AEROSPACE REPORT NO.
ATR-2022-01270

Slingshot Payload Manual

May 31, 2022

Dan J. Mabry[1], Alexander C. Utter[2], and Hannah E. Weiher[3]
[1]Sensors and Systems Department, xLab
[2]Digital Communication Implementation Department, Communication Systems and Agile Processing Subdivision
[3]Strategy and Ventures, iLab Office

Prepared for:

Senior Vice President, Engineering and Technology Group

Authorized by: Engineering and Technology Group

Abstract

Slingshot is an Aerospace research and development program to design, prototype, and test on-orbit modular and autonomous technology experiments for next-generation satellite systems. This document describes version 1.0 of the electrical and network interfaces for Slingshot-compatible payloads and provides a “getting started” manual for new payload developers.

Contents

Figures

Tables

Figure 1. Handle integration with several Slingshot 1 payloads. 1
Figure 2. Handle with annotations of key components. 2
Figure 3. Payload Development Kit (PDK). 3
Figure 4. A typical flight configuration for Handle and several payloads. 3
Figure 5. The PDK-Direct GUI. 5
Figure 6. Protocol layers when the PDK is configured for direct communications. 6
Figure 7. Protocol layers in the PDK with COSMOS configuration. 7
Figure 8. Protocol layers in the ground configuration. 7
Figure 9. Protocol layers in the flight configuration. 8
Figure 10. Harwin Gecko connector pin orientation. 13
Figure 11. Tycho Industrial Mini I/O connector drawing. 14
Figure 12. Zero padding for runt packets. 19

1. Introduction

Slingshot is an Aerospace research and development program to design, prototype, and test on-orbit modular and autonomous technology experiments for next-generation satellite systems. The platform offers more agility in the satellite development process by using modular interfaces and enabling a production-like business model to provide recurring opportunities to experiment in space. As many technologies can only be matured through on-orbit testing (such as autonomy, onboard processing, robotics, and communication systems), continuous development enables rapid innovation of these technologies for a variety of space applications. Slingshot 1 is the first of these on-orbit modularity and satellite autonomy testbeds. It will be launching in 2022 with a total of 19 payloads.

To this end, Slingshot defines a generic electrical and network interface for use by any Slingshot-compatible payload. The Slingshot network allows any number of payloads to communicate with each other, with the host spacecraft, and with operators on the ground. The same interface is provided in flight and during development and testing on the ground.

Handle, shown in Figure 1, is the in-flight system that accepts power and data from the host spacecraft and provides Slingshot-compatible power and data interfaces for all connected Slingshot payloads. This document describes the interface from Handle to each payload. Handle's interface with the host spacecraft is defined in a separate document (ATR-2022-00635, "Handle to Spacecraft Interface Control Document").

In the lab, the Slingshot Payload Development Kit (PDK) provides the same interface for one connected Slingshot payload. Command, control, and mission data are relayed over regular Ethernet ports compatible with commercial off-the-shelf (COTS) networking equipment. The PDK's payload-facing interface is effectively a miniature version of Handle, shares much of the same source code, and emulates all the same power and data services as closely as possible. The result is a "test-like-you-fly" environment that enables a smooth transition from laboratory testing to payload integration to flight operations.

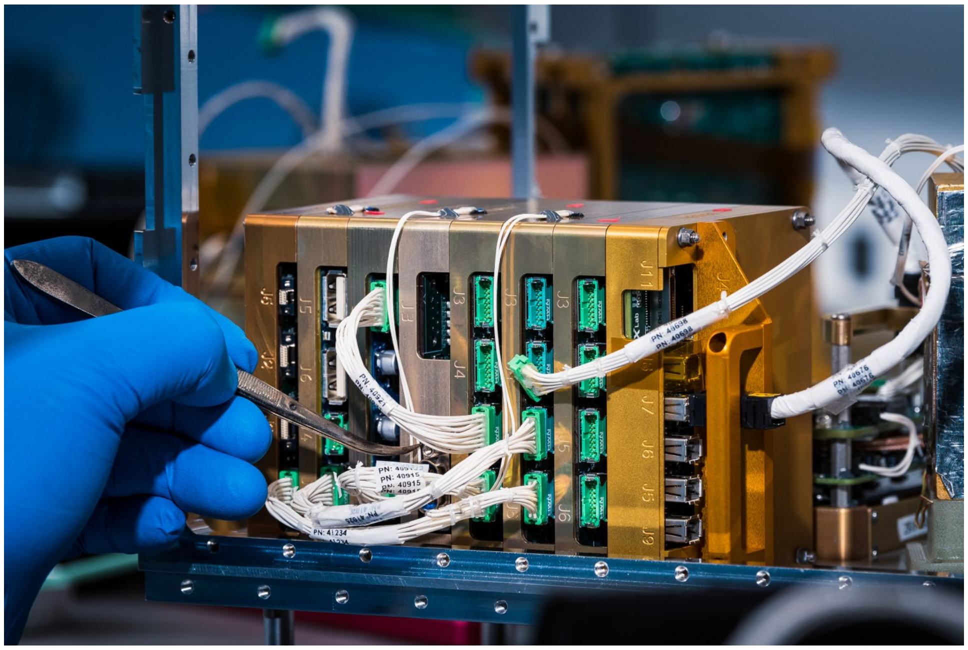


Figure 1. Handle integration with several Slingshot 1 payloads.

2. Overview

This section gives a broad overview of the Slingshot architecture and defines key underlying concepts.

2.1 Architecture

At the highest level, the key components of the Slingshot architecture are the following:

- **Spacecraft bus**. At minimum, the spacecraft bus provides unregulated power and "store-and-forward" communications with the ground. The bus stores commands from the ground and forwards them to Handle on a designated schedule. The bus stores telemetry from Handle and forwards it to the ground when communications are available.

- **Handle**. Handle, shown in Figure 2, is the unit that accepts power and data from the spacecraft bus and distributes it to each Slingshot payload using the modular interface described in this document. It includes an Ethernet switch to convey all required communications.

- **Payload Development Kit (PDK)**. The PDK, shown in Figure 3, is a ground-support analogue for Handle. It connects to COTS network equipment, but provides the same payload-facing interface.

- **Payloads**. Payloads are modular devices that conform to the Slingshot interface. A payload may be a sensor, instrument, processor, or any other functional module.

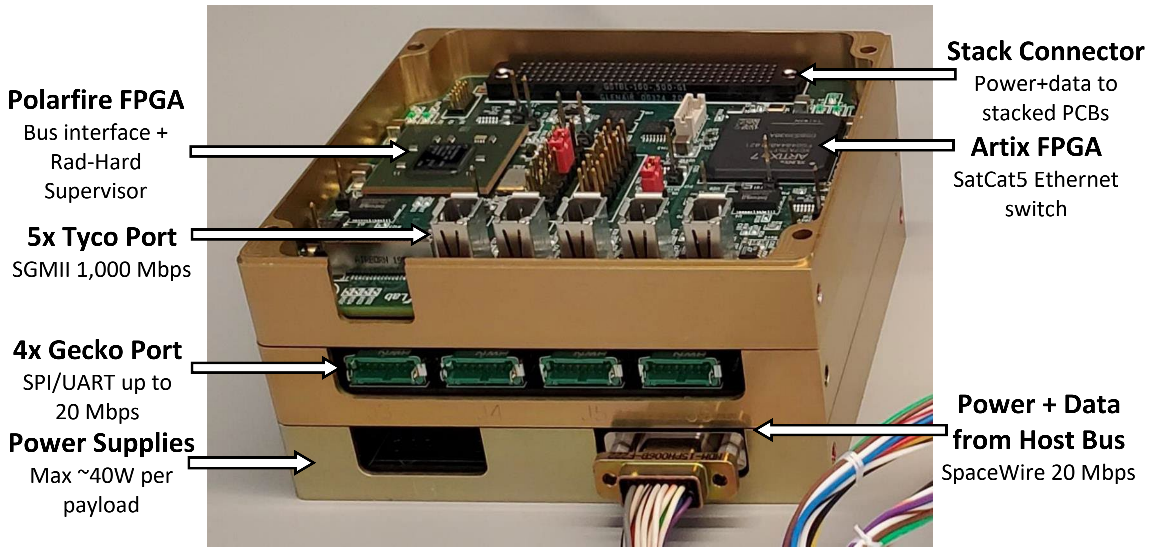


Figure 2. Handle with annotations of key components.

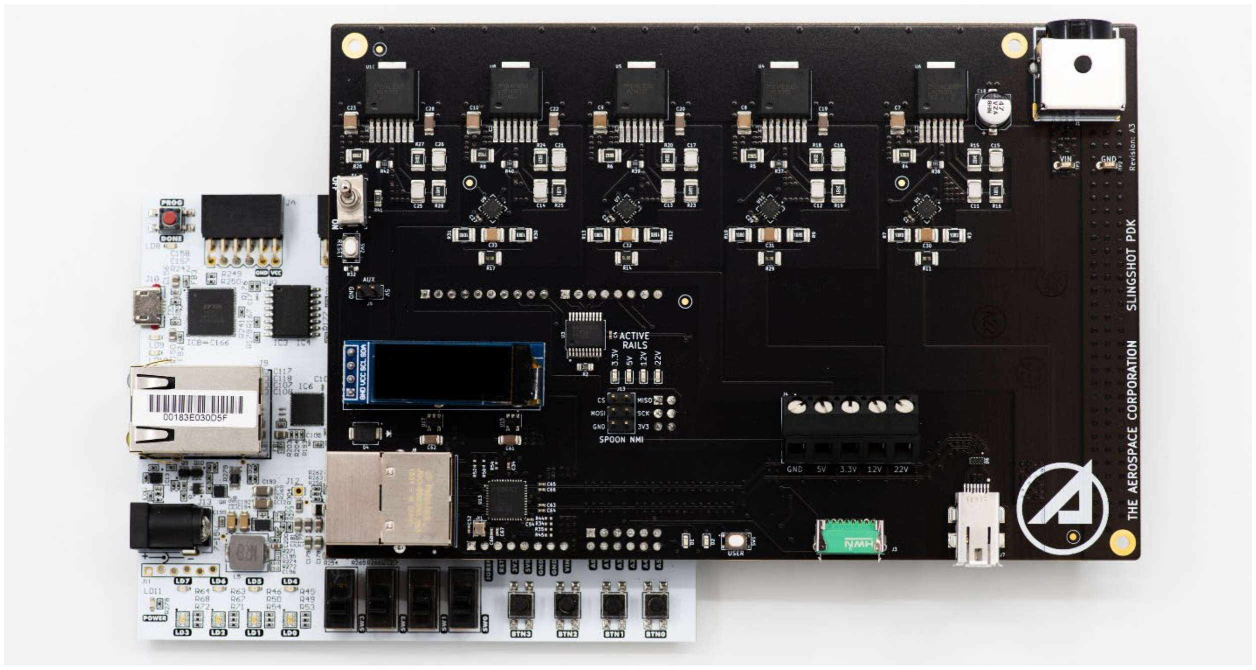


Figure 3. Payload Development Kit (PDK).

