Airbrakes Command Line Interface Reference Manual (static version)

This document explains how to use the airbrakes command line interface and provides a description of each available command.

Command Structure

All commands are prefixed with the **>** character. This is done to distinguish them from simulation data or other transmissions. Without the **>** character, the command interpreter will **ignore** the transmission. A **<** character is printed by the command interpreter following the completion of a command's execution. If the **<** character does not appear after a command is sent, it is likely that the device is in an unresponsive state.

```
>command
<
```

Commands are organized into a tree structure which works in a way that is very similar to a file system. Commands may contain zero or more other commands, called subcommands, which are acessed by writing the base command followed by the subcommand. If a command has one or more subcommands it is also called a directory. Some directories are only used to acess their subcommands and cannot be called directly. Directories which can be called directly are said to have a default command. If you try to execute a directory which does not have a default command, an error will occur.

```
>baseCommand subCommand1
This is subcommand 1
<

>baseCommand directory1 subCommand
This is subcommand 2
<
>baseCommand directory1
<[Cmd:error] No default command exists for 'directory1' directory
```

Commands may have zero or more arguments, which are values provided by the user that are used in the execution of the command. Arguments are placed after the command name and are separated by whitespace if there are multiple.

```
>command 5 -3
<

>command subCommand 0.75
<
```

There are five types of arguments that commands may require. A command will not be executed if the number or type of arguments provided to it are incorrect.

Argument Types Table:

Type	Symbol	Description	Examples	Non-Examples
integer	i	positive or negative number	-2, 4, 29	2.5, 0b000101, 0xFF, name, "-5"
unsigned	u	non-negative number; hexadecimal and binary ok	0, 12, 0xAb01, 0b01101	-5, 0.26, name, "6"
float	f	positive or negative decimal number	0.25, -1.75, 12	0xFF, name, "0.75"
word	w	text containing no whitespace	name, 0.5, 7, -6	"string data"
string	s	sequence of characters enclosed in quotes	"string data", "0.75"	5, name

Navigation Commands

The working directory is the directory whose subcommands are treated as the base commands. By default, the working directory will be set to the "root" directory, which is the root of the tree structure containing all of the commands. The working directory can be changed to any other directory in the command structure. There are a few builtin commands which are used to navigate the directories. These are called navigation commands.

Navigation Commands Table

Name	Grammar	Description

Name	Grammar	Description
cd	suffix	change the working directory
wd	constituent	print the working directory
rd	constituent	return the working directory to the root directory
ls	suffix	print all commands in the directory
lsr	suffix	print all commands and subcommands in the directory

Navigation commands can have constituent or suffix gramar, meaning that their usage is not identical. Navigation commands with constituent grammar are used like regular commands. They are written imideatly afther the **>** character and have no arguments or subcommands. The following example shows how to identify the working directory with **wd** and return to the root directory with **rd**.

```
 >wd
someDirectory
<

>rd
<

>wd
root
<
```

Navigation commands with suffix grammar are used in combination with a directory, which is written before the navigation command. They also do not have any arguments or subcommands. The following example demonstrates the usage the navigation command **cd**, which has suffix grammar.

```
 >baseCommand cd
>

>wd
baseCommand
<

>subcommand cd
<

>wd
subcommand
<

>rd
<

>baseCommand subcommand cd
<

>wd
subcommand
<
```

The navigation commands **ls** and **lsr** are used to print the subcommands of a directory. When printed, the arguments of each command are enclosed in braces **{}**. Directories are indicated with square brackets **[]**. If a directory has no default command, it will not have braces **{}**. The command **ls** only prints base commands of the prefixed directory. Any directories in the prefixed directory are displayed with **[...]** to signify that they have subcommands that are not shown. The command **lsr** expands the brackets and prints all subcommads recursively, with an increace in indentation for each level. Navigation commands are always acessable, but are not shown when using **ls** or **lsr**.

```
 >lsr
Command1 [
    Command1.1 {}
    Command1.2 {s}
]
Command2 {u} [
    Command2.1 {f}
    Command2.2 {} [
        Command2.2.1 {i}
    ]
]
<

>ls
Command1 [...]
Command2 {u} [...]
<

>Command1 ls
Command1.1 {}
Command1.2 {s}
<

>Command2 cd
<

>lsr
Command2.1 {f}
Command2.2 {} [
    Command2.2.1 {i}
]
<
```

Airbrakes Command Dictionary

Here is a dictionary containing a listing and description for each command in the airbrakes software as of August 2025. The dictionary's structure mirrors that of the actual command structure, with directories represented by collapsible menus. The search bar can be used to lookup commands, which will be hilighted if they match, or to search for keyword matches in commands or their descriptions.

arm

Places the system into the armed state. This command is used to start both real and simulated flights. After arming, a series of information, warning, and error messages will appear. These messages are also logged to the sd card if logging is enabled. Do not launch if there are any warnings or error messages, unless you know what you are doing.

