable price. The system was required to:

- Support up to
  - 24 solenoid valves
  - 24 thermocouples
  - 24 pressure transducers
  - 3 load cells
- Store all data in multiple locations
- Have a user friendly, informative graphical user interface (GUI)
- Controllable remotely from >300 feet away
- Monitor all sensor values, report redlines
- Engage shutdown procedures automatically from sensor data
- Store and repeat multiple procedures for testing, involving precise timing
- Maintain maximum expandability and modularity

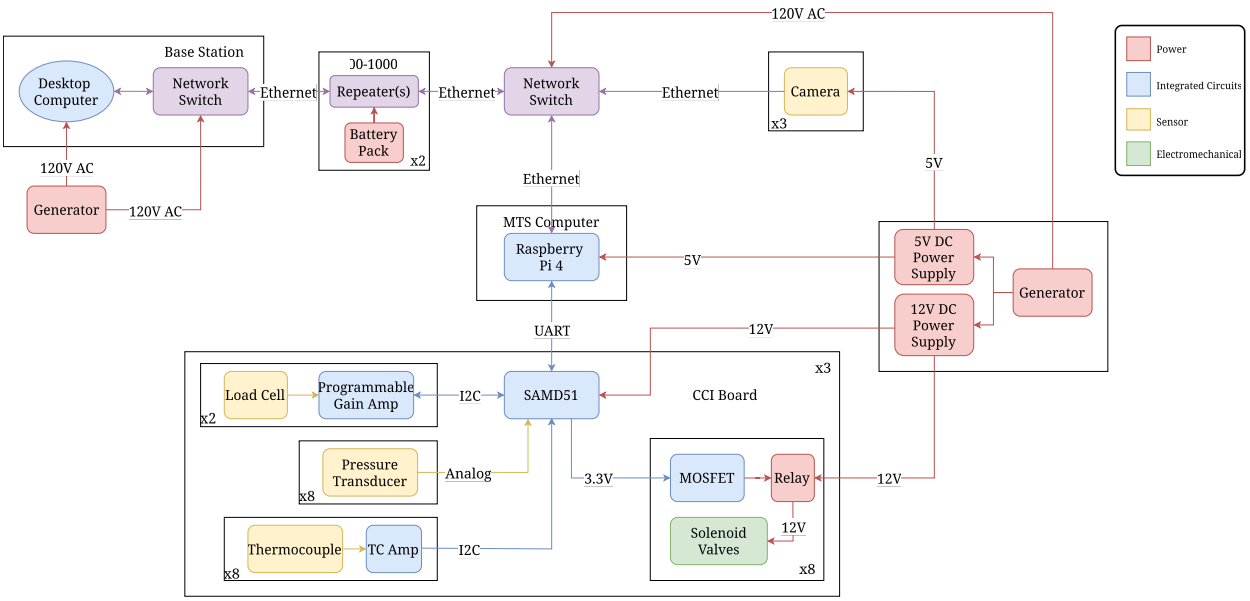

Fig. 2 Juno Hardware Block Diagram.

To purchase the modern equivalent of the system the team inherited, matching these new requirements, would be many thousands, if not tens of thousands of dollars, far exceeding the allocated budget. After much deliberation, it was decided a custom printed circuit board (PCB) would be designed, called a Command, Control, and Instrumentation (CCI) board. Three identical boards were planned, each with:

- 8 thermocouple amplifiers, 10 times faster than the previous
- 8 high precision inputs for pressure transducers, 72 times faster than previous
- 8 relays for solenoid valve actuation
- 2 load cell amplifiers

Functionality of the previous system was effectively miniaturized, while adding a host of new features.