Figure 4 shows a typical flight configuration with a spacecraft bus, Handle, and several payloads.

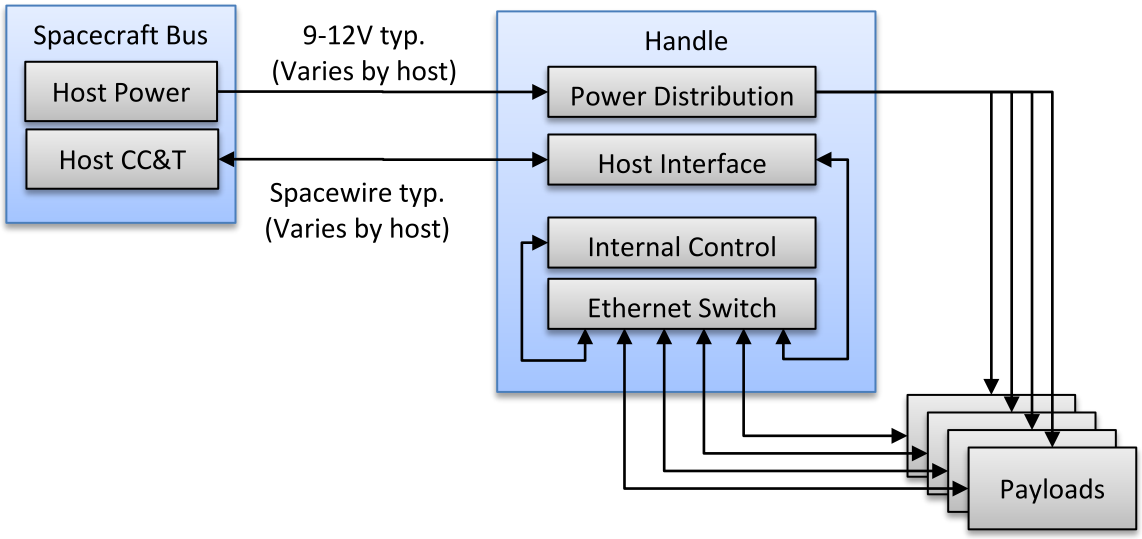


Figure 4. A typical flight configuration for Handle and several payloads.

Conceptually, Slingshot is a peer-to-peer network based on an Ethernet local area network (LAN). Every Slingshot device, real or virtual, is connected to the same LAN, has an assigned media access control (MAC) address, and acts as "just another Ethernet endpoint." This includes Handle's interface to the host, Handle's internal control functions, and each payload. Both Handle and the PDK include an embedded open systems interconnection (OSI) Layer 2 Ethernet switch that is agnostic to all higher-layer protocol choices.

Handle’s power and data interfaces to the host are modular and may be modified as needed to adapt to new spacecraft bus platforms. However, the payload-facing interfaces defined in this document remain constant. This stability includes both electrical interfaces and network protocols. Our intent is to provide a stable environment so that any Slingshot-compatible payload can fly on any future Slingshot mission. One of Handle’s key roles is to insulate payloads from changes to the spacecraft bus.

2.2 Standards

The payload-facing Slingshot interface incorporates the following standards by reference:

- IEEE 802.3 Section 3 (Ethernet Media Access Control Frame and Packet Specifications)
- IETF RFC-1055 (Serial Line Internet Protocol)
- IETF RFC-7049 (Concise Binary Object Representation)
- Cisco ENG-46158 (Serial Gigabit Media Independent Interface)

2.3 Implementation Guidance

SatCat5 is a free open-source tool for creating mixed-media Ethernet interfaces and Ethernet switches, including the Ethernet-over-Serial interfaces used for Slingshot. SatCat5 is used widely in Handle and acts as a reference implementation for many Slingshot payload functions. SatCat5 can be found on GitHub and is freely available under the GNU Lesser General Public License, version 3 (LGPLv3):

https://github.com/the-aerospace-corporation/satcat5

New payload developers should also refer to the “Slingshot PDK User’s Manual.”

2.4 Conformance Criteria

It is vital to test payload conformance to the Slingshot interface requirements. Routine day-to-day use of the PDK during payload development, starting as early as possible, is the best way to maximize the chances that this process goes smoothly. We make every effort to ensure that the PDK mimics Handle accurately. This ensures that developers can catch most problems early on, when they are relatively simple to correct.

Final, run-for-record integration and interoperability testing is performed against Handle in a flight-like configuration (i.e., Handle connected to a real or emulated spacecraft bus). Our goal is to ensure that most problems are caught well before this event, so that the integration process goes smoothly. Final integration of Handle and its payloads should take hours or days, not weeks or months.

2.5 Additional Tools

The software distributed with the Slingshot PDK includes the PDK firmware. It also includes several additional software tools that run on the end user’s personal computer (PC):

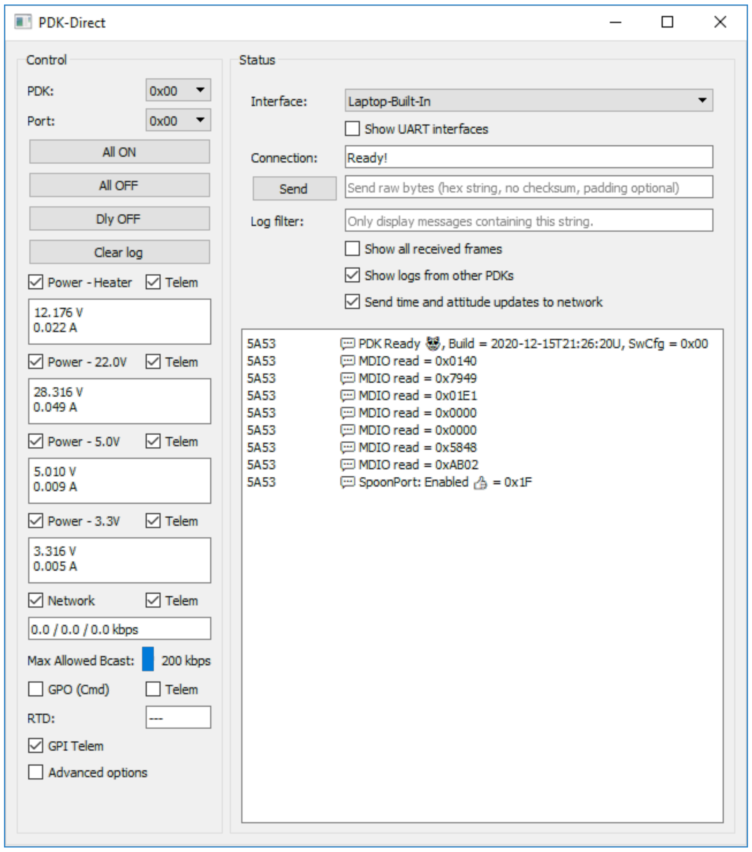
- Application programming interface (API) for control of Handle/PDK functions. (COSMOS, Python)

- API for parsing of Handle/PDK telemetry. (COSMOS, Python)

- Python “bridge” application for relay of COSMOS messages to the PDK and attached payload. The same application can also be used for record/playback of commands. (“main.py”)

- Python “PDK-Direct” graphical user interface (GUI) for control and monitoring of the PDK, shown in Figure 5. (“pdk_direct.py”)

- SatCat5 open-source framework for Ethernet interfaces and switches. (VHDL, C/C++, Python)



PDK: 0x00
Port: 0x00
☑ Power - Heater ☑ Telem
12.176 V
0.022 A
☑ Power - 22.0V ☑ Telem
28.316 V
0.049 A
☑ Power - 5.0V ☑ Telem
5.010 V
0.009 A
☑ Power - 3.3V ☑ Telem
3.316 V
0.005 A
☑ Network ☑ Telem
0.0 / 0.0 / 0.0 kbps
Max Allowed Bcast: 200 kbps
☐ GPO (Cmd) ☐ Telem
RTD: ---
☑ GPI Telem
☐ Advanced options

Interface: Laptop-Built-In
☐ Show UART interfaces
Connection: Ready!
Send raw bytes (hex string, no checksum, padding optional)
Log filter: Only display messages containing this string.
☐ Show all received frames
☑ Show logs from other PDKs
☑ Send time and attitude updates to network

```
5A53        PDK Ready, Build = 2020-12-15T21:26:20U, SwCfg = 0x00
5A53        MDIO read = 0x0140
5A53        MDIO read = 0x7949
5A53        MDIO read = 0x01E1
5A53        MDIO read = 0x0000
5A53        MDIO read = 0x0000
5A53        MDIO read = 0x5848
5A53        MDIO read = 0xAB02
5A53        SpoonPort: Enabled = 0x1F
```

Figure 5. The PDK-Direct GUI.