Arguments: None

disarm

Disarms the system and ends any real or simulated flight. Use this command after recovering the system to ensure all data is saved correctly. This command will stop the controller, motor, telemetry, and event detection.

Arguments: None

state

Prints the current state of the system.

Table of States

State	Description
standby	Initial state of the system. The system stays in this state until it is armed. All subsystems are functional in this state and can be tested with commands.
armed	First state after the system is armed. The system waits for a launch to occur and records telemetry.

State	Description
boost	This state occurs breifly when the motor accelerates the system. The system waits for burnout to occur before deploying the airbrakes and optionally switches to high speed buffering mode for telemetry.
coast	The controller activates in the coast state along with the motor. The coast state occurs from burnout to apogee.
recovery	The system enters the recovery state after apogee is detected. The airbrakes are retracted and shut down, the controller is stoped, but telemetry continues to be loged directly onto the sd card until the system is disarmed.

Arguments: None

safe

Ensures that all systems are shutdown safe. Telemetry and log are flushed, persistent variables are saved, and the motor is shutdown. Use this before disconnecting power if you want to ensure that changes to persistent variables or files are not lost.

Arguments: None

restartSD

Reinitializes the SD card. This may resolve loading or recording issues with the SD card if the connection was bad at startup. **Do not** remove or reinsert the SD card while the system is powered on or files may get corrupted.

Arguments: None

flight

The flight directory provides an interface for data and parameters relating to flight planing and event detection. Before a flight, it is good practice to look through this directory to ensure that all parameters are in the correct configuration.

Subcommands

flight buffer

This directory is for the buffering flight option, which allows the system to use high speed buffering for telemetry durring the boost and coast phases of flight. The advantage of using the buffering option is that intermitent delays from SD card writes will not occur while the time sensitive signal processing and control algorithims are running. In buffering mode, a loss of power will cause all buffered data to be lost. If the buffer overflows, formatting errors may occur in the recovered files, but data will not be lost. This option can be enabled or disabled for a real flight. If using this option, test that enough memory is allocated to the buffer to avoid an overflow.

*This option is persistent, meaning that it can be saved across resets and program updates. Use the **safe** command or **persistent save** command after making changes to persistent variables to save.*

Default Command

Prints weather the option is set or cleared. If it is set then telemetry buffering will occur. Otherwise telemetry will be recorded directly to the SD card.

Arguments: None

Subcommands

flight buffer set

Enables telemetry buffering mode.

Arguments: None

flight buffer clear

Disables telemetry buffering mode.

Arguments: None

flight actuators

This directory is for the actuators flight option, which controls weather the actuators will be enabled durring flight. This option can be used to disable the actuators durring ground tests, or to configure a control flight where actuation is disabled but other flight systems are active. **Warning**: The airbrakes will not deploy durring flight if this option is not set. Make sure that it is set for a real flight. When arming the system, a warning message will appear if this option is not set. The actuators can be used normally through the **motor** directory even if disabled for flight. The actuators option only effects the flight logic responsible for deploying the actuators.

*This option is persistent, meaning that it can be saved across resets and program updates. Use the **safe** command or **persistent save** command after making changes to persistent variables to save.*

Default Command

Prints weather the option is set or cleared. If it is set then the actuators will be enabled durring flight. Otherwise, the actuators will be disabled.

Arguments: None

Subcommands

flight actuators set

Enables the actuators durring flight.

Arguments: None

flight actuators clear

Disables the actuators durring flight.

Arguments: None

flight sample

This directory is for the sample parameter, which controls the rate at which the event detection system samples the vehicles state. This parameter can be used to tune the event detection system. The sample parameter is the time in millisecods between checks made by the event detection system. A shorter sample period gives quicker response, but is more suceptable to noise or error.

*This parameter is persistent, meaning that it can be saved across resets and program updates. Use the **safe** command or **persistent save** command after making changes to persistent variables to save.*

Default Command

Prints the current sample period of the event detection system.

Arguments: None

Subcommands

flight sample set

Changes the sample period of the event detection system.

Arguments: *unisgned* - New value in milliseconds

flight launch

This directory is for the launch detection system. The vehicle's altitude, vertical velocity, and acceleration must all meet a threshold in consecutive samples for a launch to be detected. Additionally, minimum arm state time can also be applied. The threshold values, number of consectve samples, and state transition timing are parameters that are acessed through this directory.

Subcommands

flight launch properties

Prints all launch detection system parameters.

Arguments: None

flight launch altitude

This directory is for the altitude threshold parameter of the launch detection system. The vehicle's altitude must be **above** this threshold for a launch to be detected. Altitude is measured using the barometric altimeter. This value uses meters as the unit.

*This parameter is persistent, meaning that it can be saved across resets and program updates. Use the **safe** command or **persistent save** command after making changes to persistent variables to save.*

Default Command

Prints the current value of the launch detection system's altitude threshold.

Arguments: None

Subcommands

flight launch altitude set

Changes the value of the launch detection system's altitude threshold.