CCI boards act as the last barrier to direct access of the valves and sensors. A fail-safe to errors that can appear, their reliability is derived from their deterministic loop timing, lack of OS scheduler, fixed interrupt priorities, and hardware based ADC sampling. This ensures consistent sampling, deterministic actuation latency, and reliable threshold monitoring. Overall, protecting against timing jitters originating in the supervisory layer. Powered independently from the GUI, they evaluate sensor data locally, maintain valve state locally, and sample locally. This transforms a system from a single point failure to having layered failure tolerance.

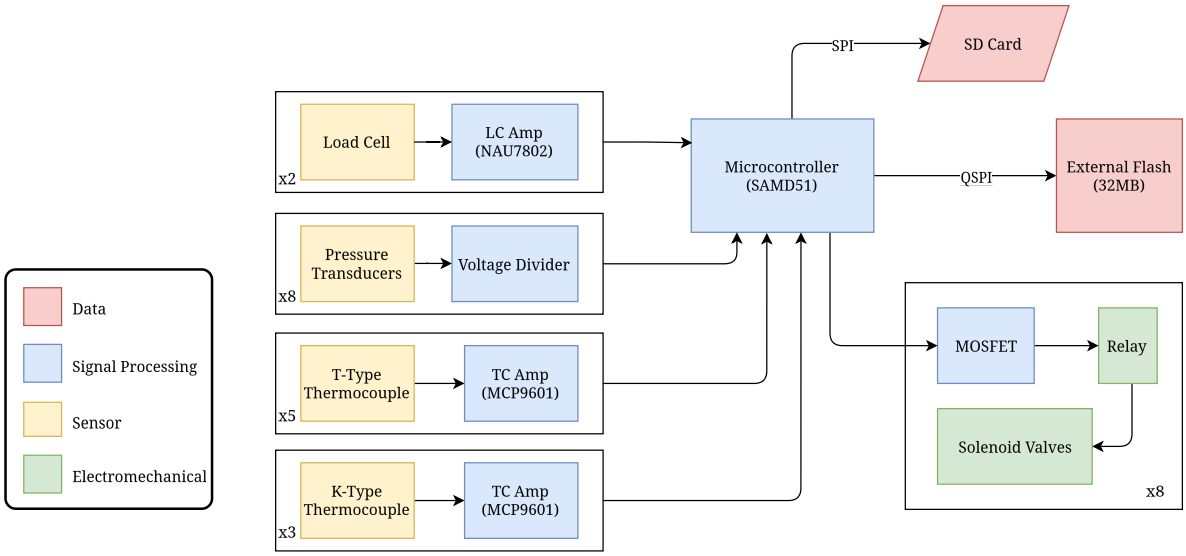

Fig. 3 CCI Board Hardware Block Diagram.

V. Testing Design - Europa

As testing approached, fluid systems finalized, engine parts machined, and procedures written, it was concluded that the CCI boards would not be reliable. Too many unknowns - ADC calibration, QSPI flash support, I2C to DMA configuration - necessitated changes be made to Juno. With a palpable irony and bubbling dread descending upon the team, the DAQs were dusted off and recommissioned for the newest and, at the time of writing, current system, Europa. Originally the hope was to switch the CCI boards with the DAQs, keeping the new connectors, wiring, and enclosures. This hope was quickly extinguished.

The Raspberry Pi 4 powered MTS computer, intended for controlling the CCI boards, was incompatible with the DAQs, as the required NI-DAQmx drivers did not exist for its ARM architecture. Replaced with the Intel NUC intended for the base station, it was discovered that while drivers did exist for the new computers architecture, they did not exist for the open-source Linux operating system and was replaced with the proprietary Windows 10. Feasibility research was conducted, and data was successfully retrieved from the DAQs using Python, as opposed to LabVIEW. The change in programming language ensured that the software would be developed faster and operable longer than the proprietary options, as experience in Python is much more common.

