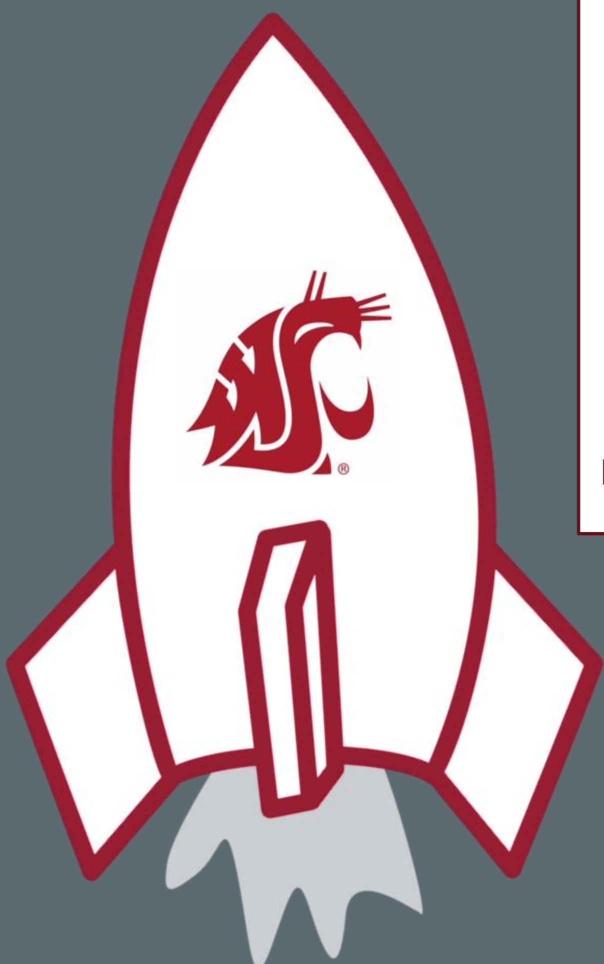
The communication between the ground and satellite is separated into packets. This document specifies the form and function of these packets.

# Packet Protocol

Detailed Description of Communication Packet

Revision: 1.1.0


Bradley Davis

# Table of Contents

# 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 ($32b$), including header. The choice of dwords allows packet lengths up to $2^8 \cdot 32b = 1024B$, usable data is $1022B$. 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<br>`0x0`: Ground<br>`0x1`: CougSat-1<br>`0x2`: CougSat-2<br>`0x3`: CougSat-3<br>`0x4`: CougSat-4<br>`0x5`: CougSat-5<br>`0x6`: CougSat-6<br>`0x7`: CougSat-7 | | | Recipient<br>`0x0`: ADCS<br>`0x1`: IFJR<br>`0x2`: IHU<br>`0x3`: PMIC<br>`0x4`: RCS<br>`0x5`: Payload 1<br>`0x6`: Payload 2<br>`0x7`: Payload 3 | | | Command ID<br><br>Up to 4 commands per recipient<br><br>See Section 2 for list of commands | |

## 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 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 $2^{32}b \approx 4GB$. 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 n$^{th}$ 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.

### 3.1 ADCS

### 3.2 IFJR

### 3.3 IHU

#### 3.3.1 Telemetry

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

<table>
<tr><th>Byte Offset</th><th>0x00</th><th>0x01</th><th>0x02</th><th>0x03</th><th>0x04</th><th>0x05</th><th>0x06</th><th>0x07</th></tr>
<tr><td>0x0000</td><td>0x28</td><td>0x31</td><td>Mode</td><td>IHU Temp</td><td colspan="4">Current Time</td></tr>
<tr><td>0x0008</td><td colspan="5">SD Card Used Size</td><td>Reset Count</td><td>Error Status</td><td>ADCS Temp</td></tr>
<tr><td>0x0010</td><td colspan="4">Latitude</td><td colspan="4">Longitude</td></tr>
<tr><td>0x0018</td><td colspan="2">Roll</td><td colspan="2">Pitch</td><td colspan="2">Yaw</td><td colspan="2">X PWM Out</td></tr>
<tr><td>0x0020</td><td colspan="2">Y PWM Out</td><td colspan="2">Z PWM Out</td><td colspan="2">X Current</td><td colspan="2">Y Current</td></tr>
<tr><td>0x0028</td><td colspan="2">Z Current</td><td>IFJR Temp</td><td>PMIC Temp</td><td>Batt A Temp</td><td>Batt B Temp</td><td>3.3V A Temp</td><td>3.3V B Temp</td></tr>
<tr><td>0x0030</td><td>PV 0 Temp</td><td>PV 1 Temp</td><td>PV 2 Temp</td><td>PV 3 Temp</td><td>PV 4 Temp</td><td>PV 5 Temp</td><td>PV 6 Temp</td><td>PV 7 Temp</td></tr>
<tr><td>0x0038</td><td>MPPT 0 Temp</td><td>MPPT 1 Temp</td><td>MPPT 2 Temp</td><td>MPPT 3 Temp</td><td>MPPT 4 Temp</td><td>MPPT 5 Temp</td><td>MPPT 6 Temp</td><td>MPPT 7 Temp</td></tr>
<tr><td>0x0040</td><td colspan="2">PV 0 Voltage</td><td colspan="2">PV 0 Current</td><td colspan="2">PV 1 Voltage</td><td colspan="2">PV 1 Current</td></tr>
<tr><td>0x0048</td><td colspan="2">PV 2 Voltage</td><td colspan="2">PV 2 Current</td><td colspan="2">PV 3 Voltage</td><td colspan="2">PV 3 Current</td></tr>
<tr><td>0x0050</td><td colspan="2">PV 4 Voltage</td><td colspan="2">PV 4 Current</td><td colspan="2">PV 5 Voltage</td><td colspan="2">PV 5 Current</td></tr>
<tr><td>0x0058</td><td colspan="2">PV 6 Voltage</td><td colspan="2">PV 6 Current</td><td colspan="2">PV 7 Voltage</td><td colspan="2">PV 7 Current</td></tr>
<tr><td>0x0060</td><td colspan="2">Battery A Voltage</td><td colspan="2">Battery A Current</td><td colspan="2">Battery B Voltage</td><td colspan="2">Battery B Current</td></tr>
</table>

