Bradley Davis

The communication between the ground and satellite is separated into packets. This document specifies the form and function of these packets.

Packet Protocol

Detailed Description of Communication Packet

Revision: 1.1.1



Table of Contents

[1 Packet Header 3](#_Toc520040080)

[1.1 Multipacket Additional Header 3](#_Toc520040081)

[2 Packet Body 4](#_Toc520040082)

[2.1 Single Packet 4](#_Toc520040083)

[2.2 Multipacket 4](#_Toc520040084)

[2.1.1 First Packet 4](#_Toc520040085)

[2.1.2 Second to (n – 1)th Packet 4](#_Toc520040086)

[2.1.2 nth Packet 4](#_Toc520040087)

[2.1.3 Lost Packet Request 4](#_Toc520040088)

[3 Commands 6](#_Toc520040089)

[3.1 ADCS 6](#_Toc520040090)

[3.2 IFJR 6](#_Toc520040091)

[3.2.1 Processor Binary Image Upload 6](#_Toc520040092)

[3.2.2 Reprogram a Device 6](#_Toc520040093)

[3.3 IHU 7](#_Toc520040094)

[3.3.1 Telemetry 7](#_Toc520040095)

[3.4 PMIC 8](#_Toc520040096)

[3.5 Comms 8](#_Toc520040097)

[3.5.1 Radio Configuration 8](#_Toc520040098)

[3.6 Payload 1 9](#_Toc520040099)

[3.6.1 Configuration 9](#_Toc520040100)

[3.6.2 Data Download 9](#_Toc520040101)

[3.7 Payload 2 10](#_Toc520040102)

[3.7.1 Configuration 10](#_Toc520040103)

[3.7.2 Data Download 10](#_Toc520040104)

[3.8 Payload 3 10](#_Toc520040105)

[3.8.1 Configuration 11](#_Toc520040106)

[3.8.2 Data Download 11](#_Toc520040107)

[4 Units 12](#_Toc520040108)

[4.1 Voltage 12](#_Toc520040109)

[4.2 Current 12](#_Toc520040110)

[4.3 Power 12](#_Toc520040111)

[4.4 Energy 12](#_Toc520040112)

[4.5 Latitude and Longitude 12](#_Toc520040113)

[4.6 Time 12](#_Toc520040114)

[4.7 Temperature 12](#_Toc520040115)

[4.8 Euler Angles 12](#_Toc520040116)

[4.9 Decibels 12](#_Toc520040117)

[4.10 Frequency 13](#_Toc520040118)

[4.11 Unitless 13](#_Toc520040119)

[4.12 “Good” Flags 13](#_Toc520040120)

[4.13 Data Rate 13](#_Toc520040121)

# 1 Packet Header

The function of the packet header is to convey the recipient, sender, and type of each packet. Specific command IDs for specific recipients use multipacket as the command is transferring a file larger than one packet can fit. See command lists for which ones are multipacket.

The header is two bytes long. The first byte contains the recipient, sender, and command ID, see bit allocation in table 1. The second byte is the length of the packet in number of dwords (), including header, minus one. A value of 0x00 indicates the packet is long. The choice of dwords allows packet lengths up to , usable data is . A packet might require padding at the end of the body to increase the body length to a multiple of single dwords, this padding should be zeros.

Table 1: Packet Header First Byte

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Bit index | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Function | Sender  0x0: Ground  0x1: CougSat-1  ~~0x2: CougSat-2~~1  ~~0x3: CougSat-3~~1  ~~0x4: CougSat-4~~1  ~~0x5: CougSat-5~~1  ~~0x6: CougSat-6~~1  ~~0x7: CougSat-7~~1 | | | Recipient  0x0: ADCS  0x1: IFJR  0x2: IHU  0x3: PMIC  0x4: Comms  0x5: Payload 1  0x6: Payload 2  0x7: Payload 3 | | | Command ID  Up to 4 commands per recipient  See [Section 2](#_2_Commands) for list of commands | |

1 These CougSats don’t exist yet

## 1.1 Multipacket Additional Header

For a multipacket, there are an additional two bytes appended to the packet header. These represent a 16b long serial number for the packet. By analyzing the serial numbers of all packets received for a multipacket, the recipient can figure which packets were lost and request those specific packets from the sender.

