CySat Command and Data Protocol Description

Jake Drahos
drahos@iastate.edu

December 19, 2014
v0.3

1 Introduction

This document is the canonical reference for the communications protocol used for CySat I and future satellites for the CySat project. This document is currently a working draft. As such, it is incomplete. Additions will be made, however, major changes to already established details will not occur except in large version number upgrades of this document (for example, version 0.7 to 1.1, but not from 0.4 to 0.5). Version history can be found in the last section of this document.

2 Protocol Overview

The protocol is based on the idea of *messages*. Messages are entirely in ASCII, with the exception of `DOWNLINK` messages, which contain a binary field. Each message has several *fields*, each of which is separated by a comma. Every message begins with a start character, and ends with a stop character.

The communications protocol involves several types of message:

Message Type	Usage	Origin
`QUERY`	Query the satellite	Base
`RESULT`	Contains return information from a query	Satellite
`COMMAND`	Issues a command to the satellite	Base
`ACK_COMMAND`	Acknowledges that a command was performed	Satellite
`DOWNLINK`	Downlink packet	Satellite
`ACK_DOWNLINK`	Acknowledges downlink packet	Base
`NACK_ERROR`	Acknowledges an invalid packet	Satellite

3 Message String Format

A valid command string takes the following format:

```
!MESSAGE_TYPE,FIELD_1,FIELD_2,CHECKSUM$
```

There are three reserved characters, which cannot appear anywhere in an ASCII command:

- `!` — The start character
- `$` — The stop character
- `,` — Field separator

Each different message type has different requirements for valid lengths.

Message Type	Minimum Fields	Count
`QUERY`	Type, Query Subtype, Checksum	3
`COMMAND`	Type, Command Subtype, Checksum	3
`DOWNLINK`	Type, ID, Size, Data, Checksum	5
`RESULT`	Type, Query Subtype, Result, Checksum	4
`ACK_COMMAND`	Type, Command Subtype, Checksum	3
`ACK_DOWNLINK`	Type, ID, Checksum	3
`NACK_ERROR`	Type, Subtype, Checksum	3

Note that these are minimum required fields. Some messages, such as `QUERY` and `COMMAND` may have different required parameters depending on the exact command or query being executed.

The first field of any message is known as the "Type" field, and corresponds to the message type. The second field is often the message "subtype". For `QUERY` and `COMMAND` messages, this is the 'command' or 'query' being performed. For `RESULT` and `ACK_COMMAND` messages, this corresponds to the subtype of the message being acknowledged or the query being answered. The last field is always a checksum.

The checksum is a one-byte ASCII-Hexadecimal pair. Any applicable Message ID fields are two-byte ASCII-Hexadecimal values (two hex pairs).

4 The Checksum

The checksum used is a NMEA checksum of all characters beginning with the `!` up to the last `,` preceeding the checksum. The start character **IS** included in the checksum. The last separator (`,`) **IS** also included in the checksum. In this sense, the checksum includes every character preceeding it in the entire message. The process of calculating the checksum is to merely `XOR` each byte with the previous byte. The calculated checksum is one byte encoded in ASCII-Hexadecimal. It immediately follows the last separator and preceeds the stop character.

NOTE: In this document, the example checksums used will be a random hex pair and are not actually valid.

5 QUERY Messages

The purpose of a QUERY message is to allow the base station to request specific information from the satellite. Each query message has a corresponding RESULT message. The subtype name of the corresponding RESULT is the same as the subtype name of the QUERY .

All QUERY messages require a Type, Subtype, and Checksum field. Some messages require additional fields following the subtype and preceeding the checksum.