Arguments: *float* - New value in meters

flight launch velocity

This directory is for the vertical velocity threshold parameter of the launch detection system. The vehicle's vertical velocity must be **above** this threshold for a launch to be detected. Vertical velocity is measured by differentiating altitude readings from the altimeter. This value uses meters per second as the unit.

*This parameter is persistent, meaning that it can be saved across resets and program updates. Use the **safe** command or **persistent save** command after making changes to persistent variables to save.*

Default Command

Prints the current value of the launch detection system's vertical velocity threshold.

Arguments: None

Subcommands

flight launch velocity set

Changes the value of the launch detection system's vertical velocity threshold.

Arguments: *float* - New value in meters per second

flight launch acceleration

This directory is for the vertical acceleration threshold parameter of the launch detection system. The vehicle's vertical acceleration must be **above** this threshold for a launch to be detected. Vertical acceleration is measured using accelerometer readings from the IMU. This value uses meters per second squared as the unit.

*This parameter is persistent, meaning that it can be saved across resets and program updates. Use the **safe** command or **persistent save** command after making changes to persistent variables to save.*

Default Command

Prints the current value of the launch detection system's vertical acceleration threshold.

Arguments: None

Subcommands

flight launch acceleration set

Changes the value of the launch detection system's vertical acceleration threshold.

Arguments: *float* - New value in meters per second squared

flight launch samples

This directory is for the consecutive samples parameter of the launch detection system. All launch detection thresholds: altitude, velocity, acceleration, and time must be met for a number of consecutive samples for a launch event to be detected. This parameter dictates this number of consecutive samples.

*This parameter is persistent, meaning that it can be saved across resets and program updates. Use the **safe** command or **persistent save** command after making changes to persistent variables to save.*

Default Command

Prints the current value of the launch detection system's consecutive samples parameter.

Arguments: None

Subcommands

flight launch samples set

Changes the value of the launch detection system's consecutive samples parameter.

Arguments: *unsigned* - New value

flight launch time

This directory is for the time threshold parameter of the launch detection system. The time from when the airbrakes was armed to a possible launch event must be **above** this threshold for a launch to be detected. This value uses milliseconds as the unit.

*This parameter is persistent, meaning that it can be saved across resets and program updates. Use the **safe** command or **persistent save** command after making changes to persistent variables to save.*

Default Command

Prints the current value of the launch detection system's time threshold.

Arguments: None

Subcommandss

flight launch time set

Changes the value of the launch detection system's time threshold.

Arguments: *unsigned* - New value in milliseconds

flight burnout

This directory is for the burnout detection system. The vehicle's altitude, vertical velocity, and acceleration must all meet a threshold in consecutive samples for burnout to be detected. Additionally, a minimum boost time can also be applied. The threshold values, number of consectve samples, and state transition timing are parameters that are acessed through this directory.

Subcommands

flight burnout properties

Prints all burnout detection system parameters.

Arguments: None

flight burnout altitude

This directory is for the altitude threshold parameter of the burnout detection system. The vehicle's altitude must be **above** this threshold for burnout to be detected. Altitude is measured using the barometric altimeter. This value uses meters as the unit.

*This parameter is persistent, meaning that it can be saved across resets and program updates. Use the **safe** command or **persistent save** command after making changes to persistent variables to save.*

Default Command

Prints the current value of the burnout detection system's altitude threshold.

Arguments: None

Subcommands

flight burnout altitude set

Changes the value of the burnout detection system's altitude threshold.

Arguments: *float* - New value in meters

flight burnout velocity

This directory is for the vertical velocity threshold parameter of the burnout detection system. The vehicle's vertical velocity must be **above** this threshold for burnout to be detected. Vertical velocity is measured by differentiating altitude readings from the altimeter. This value uses meters per second as the unit.

*This parameter is persistent, meaning that it can be saved across resets and program updates. Use the **safe** command or **persistent save** command after making changes to persistent variables to save.*

Default Command

Prints the current value of the burnout detection system's vertical velocity threshold.

Arguments: None

Subcommands

flight burnout velocity set

Changes the value of the burnout detection system's vertical velocity threshold.

Arguments: *float* - New value in meters per second

flight burnout acceleration

This directory is for the vertical acceleration threshold parameter of the burnout detection system. The vehicle's vertical acceleration must be **below** this threshold for burnout to be detected. Vertical acceleration is measured using accelerometer readings from the IMU. This value uses meters per second squared as the unit.

*This parameter is persistent, meaning that it can be saved across resets and program updates. Use the **safe** command or **persistent save** command after making changes to persistent variables to save.*

Default Command

Prints the current value of the burnout detection system's vertical acceleration threshold.

Arguments: None

Subcommands

flight burnout acceleration set

Changes the value of the burnout detection system's vertical acceleration threshold.

Arguments: *float* - New value in meters per second squared

flight burnout samples

This directory is for the consecutive samples parameter of the burnout detection system. All burnout detection thresholds: altitude, velocity, acceleration, and time must be met for a number of consecutive samples for a burnout event to be detected. This parameter dictates this number of consecutive samples.

