# SUNCQ Protocol

## Description

This document defines the Stellenbosch University Node Control for Qubes (SUNCQ) protocol. It lays out the commands to communicate with a Stellenbosch University PocketQube Ground Station via a TNC. A TNC (Terminal Node Controller) is a serial interface system which allows for control over a ground station or radio. This protocol defines the messages to and from the TNC and the computer (referred to as the host). Its current implementation requires commands to not be sent more frequently than once every second. It was authored by Gary Allen, a 2023 E&E student.

## Commands

The following lists all commands from TNC to host and vica-versa. The command reservations are:

* 0x00 to 0x2F Host-to-TNC DO commands
* 0x30 to 0x5F Host-to-TNC SET commands
* 0x60 to 0x7F Host-to-TNC GET commands
* 0x80 to 0x9F TNC-to-Host STATUS replies
* 0xA0 to 0xCF TNC-to-Host DATA replies
* 0xD0 to 0xFE Reserved
* 0xFF Invalid

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Code** | **Name** | **Function** | **Payload** | **Bytes** | **Comments** |
| 0x00 | RESET | Reset the system | - | 0 |  |
| 0x01 | CALIBRATE | Calibrate the system | - | 0 | Full calibration e.g. ground station and all sub-systems |
| 0x02 | RETURN\_TO\_START | Return the system to its starting state. | - | 0 | The starting state is post-calibration. |
| 0x03 | RETURN\_TO\_STOW | Return the system to its stow state. |  |  | The stow state is pre-calibration. Typically used before system shutdown |
| 0x30 | SET\_TNC\_MODE | Enter a certain mode | See *TNC\_MODE* | 1 |  |
| 0x31 | SET\_TRACK\_MODE | Set tracking mode | See *TRACK\_MODE* | 1 |  |
| 0x32 | SET\_PATH\_DATA | Upload flight path data | CSV file. See *Flight Path Data.* | Any | The payload length is provided as the first 8 bytes. |
| 0x33 | SET\_TRACK\_LOCATION | Set point direction, or add a received GPS location (depending on the track mode) | See *Flight Path Data* (pass only one entry) |  |  |
| 0x34 | SET\_TRACK\_TARGET | Set the target to track | See *TRACK\_TARGET* | 1 | Allows for the custom internal target, or any external target, to be tracked (in which case the internal receiver is disbable) |
| 0x60 | GET\_SIGNAL\_RSSI | Get RSSI of the signal | - | 0 |  |
| 0x61 | GET\_LOCATION | Get location of the ground station | Lat;Lng;Alt (f32;f32;f32) | 12 |  |
| 0x80 | TNC\_TELEMETRY | Sent by the TNC to send telemetry to the host | See *STATUS\_CODE.* | 1 | Status 0x00 is used as an “ACK” command. |
| 0x81 | TNC\_MESSAGE | Sent by the TNC to communicate a log message to the host. | char[] | Any | A newline character terminates the message |
| 0xA0 | SIGNAL\_RSSI | Response to *GET\_SIGNAL\_RSSI* | float | 4 |  |
| 0xD0-0xFE | RESERVED | Reserved |  |  | Reserved for future use |
| 0XFF | INVALID | Invalid |  |  | An invalid command that can be used internally but never sent |

## Details

#### TNC\_MODE

|  |  |
| --- | --- |
| **Value** | **Description** |
| 0x00 | Normal mode |
| 0x01 | KISS mode. This mode is exited using the KISS 0xFF command. |

#### TRACK\_MODE

The tracking mode is a combination flag i.e. multiple bits can be ORed together to specify that the payload must be tracked using multiple methods at once.

|  |  |
| --- | --- |
| **Value** | **Description** |
| 0x00 | No tracking. Mount can be moved by setting the pointing vector. |
| 0x01 | Use uploaded GPS data |
| 0x02 | Use received GPS data (from payload) |
| 0x04 | Use signal strength, but only for an initial scan |
| 0x08 | Use signal strength, with dynamic conical scanning |

#### TRACK\_TARGET

The tracking mode is a combination flag i.e. multiple bits can be ORed together to specify that the payload must be tracked using multiple methods at once.

|  |  |
| --- | --- |
| **Value** | **Description** |
| 0x00 | Internal |
| 0x01 | External |

#### STATUS\_CODE

The following is a list of status codes that might be sent from the TNC to the host:

|  |  |
| --- | --- |
| **Value** | **Description** |
| 0x00 | Acknowledge |
| 0x01 | Payload tracking unsuccessful/payload lost |

#### Flight path data

Flight path data can be uploaded in the form of a little-endian binary stream (i.e. each field should be least significant byte first). The first field is a 2-byte number indicating the number of flight path instances to follow. Then, the fields below should be provided. Such a file can be generated by predicting a flight path at <https://predict.sondehub.org/> , generating a CSV file, and then using the data to generated a binary stream. In the current implementation, only 200 entries are catered for – if longer flights are needed, then multiple streams should be set from the host intermittently. Time should be in Unix time (seconds since Epoch).

|  |  |
| --- | --- |
| **Name** | **Type** |
| Time | uint64 |
| Latitude | float32 |
| Longitude | float32 |
| Altitude | float32 |