Subtype	Definition	Additional Fields	Total Field Count
HELLO	Hello World	–	3
POW_PANEL	Panel information query	Axis	4
POW_BUS	Power bus information query	–	3
POW_BATTERY	Battery information query	Battery	4
FOOTPRINTS	Number of stored footprints	–	3
TIME	Current mission time	–	3

5.1 QUERY HELLO

A QUERY HELLO Message is a QUERY request for a Hello World response from the satellite. This is the most basic message exchange with the satellite. An example exchange is below:

The base station will send a request QUERY message:

```
!QUERY,HELLO,A0$
```

The satellite will now send a reply RESULT message:

```
!RESULT,HELLO,Hello World,C2$
```

5.2 QUERY POW_PANEL

A QUERY POW_PANEL message requests power information for one of the three axis panels. There is one additional field required for a QUERY POW_PANEL query: the axis field. The axis field is one of three valid axes:

- X
- Y
- Z

```
Fields:  Type, Subtype, Axis, Checksum
Example:  !QUERY,POW_PANEL,X,1C$
```

5.3 QUERY POW_BUS

This query requests bus information from the EPS. It has no additional fields.

```
Fields:  Type, Subtype, Checksum
Example:  !QUERY,POW_BUS,2D$
```

5.4 QUERY POW_BATTERY

This query requests information on one of two batteries. It has one additional field to determine which battery should be reported. Valid values for the Battery field are:

- 0
- 1

```
Fields:  Type, Subtype, Battery, Checksum
Example:  !QUERY,POW_BATTERY,0,3D$
```

5.5 QUERY FOOTPRINTS

Requests the number of stored footprints for downlink.

`Fields: Type, Subtype, Checksum`

`Example: !QUERY,FOOTPRINTS,4E$`

5.6 QUERY TIME

Requests the mission time.

`Fields: Type, Subtype, Checksum`

`Example: !QUERY,TIME,5F$`

5.7 QUERY Exchange Details

A `QUERY` exchange is a two-step exchange. First, the base station sends a `QUERY` message and waits. Upon receiving and validating a `QUERY` message, the satellite will compose and transmit a `RESULT` message.

The satellite **IS NOT** permitted to transmit multiple `RESULT` messages per `QUERY` received.

The base station **IS** permitted to retry `QUERY` messages until a `RESULT` is received.

6 RESULT Messages

RESULT messages are sent by the satellite in response to a query of the same subtype name. Each result message contains at least four fields. The Type, Subtype, and Checksum fields are required, and all RESULT messages have at least one additional field containing the actual result data. Certain result subtypes may return more than one data field. The meanings of these additional fields will be described in each RESULT section. All data fields directly preceed the checksum and are in the listed order.

Subtype	Definition	Data Fields	Total field count
HELLO	Hello World	"Hello World"	4
POW_PANEL	Panel status	Axis, Voltage, -Current, +Current	7
POW_BUS	Bus status	Battery Current, 5V Current, 3.3V Current	6
POW_BATTERY	Battery status	Battery, Temperature, Voltage, Direction, Current	8
FOOTPRINTS	Number of stored footprints	Footprints	4
TIME	Mission time	Time	4

6.1 RESULT HELLO

The HELLO subtype is a response to a "Hello World" request from the base station. The single data field of a HELLO result always contains the string "Hello World".

```
Fields:  Type, Subtype, Hello World, Checksum
Example:  !RESULT,HELLO,Hello World,A6$
```

6.2 RESULT POW_PANEL

This result contains four data fields. The first is the axis requested in the original query. The voltage and current fields contain the corresponding power values, each as a two-byte integer encoded as two hex pairs.

```
Fields:  Type, Subtype, Axis, Voltage, -Current, +Current, Checksum
Example:  !RESULT,POW_PANEL,X,12FE,43AB,11CC,B7$
```

6.3 RESULT POW_BUS

This result contains three data fields. Each current field contains the corresponding EPS value. Each is encoded as two hex-pairs representing one 16 bit integer.

```
Fields:  Type, Subtype, Battery Current, 5V Current, 3.3V Current, Checksum
Example:  !RESULT,POW_BUS,B3D4,AA12,BB34,C8$
```

6.4 RESULT POW_BATTERY

This result contains five data fields. The first field is the battery, either 0 or 1, depending on which was requested. The current, temperature, and voltage fields are two-byte integers represented as two hex-pairs. The direction field is either D or C, for discharge or charge.

```
Fields:  Type, Subtype, Battery, Temperature, Voltage, Direction, Current, Checksum
Example:  !RESULT,POW_BATTERY,0,0013,33C4,D,11B4,E9$
```

6.5 RESULT FOOTPRINTS

Returns the number of stored footprints. This number should be a four-byte big-endian integer, returned as four consecutive hex pairs.

```
Fields:  Type, Subtype, Footprints, Checksum
Example:  !RESULT,FOOTPRINTS,004B3F4C,F0$
```

6.6 RESULT TIME

Returns the mission time. This number should be a four-byte big-endian integer, formatted as four consecutive hex pairs.

```
Fields:  Type, Subtype, Time, Checksum
Example:  !RESULT,TIME,0000443B,3A$
```

7 COMMAND Messages

COMMAND messages are sent by the ground station to request the satellite perform an action or a configuration change. Each COMMAND should be followed by a ACK_COMMAND sent from the satellite to acknowledge the command was performed or a NACK_ERROR telling why the satellite was unable to perform the command.