*This parameter is persistent, meaning that it can be saved across resets and program updates. Use the **safe** command or **persistent save** command after making changes to persistent variables to save.*

Default Command

Prints the current value of the burnout detection system's consecutive samples parameter.

Arguments: None

Subcommands

flight burnout samples set

Changes the value of the burnout detection system's consecutive samples parameter.

Arguments: *unsigned* - New value

flight burnout time

This directory is for the time threshold parameter of the burnout detection system. The time from when launch was detected to a possible burnout event must be **above** this threshold for burnout to be detected. This value uses milliseconds as the unit.

*This parameter is persistent, meaning that it can be saved across resets and program updates. Use the **safe** command or **persistent save** command after making changes to persistent variables to save.*

Default Command

Prints the current value of the burnout detection system's time threshold.

Arguments: None

Subcommandss

flight burnout time set

Changes the value of the burnout detection system's time threshold.

Arguments: *unsigned* - New value in milliseconds

flight apogee

This directory is for the apogee detection system. The vehicle's altitude, vertical velocity, and acceleration must all meet a threshold in consecutive samples for apogee to be detected. Additionally, a minimum coast time can also be applied. The threshold values, number of consectve samples, and state transition timing are parameters that are acessed through this directory.

Subcommands

flight apogee properties

Prints all apogee detection system parameters.

Arguments: None

flight apogee altitude

This directory is for the altitude threshold parameter of the apogee detection system. The vehicle's altitude must be **above** this threshold for apogee to be detected. Altitude is measured using the barometric altimeter. This value uses meters as the unit.

*This parameter is persistent, meaning that it can be saved across resets and program updates. Use the **safe** command or **persistent save** command after making changes to persistent variables to save.*

Default Command

Prints the current value of the apogee detection system's altitude threshold.

Arguments: None

Subcommands

flight apogee altitude set

Changes the value of the apogee detection system's altitude threshold.

Arguments: *float* - New value in meters

flight apogee velocity

This directory is for the vertical velocity threshold parameter of the apogee detection system. The vehicle's vertical velocity must be **below** this threshold for apogee to be detected. Vertical velocity is measured by differentiating altitude readings from the altimeter. This value uses meters per second as the unit.

*This parameter is persistent, meaning that it can be saved across resets and program updates. Use the **safe** command or **persistent save** command after making changes to persistent variables to save.*

Default Command

Prints the current value of the apogee detection system's vertical velocity threshold.

Arguments: None

Subcommands

flight apogee velocity set

Changes the value of the apogee detection system's vertical velocity threshold.

Arguments: *float* - New value in meters per second

flight apogee acceleration

This directory is for the vertical acceleration threshold parameter of the apogee detection system. The vehicle's vertical acceleration must be **below** this threshold for apogee to be detected. Vertical acceleration is measured using accelerometer readings from the IMU. This value uses meters per second squared as the unit.

*This parameter is persistent, meaning that it can be saved across resets and program updates. Use the **safe** command or **persistent save** command after making changes to persistent variables to save.*

Default Command

Prints the current value of the apogee detection system's vertical acceleration threshold.

Arguments: None

Subcommands

flight apogee acceleration set

Changes the value of the apogee detection system's vertical acceleration threshold.

Arguments: *float* - New value in meters per second squared

flight apogee samples

This directory is for the consecutive samples parameter of the apogee detection system. All apogee detection thresholds: altitude, velocity, acceleration, and time must be met for a number of consecutive samples for an apogee event to be detected. This parameter dictates this number of consecutive samples.

*This parameter is persistent, meaning that it can be saved across resets and program updates. Use the **safe** command or **persistent save** command after making changes to persistent variables to save.*

Default Command

Prints the current value of the apogee detection system's consecutive samples parameter.

Arguments: None

Subcommands

flight apogee samples set

Changes the value of the apogee detection system's consecutive samples parameter.

Arguments: *unsigned* - New value

flight apogee time

This directory is for the time threshold parameter of the apogee detection system. The time from when launch was detected to a possible apogee event must be **above** this threshold for apogee to be detected. This value uses milliseconds as the unit.

*This parameter is persistent, meaning that it can be saved across resets and program updates. Use the **safe** command or **persistent save** command after making changes to persistent variables to save.*

Default Command

Prints the current value of the apogee detection system's time threshold.

Arguments: None

Subcommandss

flight apogee time set

Changes the value of the apogee detection system's time threshold.

Arguments: *unsigned* - New value in milliseconds

flight plan

This directory is for managing the vehicle's flight plan. The flight plan is used to provide the system knolwedge about atmospheric conditions, launch vehicle properties, and airbrake parameters. The flight plan also contains a mesh of target altitude values which the controller uses as reference durring flight. The flight plan is generated on an external machine and loaded from the SD card. The flight plan is automatically loaded from a saved file location at startup, but can also be changed or reloaded here.