3. Network Environment

Handle and the PDK are designed to present a consistent environment for payloads, regardless of how those devices, the spacecraft bus, radio terminals, and ground infrastructure are connected. This section gives examples to explain how this consistency is achieved.

Several of these examples use COSMOS, which is a free and open source tool developed by Ball Aerospace. COSMOS can be downloaded from its website:

> https://cosmosc2.com/
> https://github.com/BallAerospace/COSMOS

During initial development, direct connections are encouraged. It can be invaluable to use COTS tools, like SSH or FTP, to connect to Linux-based payloads. However, it is important to know that such advanced capabilities are not always possible in the flight environment. Payload developers should transition to control automation that accurately reflects one-way intermittent communication constraints as soon as possible, to ensure a seamless transition to more constrained flight-like environments. Use of COSMOS is one method for enforcing this requirement, and is encouraged but not mandatory.

3.1 PDK with Direct Communications

In this configuration, a host PC is attached to a payload through a PDK. The PC uses regular Ethernet networking to communicate directly with the PDK and the payload, using any mixture of COTS programs and regular Linux/Windows networking APIs.

In this configuration, control of the PDK is usually through the PDK-Direct GUI, which is a Python application that uses the ScaPy library to send and receive raw Ethernet frames. Diagnostic tools, such as Wireshark, can be used to monitor network traffic in both directions.

Figure 6 shows the protocol layers used for each step of this communication. Each dark blue box is a hardware device; each light blue box is a physical or logical protocol enclosing the box(es) above it; and each green box is the message packet conveyed from the operator to the payload. The user data received by the payload may contain additional protocol layers. In this example, the COTS network interface in the operator's PC sends a 1000BASE-T Ethernet frame to the PDK. The PDK decodes and removes the 1000BASE-T layer, then wraps a new layer for relay to the payload.

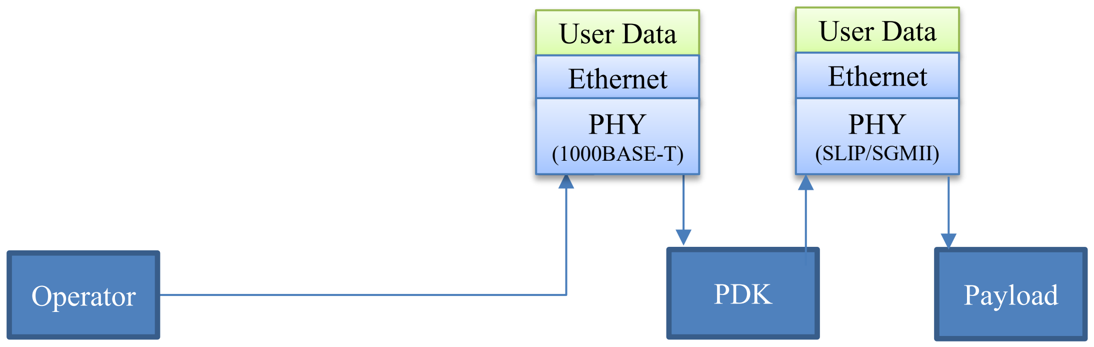


Figure 6. Protocol layers when the PDK is configured for direct communications.

3.2 PDK with COSMOS

In this configuration, shown in Figure 7, a host PC running COSMOS is used to control the payload and the PDK. The COSMOS API is connected to the Ethernet network through the provided Python bridge software (*slingshot_pdk/src/python/main.py*). Wireshark and other network diagnostic tools can still be used.

This configuration is also useful for recording sequences of commands for later playback. Some satellite bus operators use recorded command sequences to operate the spacecraft.

It is possible to run PDK-Direct and the Python bridge simultaneously. If this configuration is required, uncheck the "Send time and attitude updates" option in PDK-Direct to prevent accidental conflict and duplication of those control messages.

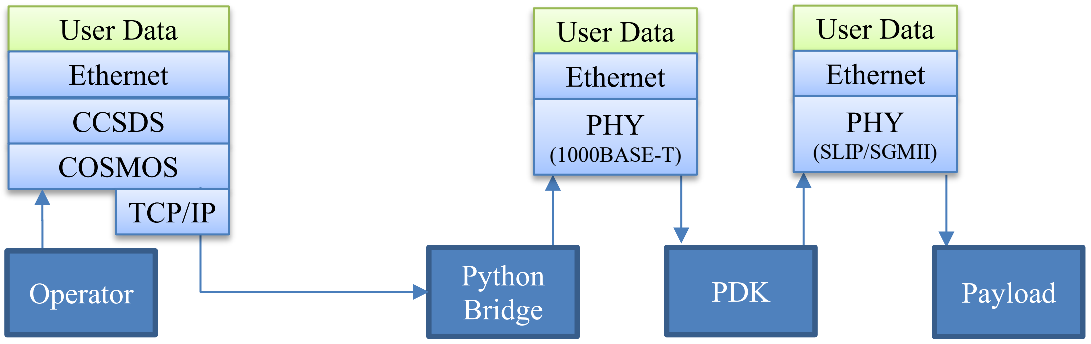


Figure 7. Protocol layers in the PDK with COSMOS configuration.

3.3 Handle with COSMOS (Ground)

In this configuration, shown in Figure 8, a host PC running COSMOS is connected to a satellite bus emulator through software and hardware provided by the bus vendor. The satellite bus emulator is then connected to Handle, which is connected to the payload.

In this configuration, direct Linux/Windows network connections are not possible and Wireshark can no longer be used. COSMOS scripts can be used to control the payload, with no changes required from the Section 3.2 configuration. Two-way realtime communication may still be possible, depending on the capabilities of the satellite bus emulator.

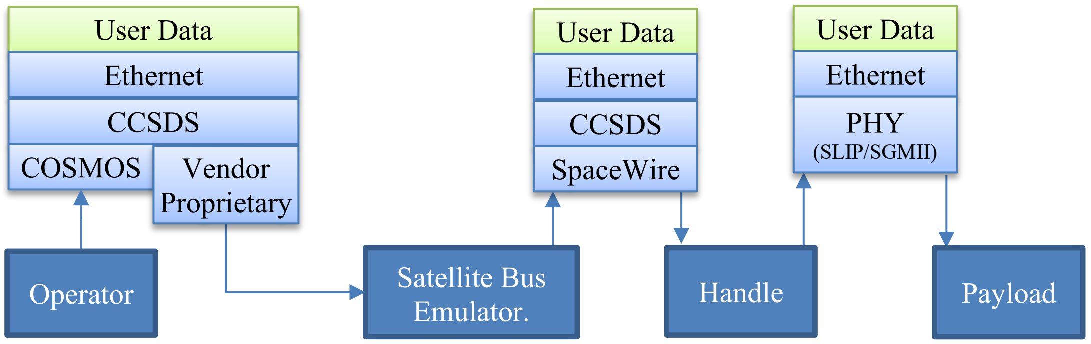


Figure 8. Protocol layers in the ground configuration.

3.4 Handle with COSMOS (Flight)

In this configuration, shown in Figure 9, the spacecraft bus, Handle, and the payload are in space, remotely operated by radio commands. Commands are routed through complex ground infrastructure, uplinked to the spacecraft, and executed on a schedule. Telemetry is downloaded hours or days later. Depending on bus capabilities, two-way realtime communication may be intermittent or even impossible.

Direct Linux/Windows network connections are not possible. COSMOS scripts can be used to control the payload, with no changes required from the Section 3.2 or 3.3 configuration.

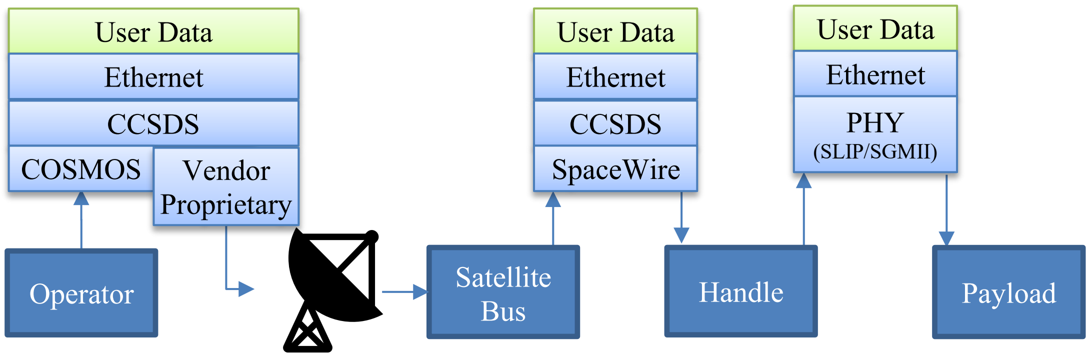


Figure 9. Protocol layers in the flight configuration.

4. Electrical Interface Specification

The Slingshot electrical interface provides power and serial command/data messaging support for Slingshot payloads. The electrical interface uses two connector types for all communications between a Payload and its Host. Host is typically Handle or the PDK.

A high-reliability Harwin Gecko 16-pin, 1.25 mm pitch connector provides the interface for low-speed asynchronous serial messages (commands/data, up to 20 Mbps), time distribution, power distribution, heater and temperature sensors, and bi-level controls and status links.

For Payloads requiring higher-speed communications, a second interface utilizing a Tyco Industrial Mini I/O connector provides gigabit Ethernet capability.

4.1 Ethernet-over-Serial Command and Data Interface

All Slingshot payloads are nodes on an Ethernet LAN. The network switch function provided by Host allows devices to transmit packets in various modes. Messages with a specific node's address are relayed only to that destination. Messages with the reserved broadcast address are relayed to all other nodes, which can see and act upon that information if desired.

To participate on the LAN, nodes must transmit and receive Ethernet frames as defined in the IEEE 802.3 standard. For more information on frame structure and network protocols, refer to Section 5.