<table>
<tr><td>0x0068</td><td colspan="2">3.3V Regulator A Voltage</td><td colspan="2">3.3V Regulator A Current</td><td colspan="2">3.3V Regulator B Voltage</td><td colspan="2">3.3V Regulator B Current</td></tr>
<tr><td>0x0070</td><td colspan="2">PR_3.3V-0 Current</td><td colspan="2">PR_3.3V-1 Current</td><td colspan="2">PR_3.3V-2 Current</td><td colspan="2">PR_3.3V-3 Current</td></tr>
<tr><td>0x0078</td><td colspan="2">PR_3.3V-4 Current</td><td colspan="2">PR_3.3V-5 Current</td><td colspan="2">PR_3.3V-6 Current</td><td colspan="2">PR_3.3V-7 Current</td></tr>
<tr><td>0x0080</td><td colspan="2">PR_3.3V-8 Current</td><td colspan="2">PR_3.3V-9 Current</td><td colspan="2">PR_3.3V-10 Current</td><td colspan="2">PR_3.3V-11 Current</td></tr>
<tr><td>0x0088</td><td colspan="2">PR_3.3V-12 Current</td><td colspan="2">PR_BATT-0 Current</td><td colspan="2">PR_BATT-1 Current</td><td colspan="2">PR_BATT-2 Current</td></tr>
<tr><td>0x0090</td><td colspan="2">PR_BATT-3 Current</td><td colspan="2">PR_BATT-4 Current</td><td colspan="2">PR_BATT-5 Current</td><td colspan="2">PR_BATT-6 Current</td></tr>
<tr><td>0x0098</td><td colspan="2">PV_3.3-0 Current</td><td colspan="2">PV_3.3-1 Current</td><td colspan="2">PV_3.3-2 Current</td><td colspan="2">PV_3.3-3 Current</td></tr>
<tr><td>0x00A0</td><td colspan="2">PR_BH-0 Current</td><td colspan="2">PR_BH-1 Current</td><td colspan="2">PR_DEPLOY Current</td><td colspan="2">PV Switching State[1]</td></tr>
<tr><td>0x00A8</td><td colspan="7">Output Switching State[1]</td><td>Energy Level</td></tr>
<tr><td>0x00B0</td><td>RCS Temp</td><td>RX Temp</td><td>TX Temp</td><td>Amp Temp</td><td colspan="2">RX Power</td><td colspan="2">RX Signal to Noise Ratio</td></tr>
<tr><td>0x00B8</td><td colspan="2">Bad Packet Count</td><td colspan="3">RX Center Frequency</td><td colspan="3">TX Center Frequency</td></tr>
<tr><td>0x00C0</td><td colspan="2">TX Power</td><td colspan="2">TX Amplifier Voltage</td></tr>
</table>

[1] See PMIC switching states for conversion details

### 3.4 PMIC

### 3.5 RCS

### 3.6 Payload 1

### 3.7 Payload 2

### 3.8 Payload 3

# 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.

## 4.1 Voltage

Unsigned $16b$ integer with $100\mu\frac{V}{LSB} = 0V\ to\ 6.55V$

## 4.2 Current

Signed $16b$ integer with $150\mu\frac{A}{LSB} = \pm 4.92A$

## 4.3 Power

Unsigned $16b$ integer with $250\mu\frac{W}{LSB} = 0W\ to\ 6.4W$

## 4.4 Energy

Unsigned $8b$ integer with $500\frac{J}{LSB} = 0J\ to\ 127kJ$

## 4.5 Latitude and Longitude

Signed $32b$ integer with $10\mu\frac{min}{LSB} = \pm 358°$

## 4.6 Time

Unsigned $32b$ integer, UNIX Epoch time

## 4.7 Temperature

Signed $8b$ integer with $1\frac{°C}{LSB} = -128\ to\ 127°C$

## 4.8 Euler Angles

Unsigned $16b$ integer with $2^{16} = 360°$

## 4.9 Decibels

Signed $16b$ integer with $1m\frac{dB}{LSB} = \pm 32.8dB$

## 4.10 Unitless

Unsigned integer (variable bit length) with $1\frac{ul}{LSB}$