Subcommands

flight plan properties

Displays all of the metadata associated with the currently loaded flight plan. If no flight plan is loaded only size of the allocated storage for flight plans is listed.

```
>flight plan properties
No flight plan is loaded
64 kB available for storage
<
```

If a flight plan fails to load because it is too large, this command can be used to see what the maximum storage space is. The maximum storage space can be increaced by changing a configuration value in the source code. If a flight plan is loaded, all of it's metadata is listed. Some metadata parameters, like the effective drag area range and mass, are used directly by the controller. Others are keped for reference, allowing you to check which values were used to produce the flight plan.

```
>flight plan properties
Flight plan 'flightPath.csv':
Target apogee: 800.00m
Deployment range: 0 degrees - 90.00 degrees
Effective drag area range: 0.0025m^2 - 0.0100m^2
Dry mass: 3.00kg
Launch site conditions: 15.00C at 101325.00pa
Vertical velocity range: 0m/s - 150.00m/s with 64 samples
Angle with horizontal range: 0 degrees - 90.00 degrees with 64 samples
Using 16 kB of available 64 kB storage
<
```

Description of Metadata

Value	Description
Target Apogee	The target apogee of the flight plan. To change the target apogee, a new flight plan needs to be generated and loaded with the SD card.
Deployment Range	The physical angle the airbrakes can be opend to. It is important that the actuators are configured so that their maximum deployment angle is the same as the one in the flight plan.
Effective Drag Area Range	The lower bound of the range is the drag area of the launch vehicle when the airbrakes are fully retracted. The upper bound is the drag area when the airbrakes are extended to the maximum angle, which is the upper bound of the deployment range.
Dry Mass	The mass of the launch vehicle and airbrakes combined.
Launch Conditions	The temperature and pressure that were used to generate the flight plan. Small deviations in actual conditons occuring durring the time between flight plan generation and launch are acceptable.
Velocity & Angle Range	The range of vehicle states for which a target altitude is available. This range should encompass the entire range of velocities and angles the vehicle is expected to fly at.
Size	The size of the flight plan when loaded into the flight computer's RAM. This is not the same as the file size. This is dependent upon the number of samples in both the velocity and angle dimensions.

It is good practice to check that the flight plan metadata is correct before a launch using this command.

Arguments: None

flight plan load

Tries to load the flight plan at the specified path. If the new flight plan fails to load, the previously loaded flight plan will **not** still be loaded and needs to be reloaded manually.

*The filename of the flight plan is persistent, meaning that it can be saved across resets and program updates. Use the **safe** command or **persistent save** command after making changes to persistent variables to save.*

Arguments: *string* - Name of the file containing the new flight plan

flight plan altitude

Prints the flight plan's target altitude for a given velocity and angle pair. This can be used to inspect the flight plan.

Arguments: *float* - vertical velocity of the vehicle (m/s), *float* - angle to the horizontal of the vehicle (degrees)

flight plan gradient

Prints the gradient of the flight plan at a given velocity and angle pair. The control algorithim uses the gradient of the flight plan, so this is available to check what

values it may be seeing.

Arguments: *float* - vertical velocity of the vehicle (m/s), *float* - angle to the horizontal of the vehicle (degrees)

controller

This directory is for the airbrakes controller. The controller commands the position of the airbrakes based on the predicted state of the vehicle. The controller relies on the flight plan, which continuously gives it a reference target altitude that it tries to approach and follow. The flight plan is designed so that the reference altitude leads to the target apogee. The controller's primary objective to drive the error between the flight plan and the current predicted altitude to zero. The controller reads the vehicle's state from the observer, a system which processes incoming sensor data in real time. The observer can be disabled using the **sim** directory to allow simulation of the controller. While active, the controller will automatically move the actuators if they are activated. This directory allows the parameters of the controller to be tuned, and for it to be manually started or stoped.

Subcommands

controller start

This command allows you to manually start the controller. The flight logic will automatically start the controller when burnout is detected, so this command is not nessesary to operate the system. It can however be usefull for testing.

Arguments: None

controller stop

This command allows you to manually stop the controller. The flight logic will automatically stop the controller when apogee is detected, so this command is not nessesary to operate the system. It can however be usefull for testing.

Arguments: None

controller period

This directory is for the period parameter of the controller. The period controls the rate at which the controller recalculates what airbrake deployment is commanded. Small update periods allow fine control, but if too small the controller can overwhelm the actuators and create chopy movements.

*This parameter is persistent, meaning that it can be saved across resets and program updates. Use the **safe** command or **persistent save** command after making changes to persistent variables to save.*

Default Command

Prints the current period at which the controller updates in microseconds.

Arguments: None

Subcommands

controller period set

Changes the period at which the controller updates.

Arguments: *unsigned* - New value in microseconds

controller decay

This directory is for the decay parameter of the controller. The rule that the controller uses to decide the airbrake deployment is based on the idea that the error between the current and reference altitude should decay exponentially. The rate of exponential decay is configurable using this parameter. The steady state error of the controller when it's drag model is incorrect is inversly porportonal to the dacay rate, so increacing the decay rate will improve accuracy in this regard. When latency occurs, either due to the actuators or sensor processing, large decay rates induce oscilations in steady state error akin to an underdamped harmonic oscilator. It is easiest to choose an appropriate decay rate that has good performance with minimal oscilation through simulation. This directory is used to manipulate the decay rate value. **Important:** The sign of the decay rate for an exponential function must be **negative**, otherwise exponential growth will occur. You will get a warning if you try to put a positive value.

