Ethernet for Modular SmallSat Payloads: One Year of Slingshot-1 Mission Operations

Alexander C. Utter

***Abstract*—Slingshot-1 is a 12U cubesat launched on 2022 July 2nd. Using a satellite bus from Blue Canyon Technologies and a modular payload interface developed by The Aerospace Corporation, the satellite carries a total of 19 research and development payloads. This document describes the Ethernet-based modular payload interface, pre-launch integration, and on-orbit operations. The modular interface has been a resounding success, enabling successful integration of all 19 payloads on a rapid schedule. Steps that previously took months or years were typically completed in a matter of weeks. During the first 15 months of mission operations, all payloads passed initial on-orbit functional checkout, and most have successfully completed their mission objectives.**

***Index Terms*—Cubesat, Modularity, Slingshot, SmallSat**

I. INTRODUCTION

THE Slingshot-1 mission was originally conceived as a research and development program to design, prototype, and test on-orbit modular and autonomous technology experiments for next generation satellite systems [1][2].

Historically, most satellites are tailored to a specific mission, with a bespoke bus, bespoke payloads, and bespoke interfaces between those subsystems. Bespoke engineering practices are effective in minimizing the total size, weight, and power (SWaP) required to meet mission objectives. However, production of bespoke designs requires considerable non-recurring engineering (NRE) activity that is costly and time-consuming.

One solution to this dilemma is the increased use of multipurpose modular subsystems. Multipurpose modules allow extensive design reuse between missions, which reduces costs and accelerates production timelines.

In theory, well-optimized tailor-made designs can always perform at least as well as reusable designs. A specialized design can be more aggressively optimized than a generic or reusable one. For example, power supplies tailored to a specific input voltage range can be made smaller and more efficient than supplies that must tolerate a wider range of inputs. Tailored designs do not carry excess capacity required for one design to meet the greatest requirement amongst several different missions, nor do they add SWaP for unused features required to conform to an industry standard.

In practice, however, budget and schedule constraints may not allow from-scratch redesign of complex systems for every mission. A well-optimized generic design may outperform a poorly optimized bespoke design.

Regardless, as smallsat customers demand reduced costs and faster timelines, the benefits of modularity increasingly outweigh its real or perceived drawbacks.

Modularity, as defined by the Slingshot model, covers technologies and architectures that make space systems more agile and adaptable. This involves open standards, interfaces, and plug-and-play systems to achieve production-rate agility, and technologies that support ground-to-space reconfigurability to adapt to changing environments.

The ideal form of the modular interface is completely forwards- and backwards-compatible. Publishing open specifications ensures that any vendor can design a compatible host or payload. Keeping the interface simple ensures the cost of doing so is minimized. A stable and thorough interface definition ensures that every payload is flight-ready for any future mission (e.g., Slingshot-1, Slingshot-2, etc.), regardless of the satellite bus, without requiring design modifications.

We hope that Slingshot fulfills at least some of these ideals.

II. MODULAR HARDWARE INTERFACE

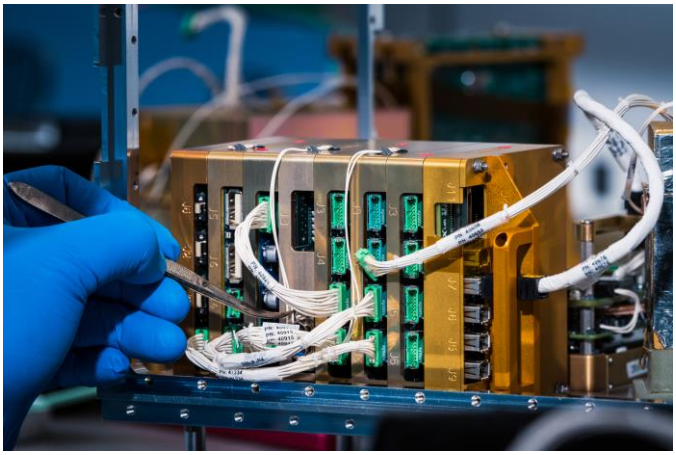

Figure 1: Handle interface unit showing both low-speed interfaces (green) and high-speed interfaces (silver, bottom-right)

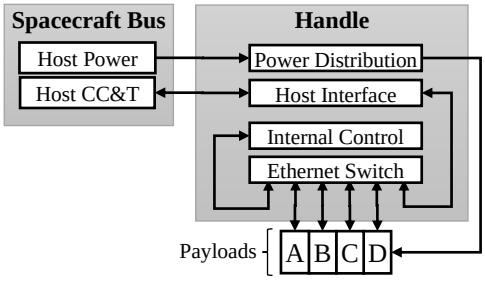

Figure 2: Handle connects to the spacecraft bus and to each payload.

The concept for the Slingshot modular interface began from an Aerospace Corporation internal survey of near-future cubesat missions [3]. Requirements identified by that survey included flexible connectivity between any two subsystems,

immunity to certain endpoint malfunctions, backwards compatibility with low-SWaP microcontrollers, scalability to gigabit rates, and power delivery up to 40W per payload.

That survey resulted in the development of a mixed-media Ethernet switch called "SatCat5". SatCat5 bridges the gap between microcontroller-friendly interfaces (I^2C, SPI, UART) and de-facto standards from the information-technology industry (10/100/1000BASE-T, RMII, RGMII, SGMII). Ports of all types are interconnected on the same Ethernet local area network (LAN), whether they operate at 100 kbps or 1,000 Mbps. SatCat5 has been released as free, open source gateware under the LGPLv3 license[1].

Ethernet (IEEE 802.3) is a ubiquitous standard in personal computer networking. Physical layer connectivity uses a star-topology with point-to-point links over a variety of media, operating at a variety of rates. The switch at the center of each star forwards packets between the individual ports, validating incoming packets and maintaining a separate transmission queue for each port. Outgoing packets may be serviced in the order received, or they may prioritize urgent traffic. Complex networks may have multiple interconnected switches.

The use of a switch has the added benefit of isolating many failures. Since every Ethernet frame ends in a checksum, the switch will simply reject any malformed frames. While excess congestion due to so-called "babbling idiot" failures remain a risk, this can be mitigated with rate limits or extensions such as time-triggered Ethernet [4].