The Harwin Gecko connector provides four nontraditional media interface (NMI) pins for Ethernet-over-Serial connectivity up to 20 Mbps. Signal levels on the NMI pins are 0.0 to 3.3 V complementary metal–oxide–semiconductor (CMOS) levels. The NMI pins can operate in serial peripheral interface (SPI) or universal asynchronous receiver transmitter (UART) mode, and the mode is detected automatically. Both modes use Serial Line Internet Protocol (SLIP) encoding as defined in IETF RFC 1055.

On startup, Host drives all four pins to ground. Once the port is activated (usually by the same command that activates Payload power), Host applies weak pull-ups to all four pins and begins listening for either UART or SPI data. The interface is ready for use once Host receives a valid SLIP interframe token (0xC0) using either interface mode. This activates the selected mode and disables all other modes until the interface is power cycled. Payloads should send at least one interframe token during startup to ensure that no later packets are lost. From that point, inputs and outputs are driven according to Table 1 below.

Table 1. Nontraditional Media Interface (NMI) Signals

Mode	Pin 1	Pin 2	Pin 3	Pin 4
Startup	Weak Pullup (100kΩ by Host)	Weak Pullup (100kΩ by Host)	Weak Pullup (100kΩ by Host)	Weak Pullup (100kΩ by Host)
SPI (4-wire)	CSb (Driven by Payload)	MOSI (Driven by Payload)	MISO (Driven by Host)	SCK (Driven by Payload)
UART (4-wire)	CTSb (Driven by Payload)	TxD (Driven by Host)	RxD (Driven by Payload)	RTSb (Driven by Host)

4.1.1 UART Mode

UART mode is the simplest communication mode supported by the NMI interface. The UART format is 1 start bit, 8 data bits, 1 stop bit, no parity. By convention, data is always sent least significant bit first. Baud rate defaults to 921,600 baud, but can be changed by sending a command just before activating the interface.

Two dedicated flow control signals identify when data can be transferred. The active-low clear-to-send (CTSb) signal is driven low by Payload to indicates that Host is permitted to send new data; if Payload is always ready to receive new data, this signal may simply be tied to ground. The active-low ready-to-send (RTSb) is driven by Host and indicates that there is new data ready to be sent; if this signal is not required, it may simply be ignored.

4.1.2 SPI Mode

In SPI mode, Host is always the SPI slave and Payload is the master (i.e., Payload generates the clock, drives chip select, and drives master out slave in (MOSI)). SPI mode 3 is the preferred mode, but other modes can be selected by sending a command just before powering the interface.

All SPI exchanges are byte aligned and transmitted most significant bit (MSB) first per SPI convention. The MOSI and master in slave out (MISO) signals are each treated as a contiguous, SLIP-encoded byte stream. The active-low chip-select (CSb) signal is not used to indicate frame boundaries, and frame boundaries in either direction may or may not be aligned to CSb.

Send and receive operations are inseparable. To execute a concurrent two-way transfer of N bytes, Payload should lower CSb, strobe the serial clock (SCK) 8×N times, and then raise CSb. During this process, Payload transmits each byte on MOSI and Host transmits each byte on MISO.

Both Host and Payload must inspect each received byte for SLIP-encoded frames. If either source has no data to send, then it should repeat the SLIP interframe token (0xC0).

To receive, Payload should send repeating SLIP interframe tokens (0xC0) on MOSI until the received stream emits two consecutive interframe tokens, indicating it has no data to send. This action should be performed frequently, since there is no explicit "data-ready" indicator.

Hardward description language (HDL) source code for a reference implementation of this interface is available on GitHub:

https://github.com/the-aerospace-corporation/satcat5

4.2 Time Distribution Interface

Time information is transmitted from Host to each Payload through the network interface. The “Realtime Clock” message (Section 6.5) indicates the current time, and is associated with a rising-edge fiducial marker that indicates the precise rollover time for each Global Positioning System (GPS) second (i.e., “time at tone”). The time information provided is broadcast every second as received from the spacecraft host and will precede the associated fiducial marker by 100 to 900 ms.

4.3 Power Distribution Interface

Host receives unregulated power from another source (i.e., the spacecraft or utility power) and produces secondary voltages for use by Slingshot payloads. Regulated supply voltages available for payloads are 3.3 V, 5.0 V, and 22 V[1]. There is also an unregulated heater supply. Each of the four power supplies is limited to a maximum of 2.0 A each (derated), with the total across all four lines limited to 4.0 A.

Note that the aggregate power for all payloads is limited by battery reserves, and operations may require duty cycling high-power payloads to fit within available power and energy resource limits.

Power regulation and ripple specifications are provided in Table 2 below:

Table 2. Handle Power Supply Tolerances

Power Supply Voltage	Regulation (at max load)	Ripple
3.3 V	±50 mV	±50 mV
5.0 V	±50 mV	±50 mV
22 V[1]	±200 mV	±200 mV

4.4 Temperature Sensor Interface

Host provides the capability to power and digitize one temperature sensor within each payload assembly at a cadence of once per second. The temperature sensor is a platinum resistance temperature detector (RTD) device with 1 kΩ resistance at 0°C.

The preferred RTD device is Littelfuse part number PPG102JA, which provides sensing over a range from –50°C to +500°C with accuracy of 0.15°C. Note that RTD-monitoring circuitry in Host clips the sensing range to –50°C to +70°C.

The RTD leads within Payload must be isolated from all other signals, chassis, and ground by 1 MΩ or greater. If the temperature sensor interface is unused, Payload should leave the two RTD signals disconnected.

4.5 Heater Interface

Host provides switched power for a single heater installed within Payload. Heater power is unregulated and will typically be in the range from 9.3 to 12.3 VDC. Host limits the heater current to each payload at 2 A to protect the interface wiring.

On command, Host can provide simple thermostatic control based upon the temperature sensor reading and a target temperature setpoint. Temperature control is regulated to ±1°C, and the setpoint temperature is configurable by ground command. By default, autonomous heater operation is disabled and states including off, on, and auto/regulate are selectable by ground command.

[1] The shared “22 V” rail is adjustable from 22.0 V to 28.0 V, with a default of 22.0 V. However, all payloads on a given mission must agree on the selected voltage.

4.6 Gigabit Ethernet Interface

The dedicated Tyco Mini I/O connector provides a gigabit Ethernet interface using the serial gigabit media-independent interface (SGMII) operating in MAC-to-MAC mode. The SGMII interface uses two low-voltage differential signaling (LVDS) pairs, each operating at 1,250 Mbaud, to operate a point-to-point gigabit Ethernet physical link.

For the complete SGMII specification, refer to Cisco Systems ENG-46158. HDL for a reference implementation of this interface is available on GitHub:

https://github.com/the-aerospace-corporation/satcat5

Each LVDS pair is AC coupled at Host's side. Payloads may include their own AC-coupling capacitors if desired.

The LVDS transmit pair (Host to Payload) must be AC terminated by a 100 Ω load (100±1%, 1/32 W) at Payload's end (i.e., 100 Ω equivalent from Tx+ to Tx–). Note that some LVDS receiving devices include built-in termination; if using this feature, ensure that it meets the accuracy requirement, or disable it and provide a precise external resistor. The outputs from Host are AC coupled. Common-mode DC biasing, if required, must be provided by Payload. Differential output voltage from Host may range from ±200 mV to ±600 mV.

The LVDS receive pair (Payload to Host) is AC terminated by a 100 Ω differential load (100±1%, 1/32 W) (i.e., 100 Ω equivalent from Rx+ to Rx–). Host's receiver is AC coupled; any level of common-mode DC bias from Payload is permitted, up to ±5 V. Differential input voltage received by Host must be at least ±100 mV.

4.7 Harwin Gecko Connector Definition

Box-mounted connectors on either Payload or Host are Harwin Gecko P/N G125-MH11605L1P (PC tail latch) or P/N G125-MH11605L3P (surface mount latch), 16-pin right angle connectors with latch lock (plug type).

Pin 1 assignment is shown in Figure 10. The upper row pins are odd (1, 3, 5, …) and lower row pins are even (2, 4, 6, …). See also Harwin's catalog for the Gecko product range:

https://cdn.harwin.com/pdfs/Harwin_PC_Gecko.pdf

Mating receptacle harness connectors are Harwin P/N G125-FV11605L0P (flex cable mount) or P/N G125-2041696L0 (wire harness mount).

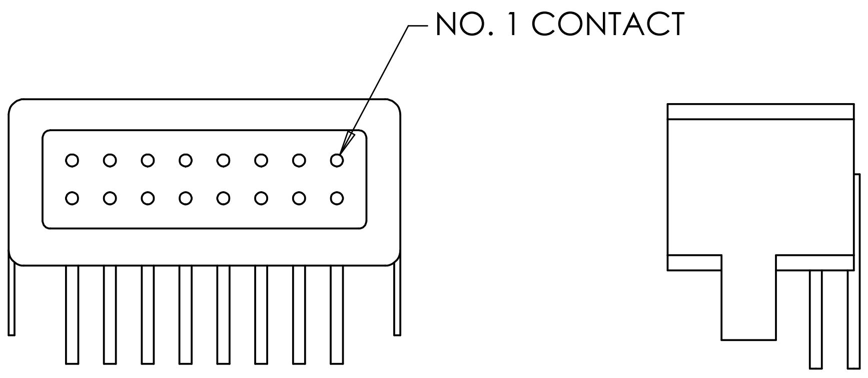


Figure 10. Harwin Gecko connector pin orientation.

4.7.1 Harwin Gecko Connector Pinout

Table 3 lists the purpose of each pin on the Harwin Gecko connector.