*This parameter is persistent, meaning that it can be saved across resets and program updates. Use the **safe** command or **persistent save** command after making changes to persistent variables to save.*

Default Command

Prints the current decay rate of the controller.

Arguments: None

Subcommands

controller decay set

Changes the decay rate of the controller. The sign of the decay rate for an exponential function must be **negative**, otherwise exponential growth will occur. You will get a warning if you try to put a positive value.

Arguments: *float* - New value

controller coast

This directory is for the coast parameter of the controller. It may be desirable to stop the controller from updating just before apogee. The solution to the exponential decay problem which is used by the controller has a singularity when vertical velocity is zero. This setting will cause the controller sustain the last computed deployment once the vehicle's vertical velocity falls below a threshold. The coast velocity has units of meters per second.

*This parameter is persistent, meaning that it can be saved across resets and program updates. Use the **safe** command or **persistent save** command after making changes to persistent variables to save.*

Default Command

Prints the current coast velocity.

Arguments: None

Subcommands

controller coast set

Changes the coast velocity.

Arguments: *float* - New value in meters per second

log

This directory is for the log. The log stores messages onto the SD card durring flight. Log messages are recorded by the user or automatically through software. The flight logic automatically records all event detections and any errors it encounters. The log also records timestamps, making it possible to line up with telemetry for post flight analysis. When the airbrakes is armed, a new log is automatically created. This directory allows the user to configure and use the log.

Default Command

Logs a message. The timestamp of the message is automatically added. If no log file exists, a new file is automatically created.

Arguments: *string* - Message to be loged

Subcommands

log new

Creates an empty log file. If the log file already exists it is overwritten. A new log will autoamtically be created when the system is armed.

Arguments: None

log name

This directory is for the name parameter of the log. This parameter controls the name of the SD card file where logs are saved.

*This parameter is persistent, meaning that it can be saved across resets and program updates. Use the **safe** command or **persistent save** command after making changes to persistent variables to save.*

Default Command

Prints the name of the current log file

Arguments: None

Subcommands

log name set

Changes the name of the file used for logging. The name of the file which contains previous logs is not changed by this command. New logs are just stored in a different file.

Arguments: None

log mode

This directory is for the log's mode option. The log is used in either recording mode or buffering mode. Recording mode immediately saves messages to the SD card. Buffering mode buffers messages in RAM before recording all of them to the SD card at once. Buffering mode eliminates the slowdowns endured by frequent SD card writes. In buffering mode, if the buffer is not flushed before powering down, the data will be lost. The flight logic auomatically switches between modes durring the appropriate stages of flight. The directory **flight buffer** is used to manage this. This directory should only be used for testing and **not** to change modes for a flight.

*This option is persistent, meaning that it can be saved across resets and program updates. Use the **safe** command or **persistent save** command after making changes to persistent variables to save.*

Default Command

Prints the mode that the log is currently in

Arguments: None

Subcommands

log mode buffer

Places the log into buffering mode.

Arguments: None

log mode record

Places the log into recording mode.

Arguments: None

log override

This directory is for the log's override option, which allows all logs to be blocked from being recorded. Override mode is used to extend the life of the SD card by not recording logs to it durring tests. This option should **not** be set for a real flight. No logs will be recorded with this option enabled. If this option is enabled when the system is armed, a warning will be printed.

*This option is persistent, meaning that it can be saved across resets and program updates. Use the **safe** command or **persistent save** command after making changes to persistent variables to save.*

Default Command

Prints weather the override option is set or cleared. If it is set, logs are blocked. Otherwise, they are recorded normally.

Arguments: None

Subcommands

log override set

Enables the log's override option. Logs are not recorded when the option is set.

Arguments: None

log override clear

Disables the log's override option. Logs are recorded normally when the option is cleared.

Arguments: None

telemetry

This directory is for the telemetry system. The telemetry system stores flight data in .csv format onto the SD card durring flight. All telemetry values are recorded periodically from the time the airbrakes is armed untill they are disarmed.

Table of Telemetry Values

Value	Description
Flight State	State of flight the program is in. *See the **state** command.*
Observer Output	Predicted altitude, vertical velocity, vertical acceleration, and angle to horizontal.
Sensor Readings	Raw altimeter pressure, temperature & altitude readings. Raw IMU acceleration, orientation, rotation, & gravity readings. *See the **altimeter** and **imu** directories.*

Value	Description
Flight Path	Reference altitude, error, and mesh derivatives. *See the **controller** and **flight plan** directories.*
Actuator Data	Requested drag area and actual drag area computed from encoder position.*See the **motor** directory.*
Controller Data	Intermideate computation values and event flags. *See the **controller** directory.*

When the airbrakes is armed, a new telemetry file is automatically created. This directory allows the user to configure and use the telemetry system.