Once the thermocouple amplifiers from SULLY were counted, the team was frustrated to find that there would not be enough. However, the components for the CCI boards, which included 24 surface mount thermocouple amplifier chips, were already ordered and delivered. Through a somewhat precarious process, the amplifier circuitry from the now defunct CCI PCB was jumped and intercepted, allowing an external microcontroller, already having been programmed and used for testing, to read the temperature from any type of thermocouple. I2C addressing made connection simple, with only 4 wires needed for every 8 amplifiers. The MTS computer handled timing synchronization between the DAQs and microcontroller, reducing potential error margins in the data.

New quantity-distance (QD) calculations done by the propulsion and fluids subteams reduced the firing radius from >500 feet to 250. This change made Ethernet repeaters unnecessary, as the industry standard CAT-5 wire already possessed was rated for 100 meters (330 feet). The two inherited Dell PowerConnect 5324 network switches [2] were given new life, as their lack of power-over-Ethernet (PoE), needed for powering repeaters, was no longer a hindrance. Their bandwidth was more than enough to handle our limited data flow, even with the added strain of streaming from cameras near the stand.

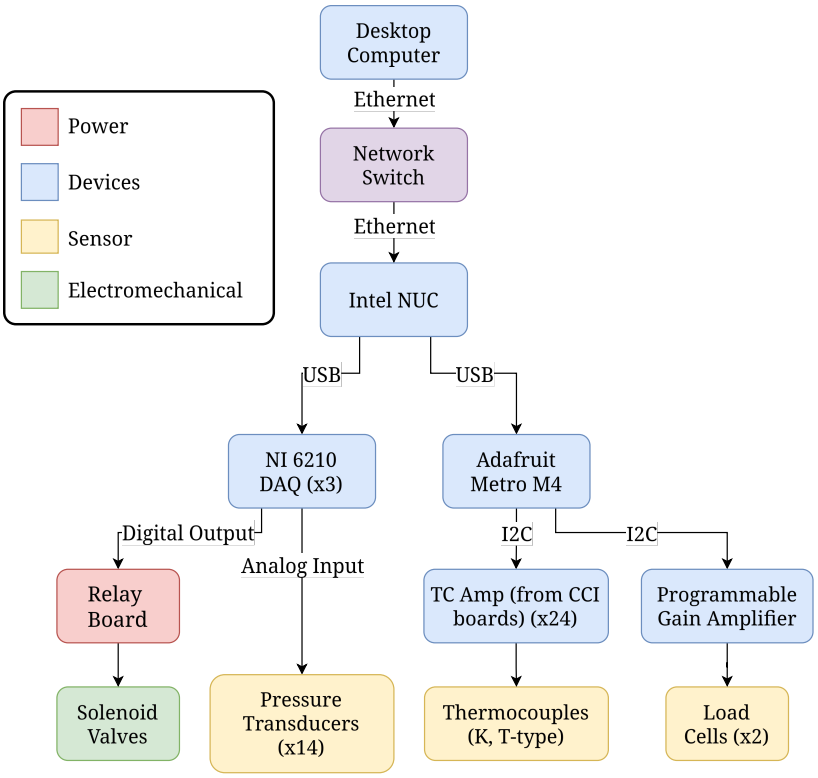

Fig. 4 Europa Hardware Block Diagram.

VI. Part Selection

In embedded system design one of the most critical aspects is component selection. A single missed specification in a datasheet, or one bad part can lead to hours of troubleshooting, damaged components, and hundreds to thousands of dollars in cost to redesign and reorder PCBs and other system components. This section outlines some of the most common pitfalls the team encountered, and how they should be avoided.

The first and most important selection was the microcontroller powering the CCI board. Originally the STM32H723 [3] family was chosen for it's high clock speed of up to 600 MHz, large RAM capacity of up to 2 MB, and extensive GPIO options. The team began reading through the almost 4000 page datasheet and acquiring a development board for testing. Once programming began it was quickly realized that although the microcontroller boasted impressive specification, its lack of good software documentation, example circuit implementations, and an inability to use the Arduino IDE to add new programs removed it from the running.