Ethernet has many advantages for rapid payload development. An Ethernet LAN allows the use of Internet protocols such as UDP or TCP/IP, but it also allows the concurrent use of simpler, ad-hoc protocols for easy compatibility with resource-constrained microcontrollers. In either case, connecting the LAN to an ordinary personal computer allows developers to debug their payloads using powerful tools such as Wireshark, an open-source packet analyzer.

To meet the requirements for Slingshot-1, we developed a two-tiered interface specification. LAN connectivity for both tiers is provided by the SatCat5 Ethernet switch. Each "low-speed" port includes the following:

- Power supplies (3.3V, 5.0V, ~12V unregulated, and 22.0V) up to 2.0A each, with per-port power switches, current-monitoring, and current-limiting circuits.
- Ethernet port operating in SPI mode (up to 20 Mbps) or UART mode (up to 921.6 kbaud). Packets are delimited using SLIP (IETF-RFC-1055).
- Support interfaces including a resistance temperature detector (RTD), a GPS-disciplined pulse-per-second (PPS) signal, one discrete input, and one discrete output.

Each "high-speed" port provides power and data connectivity only, with the following interfaces:

- Power supply (22.0V only)
- Gigabit Ethernet (GbE) port using the Serial Gigabit Media Independent Interface (SGMII, Cisco ENG-46158).

Full details for each of the above including connector selection, pinouts, and preferred message formats is provided in the Slingshot Payload Manual [5].

To provide full independence from the host satellite bus, all the above services are provided by a separate "Handle" subsystem as shown in Figure 1 and Figure 2. All payloads connect directly to ports on Handle, which acts as the Slingshot host. Handle converts unregulated bus power to each of the provided supplies; provides power-switching and current-monitoring; operates the ancillary interfaces; hosts the SatCat5 Ethernet LAN; and connects the LAN to the host bus.

Handle insulates the satellite bus developers from changes to payloads. From the satellite integrator's perspective, the interface to Handle is the only one that needs to be developed and tested. Last minute changes to the manifest do not affect the required power, command, or telemetry interfaces.

Handle also insulates payload developers from changes to the satellite bus. If future missions change to a different bus provider, there is no change to the payload-facing interface. Handle may require minor updates, but its modular design ensures such updates are straightforward. For Slingshot-1, the interface to the Blue Canyon Technologies (BCT) XB1 bus [6] provides a CCSDS-encoded SpaceWire link and a 10-12V unregulated power supply, but such details are hidden from the individual payloads.

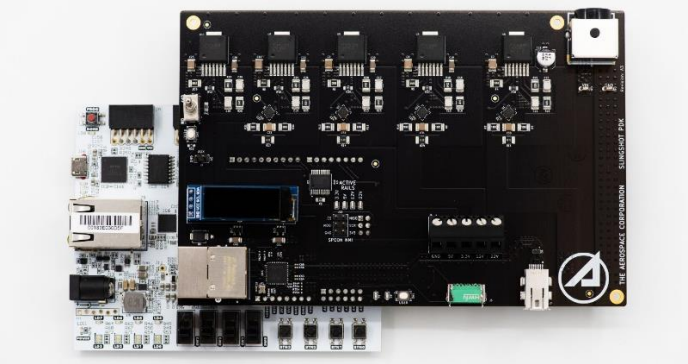

Figure 3: Slingshot Payload Development Kit (PDK)

For development purposes, we also designed and manufactured the Slingshot Payload Development Kit (PDK) as shown in Figure 3. The PDK provides the same Slingshot-compatible interface as Handle (i.e., one low-speed port and one high-speed port), but it connects directly to standard RJ-45 10/100/1000BASE-T Ethernet ports found on most PCs. The cost is low enough to allow every developer to have their own kit. A total of 50 of these low-cost PDKs were manufactured and distributed to individual payload developers. This allowed rapid, parallel development by all payload teams, working concurrently without the need to coordinate access to one-of-a-kind hardware assets.

III. Modular Software Interface

The primary objective of the modular software interface is to provide a uniform environment for operating each payload. This allows the same control scripts to be used regardless of

[1] https://github.com/the-aerospace-corporation/satcat5

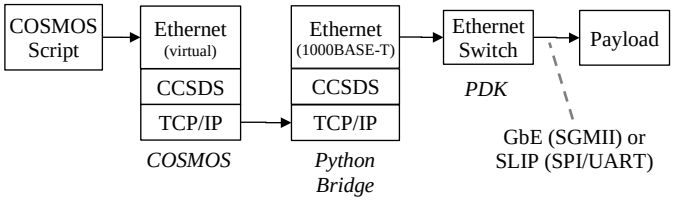

Figure 5: Protocol layers in the PDK configuration.

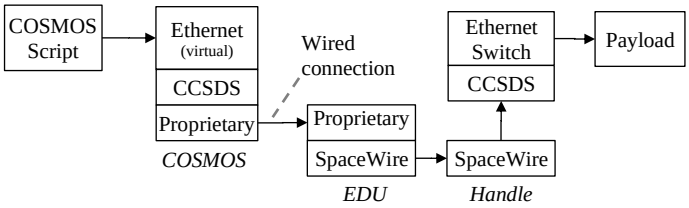

Figure 6: Protocol layers in the EDU configuration.

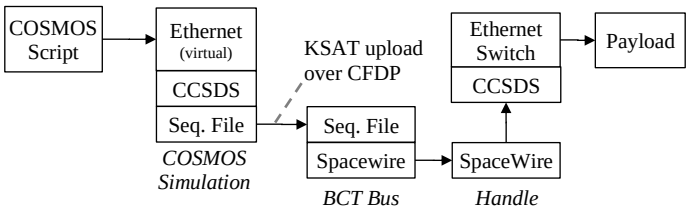

Figure 4: Protocol layers in the flight configuration.

how the payload is connected, whether the payload is in a simulation, on a lab bench, or even on-orbit.

Payload developers define their operational and test procedures using COSMOS, an open-source Ruby framework developed by Ball Aerospace. Use of COSMOS and the Slingshot applications programming interface (API) allows the same scripts to be run in all payload configurations. Various plugins connect from the COSMOS environment to the payload, providing specific connectivity required for PDK, EDU, or the spacecraft bus configuration.