Table 3. Harwin Gecko Connector Pinout

Pin #	Signal Name	Description
1	Heater Power	Heater Power (9.3 V - 12.3 V unregulated) Signal Driver: Host
2	Power 1 (22 V)	Regulated 22.0 V to 28.0 V supply up to 2.0 A. Signal Driver: Host
6	Power 2 (5.0 V)	Regulated 5.0 V supply up to 2.0 A. Signal Driver: Host
5	Power 3 (3.3 V)	Regulated 3.3 V supply up to 2.0 A. Signal Driver: Host
3,4	Power Returns	Return path for Heater, 22 V, 5.0 V, and 3.3 V power lines Signal Driver: Host Note: Pins 3 and 4 are the only power returns for Payload and they provide the total return current for all power supplies. Each of the return wires is limited to 2 A capacity so the sum of all power line currents will be limited by Host to 4 A total.
9	RTD+	Temperature sensor in Payload (high side) Signal Driver: Host Payload shall isolate from all other signals, grounds, and chassis by greater than 1 MΩ.
7	RTD–	Temperature sensor in Payload (low side) Signal Driver: Host Payload shall isolate from all other signals, grounds and chassis by greater than 1 MΩ.
8	1PPS	Time fiducial for previously received spacecraft time message, active edge is 0–3.3 V rising edge. Signal Driver: Host Payload input impedance shall be greater than 100 kΩ.
10	Bi-Level To Payload	Logic control signal from Host to Payload (0–3.3 V) set by command Signal Driver: Host Payload input impedance shall be greater than 100 kΩ.

Pin #	Signal Name	Description
12	Bi-Level From Payload	Logic status signal from Payload to Host (0–3.3 V) Signal Driver: Payload Host input impedance shall be greater than 100 kΩ.
13	NMI1	Nontraditional Media Interface, Pin #1 Function depends on interface type. 3.3 V CMOS logic compatible. SPI mode: CSb (Chip-select bar, active low, driven by Payload) UART4 mode: CTSb (Clear-to-send bar, active low, driven by Payload)
14	NMI2	Nontraditional Media Interface, Pin #2 Function depends on interface type. 3.3 V CMOS logic compatible. SPI mode: MOSI (master out / slave in, driven by Payload) UART4 mode: RxD (Received data, driven by Host)
15	NMI3	Nontraditional Media Interface, Pin #1 Function depends on interface type. 3.3 V CMOS logic compatible. SPI mode: MISO (master in / slave out, driven by Host) UART4 mode: TxD (Transmit data, driven by Payload)
16	NMI4	Nontraditional Media Interface, Pin #1 Function depends on interface type. 3.3 V CMOS logic compatible. SPI mode: SCK (Serial clock, driven by Payload) UART4 mode: RTSb (Ready-to-send bar, active low, driven by Host)
11	Digital Ground	Reference return for 1PPS, bi-levels, and NMI interfaces

4.8 Tyco Industrial Mini I/O Connector Definition

Payloads utilizing high-speed communications shall incorporate Tyco Industrial Mini I/O, 8-pin, Type I connectors with interface equivalent to TE Connectivity part number 2069552-1. Tyco Industrial Mini I/O connectors are available in different interface shapes, so it is critical to select the **Type I** configuration (as shown in Figure 11) for compatibility with Host and provided cable harnesses.

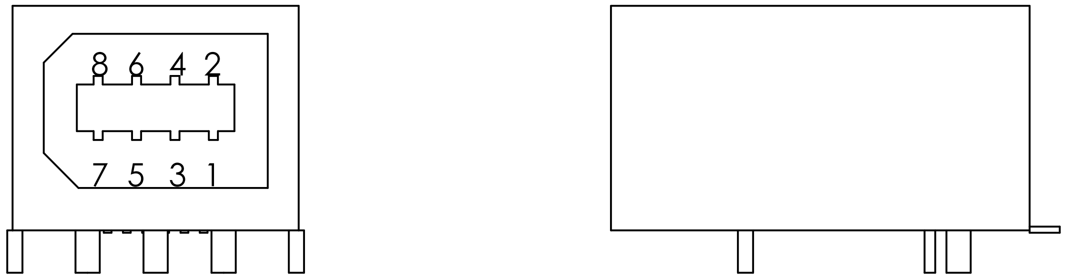


Figure 11. Tycho Industrial Mini I/O connector drawing.

4.8.1 Tyco Industrial Mini I/O Connector Pinout

Table 4 lists the purpose of each pin on the Tyco Industrial Mini I/O connector.

Table 4. Tyco Industrial Mini I/O Connector Pinout

Pin #	Signal Name	Wire Color	Description
1	SGMII Tx–	White/Orange	Low side of LVDS pair Signal Driver: Host
2	SGMII Tx+	Orange	High side of LVDS pair Signal Driver: Host
3,5	28 V Return	White/Green, White/Blue	Return for 28 V supply Signal Driver: Host
4,6	28 V Supply	Blue, Green	Regulated 28 V supply up to 0.7 A each pin Signal Driver: Host
7	SGMII Rx–	White/Brown	Low side of LVDS pair Signal Driver: Payload
8	SGMII Rx+	Brown	High side of LVDS pair Signal Driver: Payload

5. Network Interface Specification

Slingshot provides an Ethernet network for connecting Slingshot payloads to each other and to the spacecraft bus, using an Ethernet switch inside Host (i.e., Handle or the PDK). This section specifies rules and conventions for using this network.

5.1 Requirements Overview

All payloads MUST:

- For each payload interface, send a broadcast network frame at least once every five seconds. The "keep-alive" message 0xAE13 is a useful placeholder if no other messages are sent.

- Keep broadcast traffic below an average rate of 20 kbps
Our goal is to keep total broadcasts from all payloads below 100–200 kbps worst-case.

- Be able to accept and process at least 300 kbps of incoming data.

- Use a unique source MAC address for each attached interface.

- Use a unique port ID that is used for Host power-control commands.

- Ignore traffic with unrecognized EtherTypes™ or port IDs.

- Be prepared to operate with intermittent and highly delayed communications with the ground.

5.2 Ethernet Basics

The Slingshot LAN is tied together by one or more Ethernet switches, which form a tree of point-to-point links in a star topology. Each payload is an endpoint on this tree.

Ethernet messages are broken into distinct frames. Every Ethernet frame contains five fields:

- Destination address (6 bytes)
- Source Address (6 bytes)
- EtherType (2 bytes)
- Frame data (0–1500 bytes)[2]
- Frame check sequence (4 bytes)

The frame check sequence (FCS) uses the 32-bit cyclic redundancy check (CRC) algorithm specified in IEEE 802.3 Section 3.2.9. Known-good examples are also provided in Appendix A. Payloads must include this field and calculate it correctly. The host will silently discard any frame with an invalid FCS.

The method for delimiting the start and end of each frame varies by medium. NMI ports use SLIP encoding (IETF RFC 1055) and must not include any additional preambles, headers, or footers. SGMII ports are 8b/10b encoded and must include preamble, start frame delimiter, and interframe gap as specified in IEEE 802.3 Section 3.35.

[2] IEEE 802.3 has a minimum frame size of 46 bytes. Slingshot hosts can support shorter frames, but additional rules apply.

5.3 MAC Addresses

To participate in the Slingshot network, each payload must have at least one MAC address. It uses this address as the "source" for each packet it sends. Each MAC address is a 48-bit number that must be unique on the LAN. By convention, they are usually written as hexadecimal bytes delimited by hyphens or colons (e.g., 02-50-B6-16-C4-9F).

The payload integrator should assign MAC addresses or otherwise ensure that there are no conflicts. For standards compliance, the first byte should always end in hexadecimal 2, 6, A, or E. This marks it as a "locally administered address" that is not allocated or administered by IEEE. All subsequent bytes can be assigned at random.

Payloads with more than one interface (e.g., an NMI interface and an SGMII interface) must use a unique MAC address for each interface.

5.3.1 Broadcast Traffic

Broadcast frames use the special destination address FF-FF-FF-FF-FF-FF. Such frames are relayed to every endpoint on the LAN.

In general, if you have low-rate information that may be of interest to other payloads, you should send it as a broadcast frame. This ensures that other interested payloads can read pertinent information with no need for further coordination.

Please be courteous to other payloads and avoid overusing broadcast frames, as they can overwhelm low-power processors trying to keep up with a deluge of unwanted data.

5.3.2 Point-to-Point Traffic

Point-to-point frames are sent to a specific destination address. Most payload command, control, and mission data traffic should be in this category. Commonly used Slingshot services, such as diagnostic logs or uplink to the host spacecraft, have designated addresses for this purpose, as shown in Table 5.

Table 5. Slingshot MAC Addresses

Endpoint	Address	Description
Operator	12-34-56-12-34-56	Source address for commands from the ground
Virtual Payload	22-22-22-22-22-22	Simulated "virtual payload" included with PDK software
Logging Service	5A-53-43-6F-72-65	Filtering service for "diagnostic logging" messages
Reserved	5A-61-74-43-61-**	Reserved for internal operation of Host
Switch Control	5A-61-74-43-61-00	Commands for Host (e.g., port power control)
Downlink	5A-61-74-43-61-AA	Destination address for data that should be relayed to ground

5.3.3 Multicast Traffic

The Ethernet switches used in Slingshot hosts do not currently support multicast traffic. If your payload needs these features, please contact The Aerospace Corporation at oss@aero.org.

5.3.4 Virtual Local Area Networks

The Ethernet switches used in Slingshot hosts do not currently support IEEE 802.1Q virtual local area network (VLAN) tags. However, VLAN features, such as network splitting and traffic prioritization, are in the development roadmap. If your payload needs these features, please contact The Aerospace Corporation at oss@aero.org.

5.3.5 Precision Time Protocol (PTP)

The Ethernet switches used in Slingshot hosts do not currently support IEEE 1588 Precision Time Protocol (PTP). PTP messages can be sent and received, but will not apply or modify the timestamps required for full PTP accuracy. However, the development roadmap includes upgrades required to act as an "end-to-end transparent clock" as defined in 1588 Section 10.2. If your payload needs this feature, please contact The Aerospace Corporation at oss@aero.org.