Subcommands

telemetry new

Creates a fresh telemetry file. If the telemetry file already exists it is overwritten. A new telemetry file will autoamtically be created when the system is armed.

Arguments: None

telemetry name

This directory is for the name parameter of the telemetry system. This parameter controls the name of the SD card file where telemetry is saved.

*This parameter is persistent, meaning that it can be saved across resets and program updates. Use the **safe** command or **persistent save** command after making changes to persistent variables to save.*

Default Command

Prints the name of the current telemetry file

Arguments: None

Subcommands

telemetry name set

Changes the name of the file used for telemetry. The name of the file which contains previous telemetry is not changed by this command. New telemetry is just stored in a different file.

Arguments: None

telemetry mode

This directory is for the telemetry system's mode option. The telemetry system is used in either recording mode or buffering mode. Recording mode immediately saves telemetry to the SD card. Buffering mode buffers telemetry in RAM before recording all of it to the SD card at once. Buffering mode eliminates the slowdowns endured by frequent SD card writes. In buffering mode, if the buffer is not flushed before powering down, the data will be lost. The flight logic auomatically switches between modes durring the appropriate stages of flight. The directory **flight buffer** is used to manage this. This directory should only be used for testing and **not** to change modes for a flight.

*This option is persistent, meaning that it can be saved across resets and program updates. Use the **safe** command or **persistent save** command after making changes to persistent variables to save.*

Default Command

Prints the mode that the telemetry system is currently in

Arguments: None

Subcommands

telemetry mode buffer

Places the telemetry system into buffering mode.

Arguments: None

telemetry mode record

Places the telemetry system into recording mode.

Arguments: None

telemetry refresh

This directory is for the refresh parameter of the telemetry system. The refresh parameter dictates the time between each telemetry update. Logging telemetry quickly is valuble to save a more detailed version of the flight. Recording telemetry too quickly can result in huge telemetry files. In recording mode, quicker recording can slow the program and in buffering mode it can overflow the buffer more quickly.

Default Command

Prints the refresh time for the telemetry system.

Arguments: None

Subcommands

telemetry refresh set

Changes the refresh time for the telemetry system.

Arguments: *unisgned* - New value in milliseconds

telemetry override

This directory is for the telemetry system's override option, which allows telemetry to be blocked from being recorded. Override mode is used to extend the life of the SD card by not recording telemetry to it durring tests. This option should **not** be set for a real flight. No telemetry will be recorded with this option enabled. If this option is enabled when the system is armed, a warning will be printed and loged.

*This option is persistent, meaning that it can be saved across resets and program updates. Use the **safe** command or **persistent save** command after making changes to persistent variables to save.*

Default Command

Prints weather the override option is set or cleared. If it is set, telemetry is blocked. Otherwise, it is recorded normally.

Arguments: None

Subcommands

telemetry override set

Enables the telemetry system's override option. Telemetry is not recorded when the option is set.

Arguments: None

telemetry override clear

Disables the telemetry system's override option. Telemetry is recorded normally when the option is cleared.

Arguments: None

persistent

The persistent storage system allows program variables to persist across resets and power cycles. The persistent storage system automatically restores all system states and settings upon restart. When the program is modified, persistent variables will persist so long as none of them are added, reordered or removed. If this occurs, system defaults are restored and a startup message is printed.

Subcommands

persistent save

Saves any modified persistent variables. This or the **safe** command must be used before powering down to save changes to persistent variables.

Arguments: None

persistent restore

This directory is for restoring persistent variables. Persistent variables are automatically restored at startup but can also manually be restored using this directory.

Default Command

Restores all persistent variables to their previously saved values.

Arguments: None

Subcommands

persistent restore defaults

Restores all persistent variables to their default values.

Arguments: None

sim

The sim direcotry contains the interface for operating the airbrakes in simulation mode. Simulation mode allows for the full airbrakes system to be tested in a simulated flight. Simulation mode requires an external machine to provide the simulated flight envirnoment. The current airbrakes simulation system uses a Simulink model to mimic a real flight and a python script to facilitate communication between the Simulink model, airbrakes flight computer, and the user. Simulation mode causes the observer, which is responsible for producing usable vehicle state estimations from sensor readings, to be put into simulation mode. In simulation mode, the observer ignores sensor readings and allows the simulation to override it's estimated values. The simulation system also sends simulation data to the host device while also listening for simulation data sent from the host device. If using a standard serial terminal, simulation packets, which have a characteristic **#** symbol prefixing them, will flood the terminal durring simulation. This is because the same serial port is used for the command line interface and the simulation.

```
 >sim start
<


#4.607010 -2.100000 0.100000
#4.683203 -1.050670 0.200000
```

The airbrakes simulation bridge program filters these transmissions out, allowing the command line interface to be used durring simulation. A seperate document futher details how the simulation system works and how to use it.

Default Command

Prints weather or not the simulation is running.