Subtype	Description	Additional Fields	Field Count
BURN	Deploy Antennas	–	3
POW_PRINT	Print EPS Data	–	3
DOWNLINK	Initiate a downlink session	–	3
RESET_CLOCK	Reset mission clock to zero	–	3
SET_CLOCK	Set clock to provided time	Time	4
REBOOT	Perform a reboot	–	3
REBOOT_HARD	Perform an immediate reboot	–	3

7.1 COMMAND BURN

This command manually deploys the antennas. The satellite should initiate a burn routine and immediately respond with a ACK_COMMAND acknowledging the burn

```
Fields:  Type, Subtype, Checksum
Example:  !COMMAND,BURN,4B$
```

7.2 COMMAND POW_PRINT

This is a debug command. It causes the EPS driver to print all the housekeeping datat in ASCII directly to the console.

```
Fields:  Type, Subtype, Checksum
Example:  !COMMAND,POW_PRINT,5C$
```

7.3 COMMAND DOWNLINK

This command orders the satellite to initiate a downlink session. The satellite must respond with an ACK_COMMAND before sending DOWNLINK messages. For more information, see the section titled “Downlink Protocol.”

```
Fields:  Type, Subtype, Checksum
Example:  !COMMAND,DOWNLINK,6D$
```

7.4 COMMAND RESET_CLOCK

This command resets the mission clock to zero. This should not cause the satellite to go through any first-boot initialization; it should simply set the clock to zero.

```
Fields:  Type, Subtype, Checksum
Example:  !COMMAND,RESET_CLOCK,7E$
```

7.5 COMMAND SET_CLOCK

This command sets the mission clock to the provided value. The mission clock measures seconds since mission time zero, with mission time zero being the time of first boot. The parameter should be a 4-byte big-endian number, consisting of 4 continuous hex pairs.

```
Fields:  Type, Subtype, Time, Checksum
Example:  !COMMAND,SET_CLOCK,0001B217,8F$
```

7.6 COMMAND REBOOT

The satellite must transmit a ACK_COMMAND message prior to the reboot. The satellite is permitted to finish any other operations prior to the reboot.

```
Fields:  Type, Subtype, Checksum
Example:  !COMMAND,REBOOT,A2$
```

7.7 COMMAND REBOOT_HARD

The satellite must perform a hardware reset as soon as this message is parsed, without regard for any other tasks that may be in progress.

```
Fields:  Type, Subtype, Checksum
Example:  !COMMAND,REBOOT_HARD,B3$
```

7.8 COMMAND Exchange Details

A COMMAND exchange is a two-step exchange. First, the base station issues a COMMAND message and waits to receive either a ACK_COMMAND or NACK_ERROR COMMAND. The satellite, upon receiving a COMMAND message, should either transmit a NACK_ERROR COMMAND, or perform the command and return a ACK_COMMAND .

The satellite **IS NOT** permitted to transmit multiple ACK_COMMAND messages per COMMAND received.

The base station **MAY** retry COMMAND messages, but it is recommended to QUERY the satellite if possible to determine whether the command was not received, or the ACK_COMMAND was simply missed.

8 DOWNLINK Messages

The DOWNLINK message is a special case because it is a hybrid ascii-binary format. Because of this, it is the only message with a SIZE field.

Field	Purpose	Format
Type	DOWNLINK	Text
ID	Associate DOWNLINK with ACK_DOWNLINK	Two bytes hex
Size	Size of payload	Two bytes hex
Data	Data payload	Binary data
Checksum	Message Checksum	Normal checksum format

As DOWNLINK messages contain a binary segment, the stop character is not a reliable way to determine the end of the message. Care must be taken to properly take into account the size of the data payload.