With this information processed, the team decided on the SAMD51P20A [4] microcontroller. This chip, while not having specifications quite as notable, proved itself to be a much better option. Software testing was quickly started, and circuit design alongside it. The Adafruit Grand Central M4 [5] development board based on said microcontroller met the requirements of the CCI boards, and its open source layout meant that many hours of design headaches and troubleshooting were avoided.

While some members of the team were comfortable with so-called "bare metal" programming, in which the programmer changes individual registers, increasing both control and complexity, it was decided that the software for the CCI boards would be written in Arduino IDE. This limited some finer control of the microcontroller, but drastically reduced debugging time. In addition, the IDE is in widespread use across the university, which made finding people who already had experience a much simpler task.

VII. Datasheets

When researching components, datasheets are the most important documents, as well as a common source of mistakes. A single missed sentence in a 2000 page datasheet can create any number of problems in a design, from

data corruption, power failures, to incorrect sensor readings. The team was haunted by this ghost of datasheet past, hidden deep inside the load cell amplifiers used on the CCI boards. Originally, a single NAU7802 IC [6] was used. This IC communicates over I2C and, importantly, does not have a way to change its address. This fixed address makes it impossible for the microcontroller to distinguish between multiple ICs, despite theoretically. When it was determined that it would be best to have 2 load cell amplifiers per CCI board, the schematic for the load cell amplifier circuit was duplicated, without checking the set-ability of said I2C address. Most of the other chips on the board the communicated through I2C had an ADDR pin, which could be pulled high or low to set the least significant bit (LSB), or most significant bit (MSB), respectively of the I2C address. This was later caught upon review of the datasheet when troubleshooting, requiring either a redesign or a reduction in capabilities.

When the minimum working CCI board failed to connect to the computer's USB port, it was discovered that the crystal oscillator, despite working on a previous design with a variant of the processor (different pin count), had an equivalent series resistance (ESR) too high for the microcontroller to drive it properly. Upon closer inspection, it was found that while the crystal used was within the datasheet specifications, it was above the range for the "standard" gain of the oscillator circuit. The bootloader being used was using the standard gain mode, and not the high gain mode. A software fix was attempted, but with no results. To begin limited software testing, it was necessary to bypass the crystal and inject the proper frequency manually. This was accomplished using a benchtop function generator set to output a 32.768 kHz sine wave.

A. ADC to DMA

Direct memory addressing (DMA) was always planned to be an integral part of the system. DMA controllers allow data from peripherals, like the analog to digital converter (ADC), to be directly routed to memory, increasing transfer speeds and saving on processing overhead. Data from the pressure transducers, which were connected internally to the ADC, without being processed would be stored in RAM. This information would be sent in chunks to the QSPI flash onboard the CCI board, which would then be sent to the MTS computer for processing. While pressure data in units of PSI would be displayed on the base station GUI, the CCI boards would store and send a number from 0 to 2^N, where N is the number of bits of precision from the ADC. In the case of high precision instruments like the in-house manufactured venturi-meter that would be the maximum of 16 bits, giving a range of 0 to 65535. This is converted to PSI using Eq. 1, where ADC is the raw digital value and N is the bits of precision.