The PDK and EDU environments allow real-time two-way communications, but this is not possible in-flight because communications with the ground are limited to approximately one ground contact per day. Therefore, most commands are executed on a delayed schedule, and resulting telemetry is stored by an onboard data-recorder for download during the next ground contact. To facilitate testing, the PDK environment supports modes that allow full two-way communication (for rapid debugging), modes that add significant latency (as an intermediate step towards operational mode), and modes that ignore telemetry completely (to best emulate on-orbit operations).

In all configurations, the COSMOS scripts communicate with the payload through CCSDS-wrapped Ethernet frames. However, the detailed path from COSMOS to the payload depends on the active configuration:

- In the PDK configuration (Figure 5): COSMOS connects to a custom Python "bridge" application. COSMOS provides an API over TCP/IP for precisely this purpose. The bridge removes the CCSDS headers and sends Ethernet packets to the PDK, and the PDK relays those packets to the payload over the selected interface (i.e., SLIP or SGMII).
- In the BCT engineering development unit (EDU, Figure 6): BCT software connects COSMOS to the EDU, which relays encapsulated packets over SpaceWire to Handle, which de-encapsulates the Ethernet packets for relay to the payload.
- In flight operations (Figure 4): a simulated COSMOS environment stores command "sequence files" (Section VI) that are uploaded to the spacecraft bus. At the designated time, the bus reads the sequence file and relays encapsulated packets over SpaceWire to Handle, which de-encapsulates the Ethernet packets for immediate relay to the payload.

Notably, the flight operations configuration does not use COSMOS per se. To meet the requirement for full CI/CD automation and reproducibility (see Section VI), payload procedures run in a simulation that mimics the COSMOS software environment. In PDK or EDU mode, the same functions are forwarded directly to the COSMOS API. In the simulation mode, they are forwarded to a from-scratch equivalent implementation. The simulation supports large portions of the COSMOS API, including the domain-specific language used for command and telemetry formatting.

IV. Spacecraft Bus

The Slingshot-1 spacecraft comprises a 12U XB1 bus manufactured by BCT, the Handle payload adapter, and nineteen payloads listed in Section X. The photo in Figure 7 has been processed to remove the background.

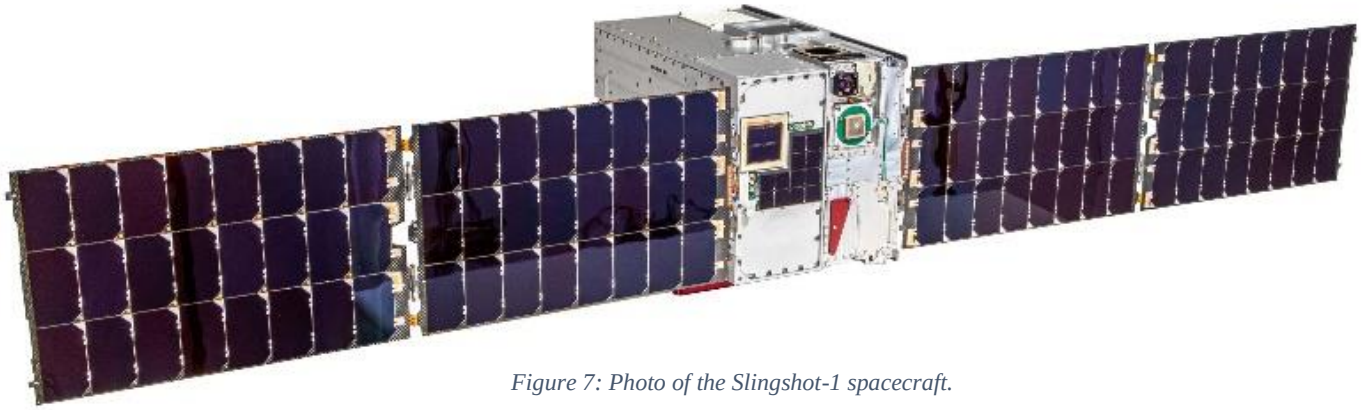

Figure 7: Photo of the Slingshot-1 spacecraft.

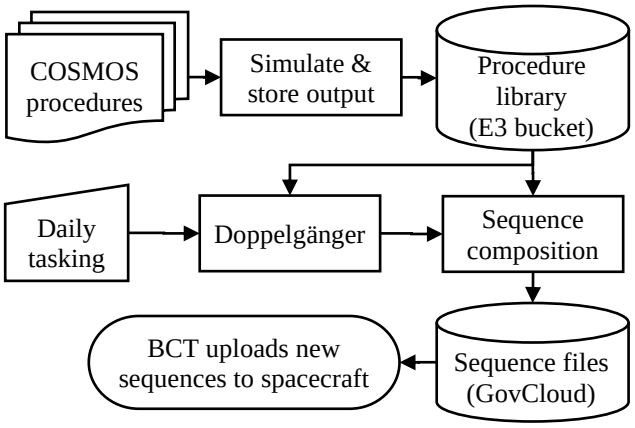

Figure 10: Command generation flowchart for translation of high-level intent to a sequence of low-level scheduled commands.

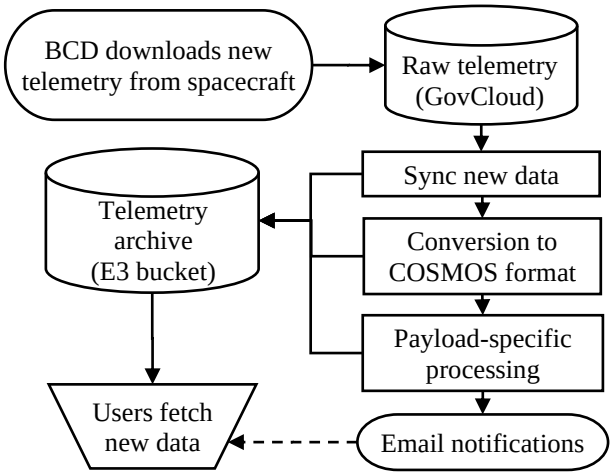

Figure 9: Telemetry download, conversion, and notification are handled by a scheduled process that polls for new data.

The XB1 bus includes the command and data handling system; the electrical power system; the guidance, navigation, and control system including two star-trackers; the S-band radio; and four deployable solar panels delivering up to 100W of solar power. The solar panels are body-fixed, requiring the spacecraft to track the sun by body-steering.

