## Vehicle Abort Manager Subsystem

The Vehicle Abort Manager Subsystem (VAMS) is a software component that is responsible for aborting a mission in the event of an emergency and acts as a watchdog responding to different failure modes. VAMS receives data from a variety of sensors, including the vehicle's GPS, inertial measurement unit and mission path data from the central guidance system. VAMS uses this data to determine if the vehicle is in danger and, if so, initiates an abort sequence. The abort sequence may include resetting the system state to avoid any further issues.

VAMS is a critical component of the vehicle's safety system. It is designed to detect and respond to a variety of potential hazards, including:

- Loss of control / bad sensor data
- Unexpected low battery power
- Vehicle too far off intended course

The VAMS algorithm is intended to check all these conditions and return 1 or 0 to the core state machine which either continues in the ARMED state or rests to the STANDBY state depending on the output of VAMS.

---

## Loss of control / bad sensor data

The VAMS algorithm should firstly perform verification of the pitch, roll and yaw angles returned by the vehicle. Verification is making sure the data is within the acceptable bounds of the inherent value.

**Pitch** has an acceptable range from -90 degrees to 90 degrees. Anything outside this range should fail verification.

**Yaw** has an acceptable range from -180 degrees to 180 degrees. Anything outside this range should fail verification.

**Roll** has an acceptable range from -180 degrees to 180 degrees. Anything outside this range should fail verification.

As the vehicle inertial measurement data is crucial for powered flight, there will be no decision matrix used to determine if failure is allowable. An immediate FAIL will be returned for this VAMS step should the sensor data not meet verification requirements.

---

## Unexpected low battery power

The VAMS algorithm will also have to account for unexpected low battery power. Before every flight the core submodule checks to make sure the vehicle has enough power to fully perform its mission configured by an operator. This initial submodule should allow or disallow flight based on battery information. However, under special circumstances the vehicle may need to respond to issues in flight. Issues include:

- The initial battery checker submodule prediction was wrong and flight was wrongly allowed
- Although the battery appeared to be charged and operational at the initial check, the battery was bad and experienced a massive drop during flight requiring immediate attention

- Unexpected conditions, such as extreme wind, rain etc caused the vehicle guidance to overwork the throttle, and motors resulting in excessive battery drain which could not be caught by the initial checker

The VAMS algorithm must contain a 5% battery buffer for any given mission profile. This means that it must make a decision on whether to fail this step based on if it will have greater than 5% of battery power remaining at the target. The algorithm must determine the duration of flight remaining and using the battery current, voltage and current percent which it can retrieve through PTAM registers, it must predict how long until the battery hits the buffer threshold. If the available flight time remaining is less than duration of flight to target remaining, the algorithm should FAIL this VAMS step.

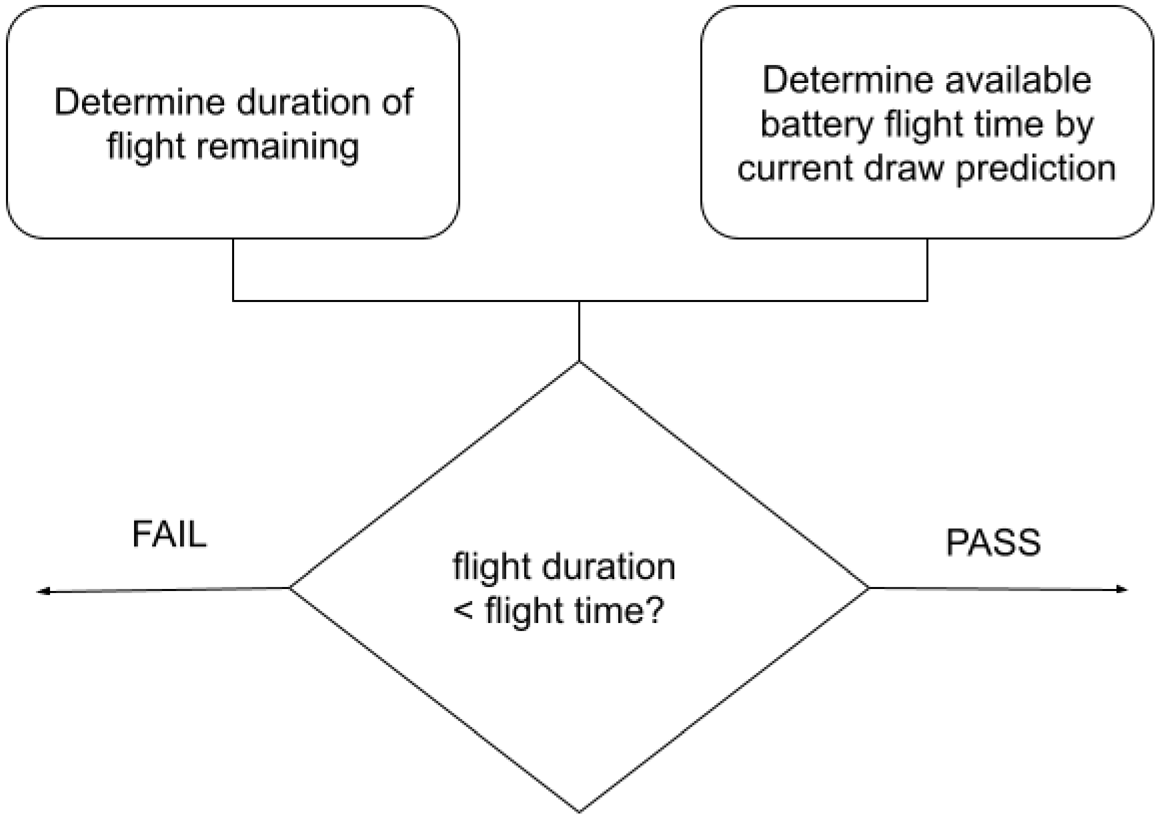
Determine duration of flight remaining
Determine available battery flight time by current draw prediction
FAIL
flight duration < flight time?
PASS