$$\text{PSI} = 250\left(\frac{5 \cdot \text{ADC}}{2^N} - 1\right) \tag{1}$$

To pipe the data in such a way, direct register manipulation was required. Figure 7 below shows the complexity of configuring the direct memory address controller (DMAC) to read data from the two analog digital converters, each with 16 individual channels, sequentially. Though information on implementing this use case was difficult to find, eventually a supposed minimum working example (MWE) was written. Unfortunately, it did not work initially. After many days of troubleshooting, it was discovered that a single mistyped character caused the entirety of the system to malfunction, as one register was mistook for another, and the system was unable to understand what sequence to initiate next. Below is a snippet of said minimum working example. The code on the top produced unhelpful output, while the one on the bottom worked flawlessly.

```
// Set descriptor address
descriptor.descaddr = (uint32_t)&descriptor_section[0];
// DMAC configures what pin the ADC reads from
descriptor.srcaddr = (uint32_t)inputCtrlAdc + 4 * sizeof(uint32_t);
descriptor.dstaddr = (uint32_t)&ADC0->DSEQSTAT.reg;
descriptor.btcnt = 4;
descriptor.btctrl = DMAC_BTCTRL_BEATSIZE_WORD
    | DMAC_BTCTRL_SRCINC
    | DMAC_BTCTRL_VALID;
// Copy descriptor to descriptor section
memcpy(&descriptor_section[0], &descriptor, sizeof(dmacdescriptor));
```

Fig. 5 MWE of DMA Sequencing for ADC (broken).

```
// Set descriptor address
descriptor.descaddr = (uint32_t)&descriptor_section[0];
// DMAC configures what pin the ADC reads from
descriptor.srcaddr = (uint32_t)inputCtrlAdc + 4 * sizeof(uint32_t);
descriptor.dstaddr = (uint32_t)&ADC0->DSEQDATA.reg;
descriptor.btcnt = 4;
descriptor.btctrl = DMAC_BTCTRL_BEATSIZE_WORD
    | DMAC_BTCTRL_SRCINC
    | DMAC_BTCTRL_VALID;
// Copy descriptor to descriptor section
memcpy(&descriptor_section[0], &descriptor, sizeof(dmacdescriptor));
```

Fig. 6 MWE of DMA Sequencing for ADC (fixed).

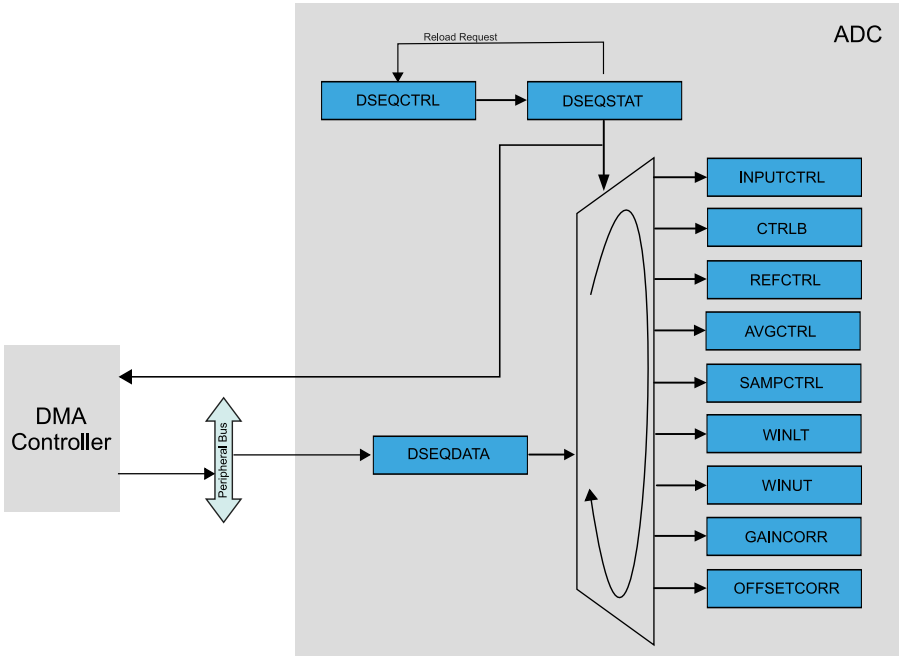

Fig. 7 DMA Sequencing.
[4, p. 1474]