5.4 IP Addresses

Payloads are encouraged, but not required, to support internet protocol (IP) networking (i.e., Address Resolution Protocol (ARP), Internet Control Message Protocol (ICMP), User Datagram Protocol (UDP), Transmission Control Protocol (TCP), etc.). Those that do should be assigned a static IP address and may need to assign static routes. Host does not provide Dynamic Host Configuration Protocol (DHCP) service. Coordinate with the payload integrator to ensure that selected IP addresses and TCP/UDP port IDs are unique.

5.5 Service Guarantees

All Ethernet frames are delivered on a first-come, first-served basis. Prioritization of specific traffic types using 802.1q VLAN tags may be supported in a future revision of this interface.

In normal operation, the network switch will make a best-effort attempt to deliver every packet. However, congestion and other conditions can cause packets to be dropped. This is a feature, not a bug.

If unconditional delivery is required, use a higher-layer protocol that is designed to guarantee this level of service through an unreliable network. TCP/IP is a prime example, and frameworks like lwIP make this practical on embedded platforms.

In all other cases, payloads must accommodate the inevitability of occasional dropped packets. Commands should be made idempotent where practical and tolerate dropped replies.

5.6 EtherType™

The EtherType field is a 16-bit number from 0x0600 to 0xFFFF used to designate the format and purpose of any given message. Do not use this field to indicate frame length.

Some EtherTypes are associated with well-known protocols, such as IPv4 (0x0800). This document defines several EtherTypes associated with specific Slingshot services in Section 6.

Payloads using customized protocols should define their own EtherType values in the range 0xA000 to 0xFEFF and coordinate with other payloads to ensure selected values are unique.

5.7 Runt Frames

The IEEE 802.3 standard requires that each frame's data field must be at least 46 bytes long (i.e., making the complete frame at least 64 bytes long, including header and FCS). In some cases, Slingshot payloads are exempted from this rule.

Specifically, payloads using an NMI interface can send "runt frames" that are shorter than this limit. NMI frames may have a payload as small as 0 bytes. However, frames that cross from an NMI port to any other port type are zero padded to the new minimum size. Zero padding is added between the end of the user data and the start of FCS, as shown in Figure 12. The FCS is always found in the last four bytes of the frame and covers all preceding bytes. As a result, zero padding modifies the FCS; Host handles this step automatically.

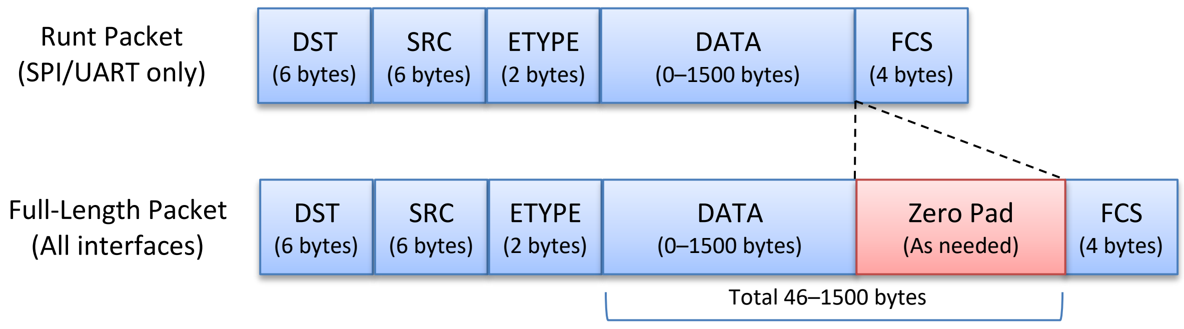


Figure 12. Zero padding for runt packets.

NMI payloads may always send runt frames. However, because they do not always know the route their packets will take, the recipients should be prepared to receive zero-padded frames and ignore the padding.

6. Network Protocol Reference

This section provides detailed byte-by-byte reference information for each of the network protocols used by Handle/PDK for control, status reporting, and information provided to payloads.

Table 6 lists EtherTypes that are reserved on all Slingshot networks. Each message is either *deprecated* (not recommended for new designs), *standard* (part of a well-known standard), *reserved* (internal use for Handle/PDK only), or *open* (recommended for use). All *open* messages are documented later in this section.

Table 6. Well-known and Slingshot-specific EtherTypes

Ether-Type	Name	Status	Description
0x0000-0x05FF	N/A	DEPRECATED	Old-style frame length.
0x0800	IPv4	STANDARD	EtherType for IPv4 traffic (ICMP, UDP, TCP/IP)
0x0806	ARP	STANDARD	Address Resolution Protocol (ARP)
0x1234	Virtual Payload	RESERVED	Commands for “virtual payload” PDK demo.
0x4321	Virtual Payload	RESERVED	Telemetry for “virtual payload” PDK demo.
0x5C00	Switch Status	OPEN	Host state-of-health telemetry
0x5C01	Switch Control	OPEN	Host command and control
0x5C02	Realtime Clock	OPEN	Time-at-tone broadcast
0x5C03	Diagnostic Logging	OPEN	Human-readable status and error messages
0x5C04	Handle Control	RESERVED	Internal control for Handle FPGAs
0x5C05	Handle Status	RESERVED	Internal telemetry for Handle-HS FPGA
0x5C06	Planned Shutdown	OPEN	Notice of impending shutdown command
0x5C07	Handle Status	RESERVED	Internal telemetry for Handle-LS FPGAs
0x5C08	N/A	DEPRECATED	Legacy message format, no longer supported
0x5C09	Loopback Testing	RESERVED	Loopback packet-error rate testing
0x5C0B	Spacecraft Attitude	OPEN	Spacecraft position and attitude
0x5C14	Handle Control	RESERVED	Internal control for Handle-HS FPGA
0x5C80-0x5C8F	Handle Flash	RESERVED	Host firmware updates
0x88F7	PTP	STANDARD	Precision Time Protocol (Reserved for future support)
0xAE13	Keep-Alive	OPEN	Empty broadcast to indicate a payload is alive
0xC5D5	Wrapped CCSDS	OPEN	Relay CCSDS frame to host spacecraft
0xCB00	Common Telemetry	OPEN	Generic state-of-health telemetry

6.1 “Common Telemetry” Message

Use of the “common telemetry” message format is encouraged but not required. It is intended to be a multipurpose, extensible status message that serves as the foundation for regular telemetry analogous to a controller area network (CAN) bus (i.e., an unmoderated publish/subscribe model where sensors broadcast data, and other devices parse data of interest out of the global data stream). The goal is to allow easy sharing of state-of-health and status information between multiple payloads.

The messages are usually sent as broadcasts at some regular cadence, not to exceed once per second per payload.

The message contents are encoded using Concise Binary Object Representation (CBOR), which is defined in IETF RFC 7049. Each message contains a single CBOR key/value dictionary with integer-valued keys. For example usage of CBOR, refer to this interactive encoder and decoder tool:

http://cbor.me/?diag={1001:%20%22Test%22,%201002:%20%22Data%22,%201003:%205678}

The numeric keys used by Handle and the PDK are listed in Table 7 below; this table is expected to expand over time. If an existing tag does not meet your payload requirements, make a new one. Coordinate with your payload integrator to ensure new tag indices are unique.

Whenever possible, use engineering units (i.e., not raw analog-to-digital converter counts, etc.). To accommodate simple microcontrollers, fixed-point formats are strongly preferred over floating-point formats. Choose scaling to give ample resolution, leaving room for future payloads that may need very fine measurements (e.g., power measurements in microwatts, even if your payload gives only 10 mW resolution).

Table 7. Tag IDs for Common Telemetry

Tag ID	Format	Description
0x00000000	UINT	GPS Time of Week, in milliseconds
0x00010000	INT or array of INT	Temperature(s), in millidegrees Celsius
0x00020000	INT	Payload power consumption in microwatts (negative numbers indicate net power generation)
0x00030000	UINT	Network packets sent since last update
0x00030001	UINT	Network packets received since last update
0x00040000	String	Software build timestamp, ISO 8601 format (e.g., “2020-12-31T23:59:59U”)

6.2 “Diagnostic Logging” Message

“Diagnostic logging” messages are used to create human-readable activity logs. Host does not store these messages, but filters and relays them for review by operators on the ground. Many Handle and PDK error messages are reported using this format.

Payloads should send these messages for significant events, such as:

- Payload startup (Tip: include useful info such as the software build date)
- Errors of any type
- Major commands (e.g., powering up an instrument)
- Major events (e.g., instrument power-up complete)
- Any other runtime milestones that would be useful in isolating faults

Please use discretion in emitting log messages, even at low priority. The logging system has a limited capacity to ingest and filter messages across all payloads.

The “diagnostic logging” message always uses EtherType 0x5C03. Payloads should send log messages to the reserved MAC address 5A:53:43:6F:72:65. The contents are a CBOR array containing a payload identifier, timestamp, priority code, and message text. Full details are included at the end of this section.

A priority code is an integer used to filter the verbosity of log messages for both download and display.

Legal priority values range from –20 to +20, with higher values indicating higher priority. Use the following examples and guidelines when setting the priority level of a given message. If in doubt, use the lowest priority that fits:

- –20: DEBUG
  Diagnostic messages that are useful only during deep-dive debugging of a specific payload. Not usually displayed.
  Suggested rate limit: <5 per second.

- –10: INFO
  Informational messages that mark useful progress milestones.
  Suggested rate limit: <2 per second, <5 for high-activity bursts.

- 0: WARNING
  Messages that indicate a potential problem, near miss, unexpected condition, or minor recoverable error.
  Suggested rate limit: <2 per second.

- 10: ERROR
  Serious but nonpermanent errors. One-time hardware faults.
  Suggested rate limit: <1 per second.

- 20: CRITICAL
  Errors that indicate permanent damage, gross design errors, or risk to operator safety.
  Suggested rate limit: Zero except under extreme circumstances.