# 2 Packet Body

The packet body takes two distinct forms: a single packet, or multipacket. The different forms are described below.

## 2.1 Single Packet

In a single packet, the data directly follows the packet header. This data is specific to the command, this data is described in [section 3](#_3_Commands) under each command.

## 2.2 Multipacket

The function of a multipacket is to transfer data that is longer than a single packet can support. The primary uses of a multipacket are images and processor binaries. Every packet, except the last packet, will be completely full. There is a low chance that the last packet is also completely full. From the file size given in the first packet and the knowledge on the length of packets, the recipient can immediately figure the number of packets it is going to receive and how full the last packet is going to be.

### 2.1.1 First Packet

The first packet provides details on the file being transferred, including length and file name. The first 4 bytes represents the file length as an unsigned integer, file sizes are limited to . The next five bytes are a cyclic redundancy check (CRC32) on the file contents used to verify validity. The next several bytes, until a null character is reached, represents the filename as a string. Immediately following the null character is the start of the file.

### 2.1.2 Second to (n – 1)th Packet

The second and onward packet, to one before the last, just contains the data of the file.

### 2.1.2 nth Packet

The last packet also contains just the data of the file; however, there is a likely possibility that this packet will not be completely full. If so, the same padding rules apply as normal packets: add padding until the body is an even multiple of dwords. Once this packet is received, or communication times out, the recipient will identify which packets were lost in transmission and request them again using their serial numbers

### 2.1.3 Lost Packet Request

If a packet is lost, the recipient will send a request for retransmission of that specific packet. The body of this message will be a byte to indicate lost packets (0xFF) and the serial numbers of lost packets sequentially appended. If number of lost packets exceeds the number that fit in a single packet, send additional requests. Once all packets are accounted for, send a multipacket transfer success message and began assembling packets.

# 3 Commands

Every command has a specific format which is detailed here. In the byte table, the first two byte are the packet header outlined in [section 1](#_1_Packet_Header). The first byte is bitwise ORed with the satellite ID when sent from a CougSat, only the five least bits are listed in the byte tables.

## 3.1 ADCS

## 3.2 IFJR

### 3.2.1 Processor Binary Image Upload

A binary image is sent by the Ground with command ID 0, see the table below. The file is sent using multipacket, see [section 2.2](#_2.2_Multipacket). The IFJR stores this file indefinitely. The IFJR expects the following file name:

[Processor].[Major].[Minor].[Patch].bin

e.g. “IHU.1.0.5.bin”

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Byte Offset | 0x00 | 0x01 | 0x02 | 0x03 | 0x04 | 0x05 | 0x06 | 0x07 |
| 0x0000 | 0x04 | Packet Length | Serial Number | | Data | Data | Data | Data… |

After the multipacket transfer, the IFJR sends a reply with command ID 0, see the table below.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Byte Offset | 0x00 | 0x01 | 0x02 | 0x03 | 0x04 | 0x05 | 0x06 | 0x07 |
| 0x0000 | 0x04 | Sat ID | 0x01 | File Good | CRC32 using 0x04C11DB7 | | | | Empty Pad |

### 3.2.2 Reprogram a Device

A request for a reprogramming of a device is sent by the Ground with command ID 1, see the table below.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Byte Offset | 0x00 | 0x01 | 0x02 | 0x03 | 0x04 | 0x05 | 0x06 | 0x07 |
| 0x0000 | 0x05 | 0x01 | Device ID1 | Ver. Major | Ver. Minor | Ver. Patch | Empty Padding | |

1 See the table below for a list of device IDs

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0x00 | ADCS | 0x01 | IFJR | 0x02 | IHU | 0x03 | PMIC |
| 0x04 | Comms |  |  |  |  |  |  |

After the programming, the IFJR sends a reply with command ID 1, see the table below.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Byte Offset | 0x00 | 0x01 | 0x02 | 0x03 | 0x04 | 0x05 | 0x06 | 0x07 |
| 0x0000 | 0x05 | Sat ID | 0x01 | Device Good | Programming Duration ( | | | Empty Padding | |