Power for Handle is provided through a single load switch, delivering unregulated power at approximately 10-12V. The data link from the bus to Handle is CCSDS-encoded SpaceWire operating at 20 Mbps in each direction. Additional interfaces include a GPS-disciplined pulse-per-second (PPS) signal and a small number of discrete general-purpose inputs and outputs (GPIO). System time is coordinated through a "time-at-tone" message that indicates the GPS time (i.e., week number and time-of-week) associated with each PPS rising edge. Full details are included in the "Handle to Spacecraft ICD" [6].

V. Ground Segment

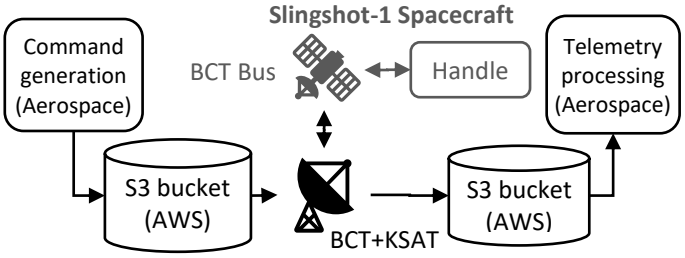

Figure 8: The Slingshot-1 ground segment. Aerospace uses AWS cloud-storage drop-boxes to relay commands to BCT; BCT uses KSAT to communicate with the spacecraft; BCT uses AWS to deliver the resulting telemetry back to Aerospace.

The ground segment shown in Figure 8 includes mission support operations spanning four companies:

- The Aerospace Corporation is responsible for payload operations. It decides which payload activities are performed each day, then generates command schedules to implement those activities. The resulting data is analyzed by the individual payload teams.
- Blue Canyon Technologies (BCT) is responsible for bus operations. It schedules ground contacts, performs routine maintenance, uploads command sequences, downloads stored telemetry, and resolves bus anomalies as needed.
- Amazon Web Services (AWS) provides the GovCloud S3 service used for easy exchange of data between Aerospace and BCT, as well as other cloud-computing services used by BCT to support their operations.
- Kongsberg Satellite Services (KSAT) operates the multinational network of ground stations used for S-band radio communications with the spacecraft.

VI. Payload Operations

Payload operations are divided into two steps: generation of command schedules for each day and processing of incoming telemetry.

Command schedules translate high-level intent into a sequence of low-level commands. Figure 10 shows a block-diagram for this process. Each "day" is the time from one ground contact to the next, which is typically about 24 hours. Operators specify a list of campaigns to be run on an upcoming day, e.g., "take a sequence of pictures the next time the spacecraft flies over Los Angeles". The payload operations software converts this into a sequence file, which is an itemized schedule of low-level bus and payload commands that should be executed at specific times.

Command generation starts from a library of COSMOS procedures stored on an E3 bucket[2]. Individual COSMOS scripts and supporting libraries are stored in a Git repository. Extensive use of Git and Bitbucket allows full historical traceability of all commands issued to the spacecraft and peer-review of changes. This workflow proved straightforward for most software developers, bringing clarity despite frequent revisions to payload procedures. However, it also proved complicated and unfamiliar for other staff, who required additional training.

An automated continuous-integration / continuous-deployment (CI/CD) process monitors the Git repository.

[2] The "E3-StorageGrid" is an S3-compatible storage service that is operated on-premises by The Aerospace Corporation.

Whenever changes are pushed, the CI/CD system runs a battery of tests and executes each COSMOS script in a simulated environment, storing the timing and contents of each command in both plaintext and binary format. If the changes are accepted through a pull request, then the results replace the stored procedures in the E3 bucket. The two-step process allows procedures to be defined through other methods including recording of manually-operated interactions, but this option is rarely used.

The next step is campaign definition. Each campaign defines a specific high-level operation, comprising a parameterizable list of bus commands, payload procedures, and constraints. For example, consider a campaign that takes a sunlit photo of a designated location. Parameters include the location to be photographed and appropriate camera settings. The photo campaign requires commands to point the camera by rotating the spacecraft (a bus command), power up the camera (a payload procedure), take a picture, download the picture, shut down the camera, and return the bus to its default sun-pointing mode. Each of these steps has a known duration, pulled from metadata in the stored procedure library. Constraints ensure that the location is within line-of-sight, the elevation angle is acceptable, the sun is up, and no other concurrent tasks are allowed to change bus orientation. The parameters, procedures, and constraints for each campaign are recorded in a version-controlled list.

The next step is schedule generation, which is performed by a digital twin called Doppelgänger. Operators specify daily tasking as a list of desired campaigns and their parameters. For each task in the schedule, Doppelgänger loads the requested campaign and attempts to find a timeslot that can meet all associated constraints. If a valid schedule can be assembled, it simulates operation of the spacecraft to perform additional safety checks.

As an example, consider the requirement to avoid over-depleting the spacecraft batteries. The power-budget simulation models solar panel generating capacity (which varies with time and orientation), expected power consumption of each payload, and battery charging and depletion. If the battery state-of-charge stays within established limits, then the planned activities are accepted. If not, then the plan is rejected, and users must submit a new plan with fewer battery-intensive tasks.

Doppelgänger is also responsible for other ancillary conversions, such as storing the pointing vector associated with a given antenna or camera. It also converts human-readable named locations into ECEF coordinates. Automating such conversions makes it much easier for operators to control the spacecraft and eliminates many common errors.

Daily tasking is stored in a separate Git repository with its own CI/CD system. Whenever changes are pushed, CI/CD runs Doppelgänger to see if a valid schedule can be generated. If so, it runs a composition tool that loads the Doppelgänger output to generate a series of timestamped low-level commands. The composition tool converts individual bus commands to their binary form, and it pulls complete payload procedures from the procedure library as needed. The resulting list of scheduled commands is stored in a binary format called a “sequence file”, which can be loaded and executed by software running on the BCT bus. The same sequence of commands is also stored in a plaintext .CSV file, to aid manual review in cases where it is needed.