Message strings are always UTF-8 encoded.

If you are sending log messages, UTF-8 is backward-compatible with the usual ASCII character set (i.e., values 32-127). For such users, one byte remains one character for English text and you can continue sending ASCII strings with no changes.

If you are receiving log messages, be prepared to receive more exotic Unicode endpoints, up to and including the poop emoji. If display of these Unicode endpoints is impractical or impossible, skip over any bytes outside the ASCII range.

For more information on Unicode encodings, refer to Joel Spolsky's essay:

> https://www.joelonsoftware.com/2003/10/08/the-absolute-minimum-every-software-developer-absolutely-positively-must-know-about-unicode-and-character-sets-no-excuses/

The field-by-field contents of the “diagnostic logging” message are shown in Table 8.

Table 8. “Diagnostic Logging” Message Contents

Index	Name	Format	
0	Log Structure	Array	Outer structure for message contents.
0.0	Payload ID (Optional)	Integer/Null	Optional payload ID number.
0.1	GPS TOW	Integer	GPS “time of week” timestamp, in milliseconds. If time is unknown, set this field to –1.
0.2	Priority	Integer	Message priority (–20 to +20).
0.3	Msg. Text	String	UTF-8 encoded message string. Note: Null termination is not required.
0.4+	Reserved	N/A	Reserved for future expansion.
1	Added info (Optional)	N/A	User-defined additional information. When practical, this information will be preserved, but may be dropped as needed (e.g., bandwidth or memory limits).

An example of a valid log message is as follows:

- Payload ID = 12, TOW = 1234, Priority = 20, Message = "Test message"
- CBOR: [12, 1234, 20, "Test message"]
- Bytes: `84 0C 19 04 D2 14 6C 54 65 73 74 20 6D 65 73 73 61 67 65`

6.3 “Keep-Alive” Message

The “keep-alive” message is an empty multipurpose placeholder that mostly functions as a “heartbeat” indicator. Most payloads should send a broadcast “keep-alive” message once on startup and again every 3–5 seconds until shutdown. It is always safe to ignore received “keep-alive” messages, but they serve a useful role in keeping address tables up to date and helping detect which other payloads are powered and connected to the network.

The “keep-alive” EtherType is 0xAE13. The destination MAC is always the broadcast address (FF:FF:FF:FF:FF:FF). The message contents are always empty or zero padding to the minimum necessary length.

6.4 “Planned Shutdown” Message

The “planned shutdown” message is sent by Host to notify payloads that are about to shut down (e.g., it is triggered automatically by opcode 0x11 of the “switch control” message).

Payloads that prefer an orderly shutdown should watch for messages containing their port ID, so they can begin preparations. Host does not maintain a port-ID-to-MAC-address index; Payload must know its own port ID. The port will have its power cut after the specified delay.

The “planned shutdown” EtherType is 0x5C06. The destination MAC is usually the broadcast address (FF:FF:FF:FF:FF:FF). The frame contents are a CBOR data structure.

The field-by-field contents of the “planned shutdown” message are shown in Table 9.

Table 9. "Planned Shutdown" Message Contents

Index	Name	Format	Description
0	Container	Array	Outer structure for message contents
0.0	Port ID	Integer	Port ID that is being shut down
0.1	Time remaining	Integer	Time remaining until shutdown, in milliseconds.
1+	Reserved	N/A	Reserved for future expansion

6.5 "Realtime Clock" Message

The realtime clock service is operated by Host. Whenever Host can synchronize to the spacecraft bus clock, it begins sending this message. The "realtime clock" message will be sent approximately once per second.

The timestamp in each message reflects the best available estimate of the time that the packet was sent. If more precision is required, the received packet should be used to disambiguate subsequent pulse-per-second (PPS) rising-edge events (i.e., "At the tone, SV internal time will be...").

Human-readable timestamps should use GPS time rather than UTC or El Segundo local time. The offset between GPS and UTC varies over time due to the insertion of leap seconds at irregular intervals. As of 2022, GPS time leads UTC by 18 seconds, with the most recent leap second inserted in 2016. Use of GPS time eliminates the need for most payloads to deal with leap seconds and other complexities.

The "realtime clock" EtherType is 0x5C02. The destination MAC is always the broadcast address (FF:FF:FF:FF:FF:FF). The byte-by-byte contents of the "realtime clock" message are shown in Table 10. All integers are big-endian.

Table 10. "Realtime Clock" Message Contents

Byte Offset	Name	Format	Notes
0-3	GPS Week	UINT32	Weeks since GPS epoch (i.e., GMT 24:00:00 Jan 5, 1980)
4-7	GPS TOW	UINT32	Milliseconds since GPS start of week (i.e., 24:00:00 Saturday or 00:00:00 Sunday) Rollover every 604,800,000 milliseconds.
8-15	ISL12082	BCD × 8	Human-readable timestamp in the ISL12082 format. From MSB: DW:YR:MO:DT:HR:MN:SC:SS Each field is a BCD-encoded byte with additional flags. Note that the 24-hour flag (HR bit 7) will always be set. Refer to the ISL12082 datasheet for additional details.
…	…	…	Additional fields reserved for future expansion

6.6 "Spacecraft Attitude" Message

The spacecraft bus usually sends attitude, position, and orbit information to Host. Host will then translate that information to a standard format and broadcast it to the Slingshot LAN for use by Slingshot payloads.

The format of that message is not yet determined. Please contact The Aerospace Corporation if your payload requires attitude information.

6.7 “Switch Control” Message

Host provides core control functions for payload management, including power control. This message type is used to control these functions using a connectionless command API.

COSMOS users should use the API provided in "pdk_utils.rb" to construct these messages.

Commands are sent to the MAC address for a given Handle/PDK control core. The address for core #xx is 5A:61:74:43:61:xx. In configurations with one control core, including the flight configuration, the index is always zero. Test configurations using more than one PDK may use different indexing.

Each “switch control” message is a raw Ethernet frame with EtherType 0x5C01. The destination MAC is always a specific Handle core address (5A:61:74:43:61:xx, see above). For safety reasons, broadcast messages of this type are ignored.

The frame contents are a CBOR data structure. However, most commands are simple enough that a full CBOR encoder library may not be required.

The field-by-field contents of the “switch control” message are shown in Table 11.

Table 11. "Switch Control" Message Contents

Index	Name	Format	Description
0	Container	Array	Outer structure for message contents
0.x	Command	Array	A set of commands, where each command is an array containing an opcode and its argument(s)
0.x.0	Opcode	Integer	See Table 12
0.x.1+	Varies	Varies	Argument(s) for the opcode, if applicable
1+	N/A	N/A	Reserved for future expansion

A full list of supported opcodes is defined in Table 12; all other values are reserved:

Table 12. "Switch Control" Opcodes

Opcode	Name	Arguments	Notes
0x10	Payload power	Int: Port ID Bool: Power-enable (Optional) Int: Bitmask	Immediately enable or shutdown power supplies for designated port ID. See ““Switch Status” for discussion of port IDs. If no mask is provided, all supplies are enabled or disabled. Otherwise, the bitmask selects which setting(s) to change: 0x01: 3.3 V power 0x02: 5.0 V power 0x04: 28.0 V power 0x08: Heater power

Opcode	Name	Arguments	Notes
			0x10: Network port
0x11	Planned shutdown	Int: Port ID Int: Delay (msec)	Planned shutdown of all power supplies for the designated port ID. A notification will be sent to affected payload(s). Power is cut to all rails simultaneously after the specified delay.
0x12	Reset switch	Int: Confirmation	Immediately reset the PDK or Handle FPGA. To reduce the chance of accidental reset, the required confirmation code is "793,136."
0x20	Set/connect GPO	Int: Port ID Bool: New GPO value -or- Int: Port ID Int: New GPO source	Change the value of "Bi-level to Payload" (Gecko pin 10) for the designated port ID. If the value is a BOOLEAN, the pin will be set high or low accordingly (all host types). If the value is an INTEGER (1-4), the pin will be driven by the designated bi-level signal from the host bus (Handle only).
0x21	Port status	(Optional) Int: Port ID	Request an immediate "Switch Status" message. This includes power supply status and the value of the "Bi-level from Payload" (Gecko pin 12). If a port ID is specified, only that information is sent. Otherwise, information for all ports is sent.
0x22	Promiscuous mode adj.	Int: Port ID Bool: New value	Enable or disable "promiscuous mode" for the designated payload's Ethernet port. When set, ALL switch traffic will be copied to the port, regardless of destination. (High-speed ports only!)
0x23	Connect GPI	Int: Port ID Int: New GPI dest.	Connect a port's GPI pin to the designated bi-level input on the spacecraft bus (Handle only).
0x24	Adjust UART	Int: Port ID Int: Baud rate (Hz)	Gecko ports only: Change the UART-mode baud rate. Default on reset is 921,600 baud.
0x25	Adjust SPI	Int: Port ID Int: New mode (Optional) Int: Filter	Gecko ports only: Change the active SPI mode (0/1/2/3, default 3) Optionally, adjust the glitch-prevention filter setting (range 0-255, higher values reduce max speed, default 1).
0x30	Adjust max broadcast	Int: New rate (bps) Int: Count before cutoff	Payloads that generate excessive broadcast traffic may jam the network. Repeat offenders are automatically disconnected; this command adjusts that alarm threshold. Default startup alarm is 80 kbps for 3 consecutive seconds.
0x31	Log-fwd. threshold	Int: New threshold	In the flight configuration, adjust the minimum priority threshold for forwarding Diagnostic Logging messages. Messages with a priority code at or above the threshold are forwarded to the spacecraft bus for later download. This opcode has no effect in the PDK configuration, where all messages are forwarded.
0x32	Idle-port threshold	Int: New timeout (secs)	Payloads are required to transmit at least one packet every five seconds. If idle for more than N