Arguments: None

Subcommands

sim start

Enters simulation mode.

Arguments: None

sim stop

Exits simulation mode.

Arguments: None

sim refresh

This directory is for the simulation system's refresh parameter. Simulation data is sent and decoded at the rate dictated by this value. Faster simulations are more accurate, but can strain the external simulation system. Make sure that the simulation refresh period matches on both the aribrakes and external simulation.

*This parameter is persistent, meaning that it can be saved across resets and program updates. Use the **safe** command or **persistent save** command after making changes to persistent variables to save.*

Default Command

Prints the refresh time for the simulation system.

Arguments: None

Subcommands

sim refresh set

Changes the refresh time for the simulation system.

Arguments: *unsigned* - New value in milliseconds

altimeter

This directory enables interaction with the airbrakes' MS5607 barometric altimeter. Based on the ISA model, the barometric altimeter's pressure and temperature readings determine the vehicle's altitude. The altimeter will give different readings depending on atmospheric conditions. The ISA model allows for a nominal pressure and temperature value to be used to measure altitude relative to the position where the nominal readings were made. This process, refered to as zeroing the altimeter, is managed from this directory.

Subcommands

altimeter pressure

Measures and prints the current ambient pressure in pascals.

Arguments: None

altimeter temperature

Measures and prints the ambient temperature in Kelvin.

Arguments: None

altimeter altitude

Measures and prints the current altitude in meters.

Arguments: None

altimeter zero

Zeros the altimeter.

*Zeroing data is persistent, meaning that it can be saved across resets and program updates. Use the **safe** command or **persistent save** command after making changes to persistent variables to save.*

Arguments: None

altimeter init

This directory is for initializing the altimeter. The altimeter is initialized automatically at startup, but can be re-initialized in this directory.

Default Command

Re-initializes the altimeter.

Arguments: None

Subcommands

altimeter init get

Prints weather the altimeter is initialized.

Arguments: None

altimeter speed

This directory is for the SPI speed parameter for the altimeter. SPI is used to communicate with the altimeter. The SPI speed parameter controls how quickly SPI communncation runs.

Default Command

Prints the current SPI speed.

Arguments: None

Subcommands

altimeter speed set

Changes the SPI speed.

Arguments: *unsigned* - New value in Hz

imu

This directory enables interaction with the airbrakes' BNO085 inertial measurement unit, or IMU. The IMU provides acceleration, orientation and rotation data. The BNO085 does alot more than simply read data from its accelerometers, gyroscopes, and magnetometer. Its runs sensor fusion algorithims and outputs data vectors adjusted to the correct frame of reference. Because the BNO085 handles much of the data processing on chip, it outputs periodic sensor reports containing computed values rather than raw sensor readings. The airbrakes system makes use of four of the BNO085's sensor reports. There are many others available, which may be usefull for future itterations or projects.

Table of Supported Sensor Reports

Report Type	Description
Linear Acceleration Vector	State of flight the program is in. *See the **state** command.*
Rotation Vector	Predicted altitude, vertical velocity, vertical acceleration, and angle to horizontal.
Gravity Vector	Raw altimeter pressure, temperature & altitude readings. Raw IMU acceleration, orientation, rotation, & gravity readings. *See the **altimeter** and **imu** directories.*
Orientation Quaternion	Reference altitude, error, and mesh derivatives. *See the **controller** and **flight plan** directories.*

Subcommands

imu status

des

Arguments: None

imu tare

des

Arguments: None

imu reset

des

Arguments: None

imu acceleration

des

Default Command

Arguments: None

Subcommands

imu acceleration period

des

Arguments: None

imu acceleration start

des

Arguments: None

imu acceleration stop

des

Arguments: None

imu orientation

des

Default Command

Arguments: None

Subcommands

imu orientation period

des

Arguments: None

imu orientation start

des

Arguments: None

imu orientation stop

des

Arguments: None

imu rotation

des

Default Command

Arguments: None

Subcommands

imu rotation period

des

Arguments: None

imu rotation start

des

Arguments: None

imu rotation stop

des

Arguments: None

imu gravity

des

Default Command

Arguments: None

Subcommands

imu gravity period

des

Arguments: None

imu gravity start

des

Arguments: None

imu gravity stop

des

Arguments: None

imu periods

des

Default Command

Arguments: None

Subcommands

imu periods set

des

Arguments: None

imu speed

des

Default Command

Arguments: None

Subcommands

imu speed set

des

Arguments: None

motor

des

Default Command

Arguments: None

Subcommands

motor position

des

Arguments: None

motor start

des

Arguments: None

motor stop

des

Arguments: None

motor zero

des

Arguments: None

motor tare

des

Arguments: None

motor speed

des

Arguments: None

motor target

des

Default Command

Arguments: None

Subcommands

motor target set

des

Arguments: None

motor mode

des

Default Command

Arguments: None

Subcommands

motor mode full

des

Arguments: None

motor mode half

des

Arguments: None

motor mode quarter

des

Arguments: None

motor mode micro

des

Arguments: None