The use of sequence files allows encapsulation of command formatting between Aerospace and BCT. The Aerospace COSMOS environment is complex, requiring multiple extensions and plugins to generate, encode, and decode payload commands. BCT was unwilling to push the required software to their production environment, due to compatibility concerns with ongoing operations for other customers. Instead, the BCT COSMOS environment treats the Aerospace commands as opaque binary blobs. Since BCT operators deliver each command to Handle regardless, they have no need to understand the detailed contents, eliminating the requirement to install additional software.

Sequence files are uploaded to the GovCloud drop-box. BCT checks the drop-box prior to each ground contact, then uploads any new or updated sequence files. Sequence files are moved to an “accept” or “reject” folder to indicate whether the files were processed successfully. During the contact, BCT also downloads stored telemetry from the onboard data-recorder and makes this information available in the same GovCloud drop-box.

Back at Aerospace, telemetry processing begins whenever new data is detected in the GovCloud drop-box. As shown in Figure 9, raw data is converted to the COSMOS format preferred for internal use. Some payload operators prefer to use this result directly; others perform additional automated steps. (For example, the conversion of packetized camera telemetry into .JPG and .PNG files for easy viewing.) All results are stored to the E3 bucket for easy retrieval, and relevant payload operators receive an email notification.

VII. Pre-Integration at Aerospace

The Aerospace Slingshot payloads were developed by a diverse group of departments within the company, many of whom had no prior flight hardware experience. This was an intentional strategy to propagate such expertise across the company and serve previously unserved spaceflight experimentation topics of interest to our customers. Principal investigators were challenged to address customer problems with a space-worthy experiment. A payload program manager kept track of schedule and budget, reporting to the core integration team.

A mechanical systems engineer at Aerospace managed the mechanical configuration of the 12U satellite. A dynamic list of eligible payloads was maintained throughout the project. Each potential payload answered a questionnaire with their needs and preferences, along with a description of their planned operations and any unusual requirements. The mechanical systems engineer selected mounting locations for each payload on one of five side panels of the spacecraft, leaving the sixth panel with no payloads for the purpose of access to the satellite interior. The flight payload selection was finalized prior to critical design review (CDR), and a computer-aided design (CAD) model for each panel was created and delivered to BCT for thermal analysis.

BCT maintained the complete Slingshot CAD model and incorporated the Aerospace CAD models for fidelity. BCT performed thermal and vibration analyses of the satellite.

Aerospace also provided a concept of operations with estimated payload power loads. The thermal analysis output was a temperature range that the panels would expect for the different load and orbit orientation scenarios. The vibration analysis of the entire Slingshot satellite resulted in derived random vibration levels for each panel.

Payload hardware and software were developed in parallel, with at least one PDK issued to each payload team starting in May 2020. By December 2020, each payload was ready to perform a "plug-in test" to validate electrical compatibility with the EDU and Handle. The plug-in test required payloads to submit COSMOS functional test scripts exercising all critical payload functions. In most cases, the plug-in test proceeded without errors, though a few payloads required minor updates due to small differences between PDK and EDU behavior. All payloads successfully completed the plug-in test by the end of January 2021.

As payloads were delivered, panels were assembled and harnessed. Some payloads were environmentally tested before handover because the principal investigators wanted assurance that the panel-level testing would not result in a failure. Others opted to proceed at-risk directly to panel-level testing. Each assembled flight panel was vibration-tested at its individual prescribed levels. One panel-level vibe test resulted in a catastrophic fracture of a mechanical support; this was remedied before repeating the test.

Once all panels were tested, they were assembled onto a mockup 12U bus frame shown in Figure 11. In this flight-like three-dimensional configuration, harnessing was completed, and that assembly experienced four TVAC cycles, operating Handle and all payloads in the simulated spaceflight environment per standard spaceflight test practices. Each payload was exercised using the same functional scripts provided for the plug-in tests. After TVAC and other closeout tasks, the entire mockup frame was packaged for shipment and delivered to BCT in July 2021.

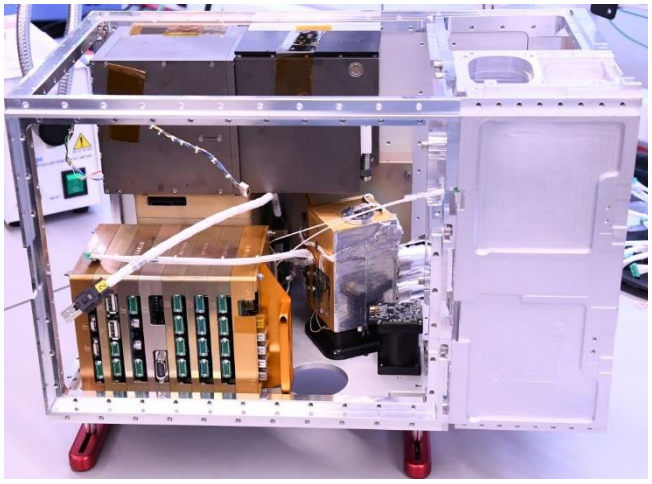

Figure 11: Mockup 12U bus frame with two installed payload panels.

VIII. Integration at BCT

At BCT, the flight panels were removed from the 12U mockup frame and reassembled onto the flight frame. Functional testing occurred continuously to minimize required rework if a problem was found. Integration was very quick, and only one harness was damaged and needed replacement. Because the Handle harnesses are standardized, a spare from another payload was utilized with minimum delay.

Once Slingshot was fully assembled, BCT applied their standard acceptance flight testing procedure to the complete satellite. The bus elements were functionally tested. Aerospace provided test scripts, operations documentation, and allowable limits for the payloads, allowing BCT staff to perform most payload acceptance tests without Aerospace engineers present, including acceptance vibration testing and TVAC cycles. For all tests, BCT returned captured telemetry to Aerospace for approval and archiving.

IX. Operations

Slingshot-1 operations began with the successful launch on 2022 July 2, as one of seven smallsats for the STP-S28A mission [7]. The launch vehicle was a "LauncherOne" rocket from Virgin Orbit.

BCT completed bus checkout quickly, and the first Aerospace-generated commands were uploaded to the vehicle on July 8. Initial bring-up of Handle was delayed by various compatibility problems in the Aerospace-to-BCT handoff. Unfortunately, no method was provided to test this handoff prior to on-orbit operations. Though all problems were resolved quickly through minor procedural changes, we recommend that future missions provide a test-like-you-fly environment for this handoff, to allow end-to-end testing prior to launch.