## 3.3 IHU

### 3.3.1 Telemetry

A request for a telemetry packet is sent by the Ground with command ID 0, see the table below. This packet contains data on every subsystem of the satellite bus (no payload). This packet is remade and sent at regular intervals during the IHU’s periodic event. A request for a telemetry packet is normally not required.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Byte Offset | 0x00 | 0x01 | 0x02 | 0x03 | 0x04 | 0x05 | 0x06 | 0x07 |
| 0x0000 | 0x08 | 0x00 | Empty Padding | |  |  |  |  |

A telemetry packet is prepared and sent by the IHU with command ID 0, see the table below.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Byte Offset | 0x00 | 0x01 | 0x02 | 0x03 | 0x04 | 0x05 | 0x06 | 0x07 |
| 0x0000 | 0x08 | Sat ID | 0x31 | Mode | IHU Temp | Current Time | | | |
| 0x0008 | SD Card Used Size | | | | | Reset Count | Error Status | ADCS Temp |
| 0x0010 | Latitude | | | | Longitude | | | |
| 0x0018 | Roll | | Pitch | | Yaw | | X PWM Out | |
| 0x0020 | Y PWM Out | | Z PWM Out | | X Current | | Y Current | |
| 0x0028 | Z Current | | IFJR Temp | PMIC Temp | Batt A Temp | Batt B Temp | 3.3V A Temp | 3.3V B Temp |
| 0x0030 | PV 0 Temp | PV 1 Temp | PV 2 Temp | PV 3 Temp | PV 4 Temp | PV 5 Temp | PV 6 Temp | PV 7 Temp |
| 0x0038 | MPPT 0 Temp | MPPT 1 Temp | MPPT 2 Temp | MPPT 3 Temp | MPPT 4 Temp | MPPT 5 Temp | MPPT 6 Temp | MPPT 7 Temp |
| 0x0040 | PV 0 Voltage | | PV 0 Current | | PV 1 Voltage | | PV 1 Current | |
| 0x0048 | PV 2 Voltage | | PV 2 Current | | PV 3 Voltage | | PV 3 Current | |
| 0x0050 | PV 4 Voltage | | PV 4 Current | | PV 5 Voltage | | PV 5 Current | |
| 0x0058 | PV 6 Voltage | | PV 6 Current | | PV 7 Voltage | | PV 7 Current | |
| 0x0060 | Battery A Voltage | | Battery A Current | | Battery B Voltage | | Battery B Current | |
| 0x0068 | 3.3V Regulator A Voltage | | 3.3V Regulator A Current | | 3.3V Regulator B Voltage | | 3.3V Regulator B Current | |
| 0x0070 | PR\_3.3V-0 Current | | PR\_3.3V-1 Current | | PR\_3.3V-2 Current | | PR\_3.3V-3 Current | |
| 0x0078 | PR\_3.3V-4 Current | | PR\_3.3V-5 Current | | PR\_3.3V-6 Current | | PR\_3.3V-7 Current | |
| 0x0080 | PR\_3.3V-8 Current | | PR\_3.3V-9 Current | | PR\_3.3V-10 Current | | PR\_3.3V-11 Current | |
| 0x0088 | PR\_3.3V-12 Current | | PR\_BATT-0 Current | | PR\_BATT-1 Current | | PR\_BATT-2 Current | |
| 0x0090 | PR\_BATT-3 Current | | PR\_BATT-4 Current | | PR\_BATT-5 Current | | PR\_BATT-6 Current | |
| 0x0098 | PV\_3.3-0 Current | | PV\_3.3-1 Current | | PV\_3.3-2 Current | | PV\_3.3-3 Current | |
| 0x00A0 | PR\_BH-0 Current | | PR\_BH-1 Current | | PR\_DEPLOY Current | | PV Switching State1 | |
| 0x00A8 | Output Switching State1 | | | | | | | Energy Level |
| 0x00B0 | Comms Temp | RX Temp | TX Temp | Amp Temp | RX Power | | RX Signal to Noise Ratio | |
| 0x00B8 | Bad Packet Count | | RX Center Frequency | | | TX Center Frequency | | |
| 0x00C0 | TX Power | | TX Amplifier Voltage | |  |  |  |  |