Opcode	Name	Arguments	Notes
			seconds (default 30), assume software malfunction and begin a "planned shutdown" (see above). The same threshold is applied to all ports. A timeout of zero disables this check.
0x40	Heater set point	Int: Port ID Bool: Enable thermostat Int: Temperature (m°C)	Enable or disable thermostat function using heater power (low-speed ports only, not available in PDK).
0x41	Current limit adjust	Int: Port ID Int: Bitmask Int: Max current (mA) Int: Max alarms	By default, each current rail is limited to 2.0 A for 3 consecutive measurements. Exceeding this limit automatically shuts down power. Use this command to specify a different limit for specific power supplies (see mask definition above). To forcibly disable this safety feature, set the "max current" and "max alarm" fields to zero.
0xF8	Suppress broadcast telemetry	Int: Debug level (0/1/2)	Suppress or limit certain state-of-health and keep-alive messages to reduce background network traffic. This mode is not recommended for flight use, but can be useful in payload bringup and other debugging. Level 0: Normal operation. Level 1: State-of-health sent in unicast mode. Level 2: Also suppress keep-alive messages.
0xF9	Request FPGA build date	None	Request that FPGA build date be included in next "common telemetry" message.
0xFB	MDIO Write	Int: Register address Int: Register value	PDK only: Write to a register on the PDK's SGMII Ethernet PHY. Useful for SGMII troubleshooting.
0xFC	MDIO Read	(Optional) Int: Reg addr	PDK only: Read from register(s) on the PDK's SGMII Ethernet PHY. Results are returned as a series of diagnostic logging messages. Useful for SGMII troubleshooting. If an address is specified, the PDK will read that register. Otherwise, it reads registers 0x00, 0x01, 0x04, 0x05, 0x0A, 0x10, and 0x11.
0xFD	Port ID aliasing	Int: Original port ID Int: Aliassed port ID	PDK only: Normally, the PDK uses fixed port IDs for the Gecko port (ID #0) and the Tyco port (ID #1). However, Handle uses several dozen port IDs, ranging from 0x11 to 0x85, which are assigned to individual payloads. This opcode adds an alias to the specified port, so that it can be addressed using either the original or alternate port ID. This can be used to facilitate test-like-you-fly configurations using the PDK. The aliased port ID must not be used by any other port. To revert, set the alias to be the same as the original ID.

Opcode	Name	Arguments	Notes
0xFE	Debug-IRQ	None	(Internal debugging) Send a diagnostic logging message with information on IRQ timing.
0xFF	Debug-echo	Int: Priority String: Message to echo	(Internal debugging) Send a diagnostic logging message with the designated priority and message string.

Here is an example of a valid "switch control" message:

- Turn on power to Payload #5, and set the GPO for Payload #6 high.
- CBOR: [[0x10,5,true], [0x20,6,true]]
- Bytes: `82 83 10 05 F5 83 18 20 06 F5`

6.8 "Switch Status" Message

Each host device sends a "switch status" message with state-of-health telemetry for each attached payload. It is usually sent once per second, but the rate is reduced if no payloads are currently active. It may also be sent on demand in response to certain commands (see ""Switch Control").

Each payload is identified by a port ID. This number identifies the physical port, but is NOT linked to the MAC address or other payload identifiers. Each port ID is assigned to a specific payload by the payload integrator and is guaranteed to be unique across the entire spacecraft network.

Most configurations will always have a single host device, including the flight configuration. However, in test configurations using multiple PDK units, each one has a unique MAC address and sends this message separately.

The "switch status" message has EtherType 0x5C00. In most cases, the destination MAC is the broadcast address (FF:FF:FF:FF:FF:FF). If requested on demand, the destination is echoed to the device that issued the request.

The frame contents are a CBOR object containing a single key/value dictionary. The keys are the port ID as discussed above. Each value is a structure with the items shown in Table 13.

Table 13. "Switch Status" Message Contents

Index	Name	Format	Description
0	Container	Map	Dictionary with one item per attached port
0.x	Port Status	Int -> Array	Key/value pair mapping port ID to its status report
0.x.0	Enabled	Bool	Is this port currently powered?
0.x.1	GPI pin	Bool	Value of the GPI pin (aka "Bi-level from payload")
0.x.2	Temperature	Integer	Temperature, in millidegrees Celsius (or NULL if temperature is unknown or unreadable).
0.x.3	Voltage	Array[Int]	Voltage of each active power rail, in ascending order by nominal voltage. Each field in millivolts. Interpretation varies by port type: 0 (Shutdown) 1 (Tycho): 28 V

Index	Name	Format	Description
			3 (Handle-LS): 1.5 V, 3.3 V, 5.0 V 4 (Gecko): 3.3 V, 5.0 V, Heater, 28.0 V
0.x.4	Current	Array[Int]	Current of each active power rail, in mA. Same length/order as the voltage telemetry, see above.
0.x.5	Traffic stats	Varies	Array[Int]: Traffic statistics for this port (bytes/sec): Broadcast (payload to network) Total sent (payload to network) Received (network to payload) Status word (meaning varies per port) True: Network port is active, but statistics unavailable Null: Network port is inactive
0.x.6	GPO pin	Bool	Value of the GPO pin (aka "Bi-level to payload")
0.x.7+	N/A	N/A	Reserved for future expansion
1+	N/A	N/A	Reserved for future expansion

An example "switch status" message:

- Port ID = 5: Power=Off, GPI=0, Temperature=20.0°C, Power=Off
- Port ID = 6: Power=On, GPI=1, Temperature=20.1°C, Power=3.3V@500mA, 28.0V@250mA
- CBOR: {5:[false,false,20000,[],[]], 6:[true,true,20100,[3300,28000],[500,250]]}
- Bytes: `A2 05 85 F4 F4 19 4E 20 80 80 06 85 F5 F5 19 4E 84 82 19 0C E4 19 6D 60 82 19 01 F4 18 FA`

6.9 "Wrapped CCSDS" Message

In the flight configuration, Handle relays Ethernet telemetry from payloads to the spacecraft bus. The bus then records the data so it can be downlinked during the next pass over a ground station. These Ethernet frames are wrapped in a specific CCSDS format to designate them for recording.

The "wrapped CCSDS" message is an alternate format that is used to send CCSDS telemetry packets directly to the flight computer on the spacecraft bus, if it allows such forwarding. It effectively inverts the wrapping that is performed for normal telemetry. This can be used to notify the host bus of specific required actions. The packet will be unwrapped by Handle before transmission to the flight computer.

The EtherType is 0xC5D5 and the destination MAC is always the Handle PolarFire (5A:61:74:43:61:A0).

The frame contents are a complete CCSDS frame, as shown in Table 14:

Table 14. "Wrapped CCSDS" Message Contents

Byte Offset	Name	Value	Notes

0-1	Header flags	Version = 0
		Type = 0
			Big-endian.

Byte Offset	Name	Value	Notes

		Flag = 0
		APID = 0x300–0x3FF	

2-3	Grouping and Sequence Count	Grouping = 0x3 Monotonic counter	Big-endian.
4-5	Packet length	UINT16	Big-endian. Value = N–1, where N is the length (in bytes) of the contained packet data.
6+	Packet Data	N Bytes	Handle calculates and appends CRC.

The message contents inherently vary by spacecraft bus. For compatibility's sake, most payloads should avoid relying on "wrapped CCSDS" messages for routine operation. If needed, refer to the bus vendor's documentation for all details of the CCSDS message contents, and coordinate with the payload integrator to ensure all applicable requirements are met.

Appendix A.   Frame Check Sequence Examples

All Ethernet frames must end with a valid frame check sequence (FCS). Packets with an incorrect FCS will be dropped silently at the switch.

The FCS is calculated using the CRC32 algorithm. However, there are many variants of this algorithm. Note that, due to mathematical convention, the least significant byte of the CRC remainder is transmitted first. The following two packets are known reference sequences that have been verified against COTS Ethernet equipment, with all byte sequences in order of network transmission.

Reference #1:

```
FF-FF-FF-FF-FF-FF-00-20-AF-B7-80-B8-08-06-00-
01-08-00-06-04-00-01-00-20-AF-B7-80-B8-80-E8-
0F-94-00-00-00-00-00-00-80-E8-0F-DE-DE-DE-DE-
DE-DE-DE-DE-DE-DE-DE-DE-DE-DE-DE-DE-DE-DE-DE
➔ FCS = 9E-D2-C2-AF
```

Reference #2:

```
FF-FF-FF-FF-FF-FF-00-00-00-04-14-13-08-00-45-
00-00-2E-00-00-00-00-40-11-7A-C0-00-00-00-00-
FF-FF-FF-FF-00-00-50-DA-00-12-00-00-42-42-42-
42-42-42-42-42-42-42-42-42-42-42-42-42-42-42
➔ FCS = 9B-F6-D0-FD
```

AEROSPACE REPORT NO.
ATR-2022-01270

Slingshot Payload Manual

Cognizant Program Manager Approval:

Charles J. Player, PRINCIPAL DIRECTOR
STRATEGY INTEGRATION OFFICE
OFFICE OF THE CHIEF TECHNOLOGY OFFICER

Aerospace Corporate Officer Approval:

Todd M. Nygren, SENIOR VP ENGINEERING & TECHNOLOGY
OFFICE OF EVP

Content Concurrence Provided Electronically by:

Alexander C. Utter, SENIOR ENGINEER SPECIALIST
DIGITAL COMM IMPLEMENTATION DEPT
COMMUNICATION SYS & AGILE PROC SUBDIV
ENGINEERING & TECHNOLOGY GROUP

Office of General Counsel Approval Granted Electronically by:

Kien T. Le, ASSISTANT GENERAL COUNSEL
OFFICE OF THE GENERAL COUNSEL
OFFICE OF GENERAL COUNSEL & SECRETARY

AEROSPACE REPORT NO.
ATR-2022-01270

Slingshot Payload Manual