During this time, we also discovered[3] an on-orbit hardware anomaly, where one of the Handle FPGAs failed to boot at temperatures below -10°C. This anomaly was resolved by changes to the startup procedure, enabling FPGA power a few seconds before releasing it from reset, allowing "self-heating" to permit normal operation.

Handle bring-up was completed successfully on July 14. Once Handle was operational, initial checkout for remaining payloads proceeded quickly. For most payloads, the entire checkout procedure was to simply run the same functional tests previously established for acceptance testing.

However, a few payloads required additional intervention. For example, Camera and LaserCat initially failed to boot from extreme cold, requiring the addition of similar self-heating procedures. (Both shared significant design heritage with Handle's support circuitry for Xilinx 7-Series FPGAs, resulting in a common design flaw.) LaserCat bringup was further complicated by a procedural error that loaded the incorrect microcontroller firmware revision prior to launch, requiring additional workarounds.

For all payloads, the pace of troubleshooting was dictated by the once-per-day contact schedule. For typical trial-and-error diagnostics, feedback requires a 48-hour cycle[4]:

[3] The problem was initially observed during thermal-vacuum environmental testing, but root-cause was misidentified at the time, resulting in a quality-assurance escape.

[4] The same could be said of the tactical observe / orient / decide / act (OODA) cycle. Once-per-day contacts do not allow for tactical agility.

- Commands are generated a few hours in advance.
- Upload sequence file on first contact (T = 0).
- Experiment executed on schedule.
- Telemetry available on next contact (T = 24 hours).
- Generate revised commands and sequence file.
- Upload sequence file on next contact (T = 48 hours).

As payload operations began to normalize, the daily allotment for downlink of payload data became an ongoing problem. Once-per-day contacts set a time limit for all data downlinked from the spacecraft. By necessity, bus data required for continued spacecraft operations takes top priority. Depending on contact quality, this typically leaves about 100 MB/day available for all Slingshot payloads.

Unfortunately, Handle state-of-health telemetry inadvertently consumed a large portion of this allotment. Several Handle subsystems generate routine status reports containing voltage, current, temperature, packet counters, etc. In total, such reports average to approximately 7 kbps whenever Handle is powered. This modest rate sounds innocuous but accumulates to nearly 80 MB over 24 hours. These reports are essential for certain diagnostics but generally ignored. In hindsight, all routine telemetry should have included runtime-adjustable options to enable, disable, or adjust the rate to prevent this misallocation of scarce resources.

By mid-December, all payloads had completed their functional checkout successfully.

Ground operations software continued to iterate with accumulated experience. Improvements included new procedures to ensure each day starts from a known-good state, graceful recovery from various telemetry-parsing errors, conversion tools to streamline verification testing of new procedures on the EDU, and various features added to support unanticipated payload experiments. Automation ensured operations could continue with no dedicated full-time staff.

Handle has proved to be extremely reliable. After initial checkout was completed, the only known malfunction occurred on 2023 September 1. During operation of the SDR 2.0 payload, there was an unexpected bus reset, resulting in a sudden loss of power. This transient event overloaded a MOSFET inside Handle, resulting in a permanent failure of the 22V power rail for that payload. Future revisions will update the Handle design to prevent this failure mode.

The XB1 bus performed well, though it was prone to occasional anomalies. Approximately once per month, an anomaly would cause the bus to enter safe mode, aborting any remaining payload activities until the next contact. Most such anomalies were consistent with radiation-induced single event upset (SEU) and recovered completely within one day.

However, there was one bus malfunction with significant impact. In early June 2023, the XB1's onboard GPS receiver lost all functionality. Despite several attempts to recover, GPS functionality was never restored, and root cause was never identified. Most Slingshot payloads were unaffected, but without GPS, the bus could no longer point accurately enough to allow the LaserComm payload to maintain contact.

A workaround was developed, leveraging the Blinker payload and the flexibility of the Slingshot payload LAN. The Blinker payload contains a GPS receiver; the workaround updated Blinker to relay those GPS fixes through Handle to the XB1 bus, attempting to restore sufficient clock and position accuracy for LaserComm operation. Thanks to the flexibility of the Slingshot interface, such changes were possible with software patches, even though the original design never anticipated that such a need could arise.

Unfortunately, this workaround was never successfully applied. Aerospace patched Blinker to send the required data by the end of July. BCT patched the bus to detect and parse those messages in mid-August, but struggled with debugging until the problem was finally solved in mid-October. At time of writing, the workaround was never successfully applied to restore LaserComm operation.

X. PAYLOADS

This section lists each of the individual payloads and gives a brief overview of the objectives and outcome for each one.

ACETaTE (Aerospace C-band Experimental Transmitter and Technology Enabler) transmits an experimental navigation signal. The overall goals of ACETaTE are to assert use of C-band spectrum for navigation, validate the LEO-to-ground link budget, characterize GNSS acquisition and tracking performance, demonstrate navigation through pseudo-side-channel information, and demonstrate system viability [8].

Once activated, ACETaTE transmits a signal that is then recorded by a receiver on the ground. Recordings of more than 50 passes were gathered in each operating mode, measuring statistics on received signal power in both fair and adverse weather. The full experiment report is available in TOR-2023-02093 [9].

Blinker is a location-tracking transponder for space traffic management. By integrating a radio, GPS receiver, battery, and solar panels into a compact self-contained unit, it can be attached to any spacecraft to allow precise orbit determination even if the host is inoperable. It also contains LEDs that can flash at programmable cadence, for use in initial identification of unknown spacecraft by optical telescope.

The Blinker payload completed all its objectives [10], including orbit determination using GPS and optical tracking of the LEDs. Its GPS receiver also supported orbit-determination for measuring HyPer performance (see below). As a self-contained unit, Blinker was instrumental in diagnosing certain bus anomalies, since it continued to operate and provide independent telemetry. It was also a vital part of the attempted lost-GPS workaround discussed in Section IX.

CoralReef is a machine-learning experiment based on Google's Coral Tensor Processing Unit (TPU). The TPU is a low-power platform for hardware-accelerated inference, and successfully demonstrated normal operation as well as robust procedures for on-orbit software updates.