1 See PMIC switching states in its documentation for conversion details

## 3.4 PMIC

## 3.5 Comms

### 3.5.1 Radio Configuration

A change to the radio configuration is sent by the Ground with command ID 0, see the table below.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Byte Offset | 0x00 | 0x01 | 0x02 | 0x03 | 0x04 | 0x05 | 0x06 | 0x07 |
| 0x0000 | 0x10 | 0x05 | TX Baseband Frequency | | | RX Baseband Frequency | | |
| 0x0008 | CW TX Power | | Voice Beacon TX Power | | High Speed TX Power | | CW Data Rate | |
| 0x0010 | Voice Beacon Data Rate | | High Speed Data Rate | |  |  |  |  |

A reply is generated and sent by the Comms with command ID 0, see the table below.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Byte Offset | 0x00 | 0x01 | 0x02 | 0x03 | 0x04 | 0x05 | 0x06 | 0x07 |
| 0x0000 | 0x10 | Sat ID | 0x05 | RX Good | TX Good |  |  |  |  |

## 3.6 Payload 1

CougSat-1’s Payload 1 is health data logging

### 3.6.1 Configuration

A change to Payload 1’s operation is sent by the Ground with command ID 0, see the table below. The file is sent using multipacket, see [section 2.2](#_2.2_Multipacket). The file is specific to the payload, see Payload 1 documentation for details.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Byte Offset | 0x00 | 0x01 | 0x02 | 0x03 | 0x04 | 0x05 | 0x06 | 0x07 |
| 0x0000 | 0x14 | Packet Length | Serial Number | | Data | Data | Data | Data… |

After the multipacket transfer, Payload 1 sends a reply with command ID 0, see the table below.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Byte Offset | 0x00 | 0x01 | 0x02 | 0x03 | 0x04 | 0x05 | 0x06 | 0x07 |
| 0x0000 | 0x14 | Sat ID | 0x01 | File Good | CRC32 using 0x04C11DB7 | | | | Empty Pad |

### 3.6.2 Data Download

A request for the next enqueued data package is sent by the Ground with command ID 1, see the table below. The file is sent using multipacket, see [section 2.2](#_2.2_Multipacket). If Ack is a 0, the rest of the message will be ignored and Payload 1 will start a multipacket transfer of the next data. If Ack is a 1, the status flag is success, and the CRC32 is valid, the file was successfully downloaded, and it will be removed from the queue.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Byte Offset | 0x00 | 0x01 | 0x02 | 0x03 | 0x04 | 0x05 | 0x06 | 0x07 |
| 0x0000 | 0x15 | 0x01 | Ack | File Good | CRC32 using 0x04C11DB7 | | | |

If ACK is a 0, Payload 1 will send a multipacket with command ID 1, see the table below.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Byte Offset | 0x00 | 0x01 | 0x02 | 0x03 | 0x04 | 0x05 | 0x06 | 0x07 |
| 0x0000 | 0x15 | Sat ID | Packet Length | Serial Number | | Data | Data | Data | Data… |

## 3.7 Payload 2

CougSat-1’s Payload 2 is a ground facing camera.

### 3.7.1 Configuration

A change to Payload 2’s operation is sent by the Ground with command ID 0, see the table below. The file is sent using multipacket, see [section 2.2](#_2.2_Multipacket). The file is specific to the payload, see Payload 2 documentation for details.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Byte Offset | 0x00 | 0x01 | 0x02 | 0x03 | 0x04 | 0x05 | 0x06 | 0x07 |
| 0x0000 | 0x18 | Packet Length | Serial Number | | Data | Data | Data | Data… |

After the multipacket transfer, Payload 1 sends a reply with command ID 0, see the table below.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Byte Offset | 0x00 | 0x01 | 0x02 | 0x03 | 0x04 | 0x05 | 0x06 | 0x07 |
| 0x0000 | 0x18 | Sat ID | 0x01 | File Good | CRC32 using 0x04C11DB7 | | | | Empty Pad |