**Fig 1. Battery algorithm steps**

## Vehicle too far off intended course

The VAMS algorithm will also have to account for the vehicle moving too far off its intended course. As the vehicle is in flight, certain extreme circumstances may cause the central vehicle guidance systems to not effectively compensate in response. The VAMS algorithm acts as a watchdog to monitor and respond to this case. Using path data pushed to VAMPS, latitude, longitude and altitude are compared with current sensor data and if the current vehicle position exceeds the distance of the area of acceptability the VAMPS step is immediately failed.

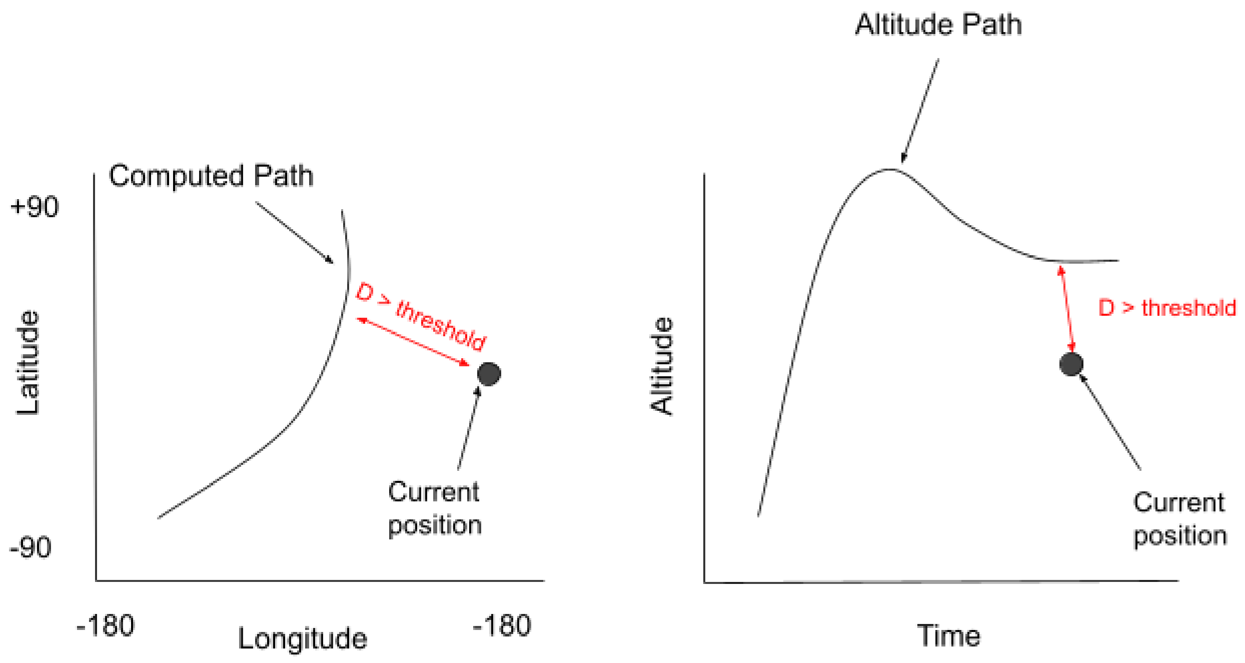

**Fig 2. Off-course scenarios**

## VAMPS Decision Step

After the algorithm reviews the possible failure modes / VAMPS steps, it reaches a final decision step to which it returns a 0 or a 1 to the firmware core state machine. Each VAMPS step has a weight ($w$) assigned to it to describe its operational priority. Each current step is crucial to flight and is weighted equally as ($w$ = 2) however with the addition of further failure modes, there will be additional varying weights.

- Loss of control / bad sensor data (LOC) ($w$ = 2)
- Unexpected low battery power (ULB) ($w$ = 2)
- Vehicle too far off intended course (FOC) ($w$ = 2)

The failure mode / VAMPS step is multiplied by its weighted value and the total is stored. If the sum of all totals is greater than our assigned threshold value, the algorithm returns a 1 representing an abort to the firmware core state machine.

<u>Example decision matrix scenario with a battery failure:</u>

FAIL = 0

PASS = 1

| VAMPS Step | Status | Weighted Status | Weighted Total |
|---|---|---|---|
| LOC/BSD | 1 | 1 x ($w$ = 2) | 2 |
| ULB | 0 | 0 x ($w$ = 2) | 0 |
| FOC | 1 | 1 x ($w$ = 2) | 2 |

**Fig 3. Final VAMPS decision matrix**

$Threshold = 2$

Therefore:

$xtotal = (LOC/BSD) + (ULB) + (FOC)$

$xtotal = (2) + (0) + (2) = 4$

Our total value is greater than the assigned threshold so an abort is called.