ECLIPSE-RR is an instrument developed by the U.S. Naval Research Laboratory (NRL). ECLIPSE-RR consists of a low-SWAP ultraviolet remote-sensing instrument called the "Triple Magnesium Ionospheric Photometer" (Tri-MIP), designed to observe emissions from magnesium ions in the atmosphere. The NRL-built Tri-MIP sensor, funded by DARPA, was coupled with a 1U scan mirror, and included on Slingshot-1 as a risk-reduction effort for the "Experiment for Characterizing the Lower Ionosphere & Prediction of Sporadic-E" (ECLIPSE) instrument (now flying on the International Space Station as

part of STP-H9). ECLIPSE-RR was an overwhelming success, collecting hundreds of datasets in both staring and limb-scan mode [11].

ExoRomper is a machine vision testbed for satellite pose estimation, incorporating a deployable model of a satellite, visible and infrared cameras, and a Coral TPU. The TPU poses the model, captures snapshots with both cameras, and applies machine vision to estimate the model pose. All steps were demonstrated successfully, with more than 1,000 images downloaded to support future training.

Handle provides power and data connectivity for all other payloads. Its primary goal was to demonstrate the Slingshot modular interface, which has been a total success. During flight operations, the only black mark has been the 22V power anomaly discussed in Section IX.

HyPer is a hydrogen peroxide thruster. The thruster completed more than 30 test campaigns, varying tank temperature and valve timing to fully characterize its operation. After each campaign, the change in velocity was measured by detailed orbit-determination using telemetry from Blinker. Tests continued until all propellant was exhausted in June 2023.

KeySpace is a cryptographic system based on the Viasat ES-1850, a certified NSA Type-1 cryptographic module. Its demonstration mission was completed with successful functional checkout in July 2022. This test was repeated throughout the mission with no change in functionality.

LaserComm is a free-space optical communication terminal supporting high-bandwidth downlink from space to ground. Initial checkout of the electronics was completed in December 2022. Alignment calibration was delayed by inclement weather in El Segundo but completed February 2023. Experiments continued using pseudorandom bit sequences (PRBS) through June 2023, showing excellent link quality up to the maximum tested rate of 250 Mbps. Further testing was precluded by the sudden XB1 GPS failure.

NTE is a government-furnished payload consisting of a retroreflective antenna. Intended to demonstrate enhanced radar detectability, it is entirely passive and requires no supporting electronics. Unfortunately, difficulties in securing a suitable ground-based radar meant this payload was never used.

ROESA is a massless payload demonstrating the use of the Constrained Applications Protocol (CoAP) to interconnect Slingshot payloads. CoAP is a lightweight communications protocol typically used for "Internet of Things" applications. In this case, data collected by Vertigo was sent to t.Spoon Processor for analysis that could not be performed by the Vertigo microcontroller. The system was successfully demonstrated in October 2022, allowing power-spectral analysis for onboard estimation of transfer functions for system identification and attitude control.

SatCat5 is the Ethernet LAN that connects all other payloads, hosted as part of Handle. SatCat5 has been operating without incident since its initial bringup.

SDR 2.0 (Software Defined Radio) is a CCSDS- compatible radio that can forward packets directly to and from the payload LAN. This allows it to act as a backup radio, augment uplink or downlink capacity, or simply allow direct connectivity independent of BCT. Unfortunately, licensing issues delayed testing until August 2023. Testing was initially successful but cut short by Handle's 22V power anomaly, which left SDR 2.0 unable to transmit.

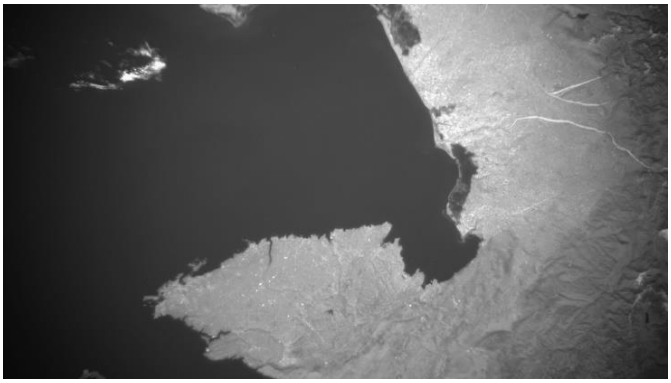

Figure 12: Venice, Italy on 2023 June 26.

Figure 13: Los Angeles, California on 2022 August 12.

STarfish is a low-power microcontroller with TrustZone security extensions. It acts as a cryptographic random-number generator supporting the KeySpace payload and allows collection of radiation upset statistics for the ArmV8-M microcontroller. Both objectives were successful, including the observation of at least one single-event latchup (SEL).

Starshield is a cybersecurity payload running intrusion-detection algorithms on the XB1 and on t.Spoon processor.

t.Spoon is a mechanical framework for modular hardware interfaces, allowing rapid assembly or replacement of compatible electronic modules. STarfish, t.Spoon Camera, and t.Spoon Processor all conform to this interface. Objectives included a timed board-swap operation, which was completed successfully during payload integration.

t.Spoon Camera is an FPGA-based camera payload for context imagery and star-tracking. It originally included three cameras, but one failed late in vehicle integration and could not be replaced. The remaining two cameras collected imagery throughout mission operations; a few examples are shown in Figure 12 and Figure 13.

Notably, t.Spoon Camera also supported another research project titled "Star Tracker Image-Derived Data Exploitation" (STRIDE). STRIDE uses telemetry from star-tracker cameras to detect other satellites for space traffic management. Initial demonstrations of the technology used sparse diagnostic data from archived AeroCube telemetry. With Slingshot, STRIDE was able to predict opportunities and collect much more data, including uncompressed full-frame imagery. This resulted in several confirmed detections of other spacecraft, using a payload that was not originally designed for this purpose.

t.Spoon Processor is a reconfigurable, network-connected computing platform supporting several other payloads. It is based on a Xilinx Zynq Ultrascale+ system-on-chip with a dual-core ARM CPU, FPGA fabric for hardware acceleration of intensive tasks, and gigabytes of attached RAM and flash storage. It allows other payloads to offload compute-intensive tasks, using containerized software to allow reconfiguration on demand. Example usage includes ROESA and Starshield.