VIII. Onboarding

One of Tartarus' main principles is to teach students how to design and build every facet of a liquid rocket engine and supporting test equipment. The controls team in particular seeks to enable students to learn the basics of complex electronic systems, gain experience with PCB design and manufacturing, software development, and much more. The speed at which new members are onboarded generally relies on the member's skill level going in to the project. The university's Space Hardware Club, on which Tartarus is a project, relies on an in-house, completely student run program designed to teach and enable students new to the field the basics of a variety of engineering principles, tools, and techniques. The program is called Two-Month, and, as the name implies, it takes place over two months. The first month students are placed into teams and participate in a variety of student-taught classes. They are able to learn about CAD, programming, circuit design, 3D printing, and more. The second month is spent designing and manufacturing to a given specification. This knowledge that many incoming members have drastically reduces the amount of time needed to teach these skills, especially as older members are often busy with their own tasks. On Tartarus, each incoming individual is allowed to choose what subteam they would like to work on, based on their interests, but are encouraged to try their

hand at many roles throughout the team. Some onboarding tasks for the controls subteam include, but are not limited to:

- Practicing soldering
- Making and testing connectors
- Using CAD to design component enclosures
- Debugging sensor software
- Coding using various microcontrollers (ie. Raspberry Pi Pico, Arduino)
- Familiarizing with test equipment (multimeters, oscilloscopes)

Seasoned members are available to assist in any of these pursuits or provide additional support when moments of confusion arise. While this has worked as an adequate method of teaching new members, the project itself is heavily focused on reaching hot fire. This means there is little time available for members to focus on onboarding so much as debugging and final construction of the system. New members are more likely to be assigned integral tasks to prepare for various system tests preceding hot fire. This is not necessarily a bad thing for them, as it can often be easier to learn skills through application, and under the pressure of a tangible due date. It is important to acknowledge that Tartarus is a small project, and the subteams are similarly small. New members are necessary for the survival of the project, but at a time where the entire project's focus is on the completion of an engine and supporting systems, each with incredible complexity and years of institutional knowledge behind the designs, it can be difficult to include them.

IX. Acknowledgments

The authors would first and foremost like to thank Dr. Gang Wang and Dr. Richard Tantaris for their mentorship to The University of Alabama in Huntsville's Space Hardware Club. We would also like to thank Mr. Lorenzo Coley for representing AIAA at The University of Alabama in Huntsville. We are also very grateful to the MAE department at UAH and Mr. Jon Buckley in particular for the use of their machine shop, and mentoring in manufacturing techniques. The Alabama Space Grant Consortium and the University of Alabama in Huntsville have graciously given us the resources necessary to fund this project through the Space Hardware Club. Thank you especially to AVCO for giving insightful technical feedback and providing cryogenic valves for the Modular Fluids System. Thank you to Swagelok for their generous donation and as well to MiTiGen for their in-kind contribution and support. Finally, we would like to thank all members of Tartarus for their continued support and dedication to the project.

References

[1] *182921-1 AMP Circular Power Connectors*, TE Connectivity, May 2025. URL https://www.te.com/en/product-182921-1.html.

[2] *Dell PowerConnect 5324 User Guide*, Dell Inc., September 2005. URL https://www.manualslib.com/manual/446027/Dell-Powerconnect-5324.html.

[3] *STM32H723 Family Datasheet*, ST Electronics, May 2025. URL https://www.st.com/resource/en/datasheet/stm32h723zg.pdf.

[4] *SAM D5x/E5x Family Data Sheet*, Microchip Inc., Dec. 2024. URL https://onlinedocs.microchip.com/oxy/GUID-F5813793-E016-46F5-A9E2-718D8BCED496-en-US-14/index.html, rev. M.

[5] *Adafruit Grand Central M4 Technical Details*, Adafruit Industries, 2025. URL https://www.adafruit.com/product/4064#technical-details.

[6] *NAU7802 Datasheet*, Nuvoton Inc., Feb. 2023. URL https://www.nuvoton.com/export/resource-files/en-us--DS_NAU7802_DataSheet_EN_Rev2.4.pdf, rev 2.4.