### 3.7.2 Data Download

A request for the next enqueued data package is sent by the Ground with command ID 1, see the table below. The file is sent using multipacket, see [section 2.2](#_2.2_Multipacket). If Ack is a 0, the rest of the message will be ignored and Payload 2 will start a multipacket transfer of the next data. If Ack is a 1, the status flag is success, and the CRC32 is valid, the file was successfully downloaded and it will be removed from the queue.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Byte Offset | 0x00 | 0x01 | 0x02 | 0x03 | 0x04 | 0x05 | 0x06 | 0x07 |
| 0x0000 | 0x19 | 0x01 | Ack | File Good | CRC32 using 0x04C11DB7 | | | |

If ACK is a 0, Payload 2 will send a multipacket with command ID 1, see the table below.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Byte Offset | 0x00 | 0x01 | 0x02 | 0x03 | 0x04 | 0x05 | 0x06 | 0x07 |
| 0x0000 | 0x19 | Sat ID | Packet Length | Serial Number | | Data | Data | Data | Data… |

## 3.8 Payload 3

CougSat-1’s Payload 3 is a germination experiment

### 3.8.1 Configuration

A change to Payload 3’s operation is sent by the Ground with command ID 0, see the table below. The file is sent using multipacket, see [section 2.2](#_2.2_Multipacket). The file is specific to the payload, see Payload 3 documentation for details.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Byte Offset | 0x00 | 0x01 | 0x02 | 0x03 | 0x04 | 0x05 | 0x06 | 0x07 |
| 0x0000 | 0x1C | Packet Length | Serial Number | | Data | Data | Data | Data… |

After the multipacket transfer, Payload 3 sends a reply with command ID 0, see the table below.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Byte Offset | 0x00 | 0x01 | 0x02 | 0x03 | 0x04 | 0x05 | 0x06 | 0x07 |
| 0x0000 | 0x1C | Sat ID | 0x01 | File Good | CRC32 using 0x04C11DB7 | | | | Empty Pad |

### 3.8.2 Data Download

A request for the next enqueued data package is sent by the Ground with command ID 1, see the table below. The file is sent using multipacket, see [section 2.2](#_2.2_Multipacket). If Ack is a 0, the rest of the message will be ignored and Payload 3 will start a multipacket transfer of the next data. If Ack is a 1, the status flag is success, and the CRC32 is valid, the file was successfully downloaded and it will be removed from the queue.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Byte Offset | 0x00 | 0x01 | 0x02 | 0x03 | 0x04 | 0x05 | 0x06 | 0x07 |
| 0x0000 | 0x1D | 0x01 | Ack | File Good | CRC32 using 0x04C11DB7 | | | |

If ACK is a 0, Payload 3 will send a multipacket with command ID 1, see the table below.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Byte Offset | 0x00 | 0x01 | 0x02 | 0x03 | 0x04 | 0x05 | 0x06 | 0x07 |
| 0x0000 | 0x3D | Sat ID | Packet Length | Serial Number | | Data | Data | Data | Data… |

# 4 Units

When a number is sent down, it is raw byte data that needs to be reassembled and converted to appropriate units. Below is a list of all numbers transferred and how to convert their byte data to a real number. Unless otherwise specified, these conversion factors are used for every value of the appropriate type.

All values use big-endian byte ordering. The most significant byte is located in the first index.

## 4.1 Voltage

Unsigned integer with

## 4.2 Current

Signed integer with

## 4.3 Power

Unsigned integer with

## 4.4 Energy

Unsigned integer with

## 4.5 Latitude and Longitude

Signed integer with

## 4.6 Time

Unsigned integer, UNIX Epoch time

## 4.7 Temperature

Signed integer with

## 4.8 Euler Angles

Unsigned integer with

## 4.9 Decibels

Signed integer with

## 4.10 Frequency

Unsigned integer with

## 4.11 Unitless

Unsigned integer (variable bit length) with

## 4.12 “Good” Flags

Unsigned integer representing an error code, see [Code SOP, section 6.4.1](https://github.com/CougsInSpace/Resources/tree/master/StandardOperatingProcedures)

## 4.13 Data Rate

Unsigned integer with