The message ID must be unique among all recent messages. For more information, see the section titled "Downlink Protocol"

```
Fields:  Type, ID, Length, Binary Data, Checksum
Example:  !DOWNLINK,013B,0010,aB!12cOb,e$h9E.e,D4$
```

9 ACK_DOWNLINK Messages

The ACK_DOWNLINK message acknowledges a DOWNLINK message and permits the satellite to discard the contained footprints or other downlink data. The ID contained in the ACK_DOWNLINK message corresponds to the received and stored DOWNLINK .

```
Fields:  Type, ID, Checksum
Example:  !ACK_DOWNLINK,013B,E5$
```

10 ACK_COMMAND Messages

ACK_COMMAND messages acknowledge that a command was received and will or has been performed. If the command cannot be performed, a NACK_ERROR should be returned instead. Similar to RESULT messages, the subtype of a ACK_COMMAND is identical to the subtype of the COMMAND sent.

Subtype
BURN
POW_PRINT
DOWNLINK
RESET_CLOCK
SET_CLOCK
REBOOT

```
Fields:  Type, Subtype, Checksum
Example:  !ACK_COMMAND,BURN,F6$
Example:  !ACK_COMMAND,SET_CLOCK,A7$
```

11 NACK_ERROR Messages

NACK_ERROR messages are sent by the satellite to inform the base station that a message was received, but an error occurred during parsing. Some NACK_ERROR messages have four fields. This one additional field is a human-readable description following the error subtype.

Subtype	Error	Description field?
TYPE	The message type could not be determined	No
SUBTYPE	The message subtype could not be determined	No
LENGTH	Invalid field count for the message type or subtype	No
CHECKSUM	Invalid or missing checksum	No
PARAM	Invalid field in QUERY or COMMAND	Yes
COMMAND	Unable to perform a command	Yes
UNSPECIFIED	Other error occurred	Yes

```
Fields:  Type, Subtype, Checksum
Example:  !NACK_ERROR,TYPE,B8$
Fields:  Type, Subtype, Description, Checksum
Example:  !NACK_ERROR,PARAM,W is not an axis,D9$
```

12 Downlink Protocol

During downlink, the satellite operates in a slightly different manner. Upon receiving a the to begin a downlink session, the satellite will begin to transmit DOWNLINK packets. The downlink session ends when one of two conditions is met: The satellite runs out of footprints, or the satellite times out on waiting for a downlink packet to be acknowledged. The details of these are covered in the subsection titled "Cleanup"

The rate of transmission is determined by the satellite. Messages should be transmitted fast enough to ensure continuous communication, but the rate of arrival of ACK_DOWNLINK messages should be taken into account to ensure that messages are not likely to be dropped.

The satellite should pull data from non-volatile memory and cache it for transmission. Should the transmission be successfully acknowledged, the data should be discarded. If the transmission times out, the data must be returned to safe storage.

12.1 Cleanup

If the session ends due to the satellite exhausting its supply of buffered data, it simply ceases transmission of DOWNLINK messages. Upon receipt of the last ACK_DOWNLINK , the satellite can safely leave downlink mode. The station can confirm that the satellite has no more data to transmit via a QUERY . Should the satellite transmit all available downlink data, but fail to receive a needed ACK_DOWNLINK , the session ends exactly as if the time-out had occurred at any other point.

Should the session end due to a time-out, the satellite immediately ceases transmission of new DOWNLINK messages and begin waiting for a time-out of any currently transmitted but not yet acknowledged messages. Any ACK_DOWNLINK messages received during this time are taken into account and handled properly. At the end of the last time-out, any unacknowledged data is safely returned to non-volatile memory and the satellite leaves downlink mode until a new downlink session is initialized.

13 Version History

0.3

Added fields and examples to each message subtype
Added definition of downlink messages and protocol
Added many commands, including time, reboot, and downlink.
Added several queries, including time and footprint queries

0.2

Created descriptions of COMMAND messages
Began table of COMMANDs
Began table of ACK_COMMANDs
COMMAND BURN added
Removed TIME queries; not possible with this RTC
Added NACK_ERROR reference
Added checksum description
Added full definitions for power related queries

0.1

Initial release.