Vertigo is a reconfigurable attitude control payload [12]. It operates independently of the attitude control system provided by the XB1 bus, with its own reaction wheels, magnetometers, and rate gyros. In conjunction with ROESA, it performs system identification on-orbit to allow automatic controller synthesis. This allows late-breaking changes to satellite configuration that would otherwise require time-consuming testing on the ground, and it can prevent errors like those that led to the catastrophic loss of the Hitomi X-ray observatory [13].

XI. Lessons Learned

Slingshot was an ambitious project, leading to many valuable lessons learned.

Most of the payloads were developed by different organizations. Therefore, a unified and mature approach to circuit design for reliability could not be assured. Given this reality, Handle and its experienced design team were essential in providing upstream power protection for various fault conditions, including radiation-induced single event latchup.

Ironically, Handle itself suffered permanent damage due to an unexpected power transient. It is vitally important to consider all possible conditions for power-protection circuitry. Disorderly shutdown events can produce unexpected sequencing conditions among sources and loads.

The modular interface was quite successful, especially in conjunction with the PDK. Careful attention to maintaining consistent behavior between the PDK and the EDU ensured most payloads completed plug-in testing without incident. Though this process did reveal some unanticipated differences, the modular interface ensured such problems were minor and readily fixed, allowing integration testing of 19 separate payloads to be completed in less than six weeks.

The connector for Handle's high-speed ports was selected after considering many factors including size, signal integrity, mechanical robustness, lead time, and unit cost. Unfortunately, that selection process did not anticipate the substantial additional volume required for strain relief of the cable assembly, which protruded several inches. This unplanned protrusion severely complicated mechanical layout of the various payloads. For that reason alone, this connector may be replaced in future missions.

Handle's low-speed connector was much more successful, though not without problems. Its locking feature sometimes required manual manipulation to engage, and in one case a loose pin walked out of the connector. Future missions will use a different locking feature when practical.

The excessive amount of Handle state-of-health telemetry was a significant oversight. All routine telemetry should include rate controls selectable at runtime, to allow mitigation of such problems without compromising diagnostic capability.

The telemetry overruns were exacerbated by the selected downlink strategy. By default, each contact automatically started by requesting the last 24 hours of bus and payload telemetry. Downlink proceeded in first-in, first-out order; any data still queued when the contact ended is simply dropped.

This strategy fails when the amount of bus and payload telemetry exceeds the downlink capacity. Because each pass is different in duration and quality, downlink capacity is variable. Furthermore, the bus and payload telemetry rates also vary depending on payload operations. The result of these variations is that some telemetry records are trapped on the satellite and never retrieved, while other contacts have idle time that is never used. A solution is more aggressive downlink management, better prioritization of downlink capacity, prompt feedback from payloads confirming whether they received expected telemetry, and improved automation for opportunistically retrieving truncated data.

Initial on-orbit operations were complicated by incomplete testing of the Aerospace-to-BCT handoff through the S3 bucket. Miscommunication regarding this process resulted in unpredictable inconsistencies in the uploaded command sequences, complicating the diagnosis of other problems discussed in Section IX. Though these problems were resolved within a few days, the process could have been accelerated by providing a cloud environment to test the end-to-end process prior to launch.

Maximum power for the modular interface is a known concern for future missions. Handle can deliver up to 40W per payload, which exceeds the requirements of all Slingshot-1 payloads with ample margin. Nevertheless, this limit has been a pain point for scaling to larger payloads, such as synthetic aperture radar payloads requiring 300W or more. Keeping separate power and data connectors may have resulted in better modular scalability.

Acknowledgment

The author thanks Dan Mabry, who was instrumental to Handle and Slingshot but retired before creation of this document; and Slingshot project manager Hannah Weiher for herding these many payloads to success. The author also thanks Blue Canyon Technologies, Space Test Program, Virgin Orbit, and countless individuals across The Aerospace Corporation for making this mission possible.

References

[1] H. Weiher, D. Mabry, and A. Utter. "Slingshot: In-space modularity test platform," *AIAA SCITECH 2022 Forum* (2022).

[2] "Slingshot platform to showcase advantages of modular payload architecture" *The Aerospace Corporation press release* (2022). https://aerospace.org/article/slingshot-platform-showcase-advantages-modular-payload-architecture

[3] A. Utter, et al. “SatCat5: A low-power, mixed-media Ethernet network for smallsats,” *34th Annual Small Satellite Conference* (2020).
[4] O. Daniel and O. Roman. “Fault injection framework for assessing fault containment of TTEthernet against babbling idiot failures,” *IEEE/ACM 26th International Symposium on Quality of Service* (2018).
[5] D. Mabry, A. Utter, and H. Weiher. “Slingshot payload manual,” *The Aerospace Corporation Report Number ATR-2022-12070* (2022). https://github.com/the-aerospace-corporation/satcat5/blob/main/examples/slingshot/ATR-2022-01270 - Slingshot Payload Manual.pdf
[6] D. Mabry, A. Lin, and A. Utter. “Handle to Spacecraft interface control document,” *The Aerospace Corporation Report Number ATR-2022-00635* (2022). https://github.com/the-aerospace-corporation/satcat5/blob/main/examples/slingshot/ATR-2022-00635 - Handle to Spacecraft Interface Control Document.pdf
[7] J. Faust. “Virgin Orbit launches Space Force mission,” *SpaceNews* (2022). https://spacenews.com/virgin-orbit-launches-space-force-mission/
[8] A. Arredondo, et al. “Very Inexpensive Navigation Enhancement Layer Reference Design.” *The Aerospace Corporation Report Number TOR-2020-01102* (2020).
[9] V. Petrosyan, et al. “Aerospace C-Band experimental transmitter and technology enabler (ACETaTE)” *The Aerospace Corporation Report Number TOR-2023-02093* (2023).
[10] “Blinker payload advances space traffic management capabilities.” *The Aerospace Corporation press release* (2023). https://aerospace.org/article/blinker-payload-advances-space-traffic-management-capabilities
[11] B. Fritz, et al. “First-Year Results from a Space-based Sporadic-E Detector.” *Ionospheric Effects Symposium* (2023).
[12] H. Weiher, et al. “Automated system identification for satellite attitude control.” *34th Annual Small Satellite Conference* (2020).
[13] A. Witze. “Software error doomed Japanese Hitomi spacecraft.” *Nature 533.7601* (2016).