UtahStateUniversity
Electrical & Computer
Engineering

UAV Flight Controller Specifications Document

Tanner Chase
Christian Hall
Anthony Wood

November 2, 2025

Signature Page

Signature	*Signature*	*Signature*
Date and email	*Date and email*	*Date and email*

Revision History

Revision	Description	Author	Date	Approval
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				

Contents

1 SCOPE 6

2 APPLICABLE DOCUMENTS 7

3 STAKEHOLDER REQUIREMENTS 8

4 ENGINEERING REQUIREMENTS 9

4.1 System Requirements 10
4.1.1 Interfaces 10
4.1.2 Processing 11
4.1.3 Built in HW Capabilities 11
4.1.4 HW Interfaces 12
4.1.5 HAL 12
4.1.6 Power Management 13
4.1.7 Sub-Parts 13
4.2 Planner 14
4.2.1 Interfaces 14
4.2.2 General Requirements 14
4.2.3 Waypoint Planner 15
4.2.4 RC Mixer 15
4.2.5 State Select 16
4.3 Controller 17
4.3.1 Interfaces 17
4.3.2 Preprocessor 17
4.3.3 PVA Controller 18
4.3.4 ATL Controller 18
4.3.5 Control Distributor 18
4.4 Navigation 20
4.4.1 Interfaces 20
4.4.2 Sensor Selector 20
4.4.3 Estimator 21
4.5 Logger 22
4.5.1 Interfaces 22
4.5.2 General Requirements 23
4.5.3 Storage 23
4.5.4 Transmitt 23

5 VERIFICATION OF REQUIREMENTS 24
5.1 System Verification 25
5.1.1 Processing 25
5.1.2 Built in HW Capabilities 25
5.1.3 HW Interfaces 25
5.1.4 HAL 25
5.1.5 Power Management 25
5.2 Planner Verification 26
5.2.1 Waypoint Planner 26
5.2.2 RC Mixer 26
5.2.3 State Select 26
5.3 Controller Verification 27
5.3.1 Preprocessor 27
5.3.2 PVA Controller 27
5.3.3 ATL Controller 27
5.3.4 Control Distributor 28
5.4 Navigation Verification 29
5.4.1 Sensor Selector 29
5.4.2 Estimator 29
5.5 Logger Verification 30
5.5.1 Storage 30
5.5.2 Transmitt 30
5.6 Verification Methods 31
5.7 Verify Coverage of Stakeholder Requirements 32

Specifications

1 SCOPE

(a) **General:**

(b) **Acronyms:**

2 APPLICABLE DOCUMENTS

The following documents shown shall form part of the specifications for this project. In the event of a conflict between requirements, priority shall first go to the contract, second to this document, and lastly to these reference documents.

(a) **Government Documents** *This is where to put MIL-Specs, MIL-STDs, NASA specs and so forth. Be sure to include the revision level and date.*

(b) **Industry Documents** *This is where to put ANSI, ASTM, ASME, IEEE, Company specifications and so forth. Both this section and government documents can be divided up into logical subcategories.*

3 STAKEHOLDER REQUIREMENTS

4 ENGINEERING REQUIREMENTS

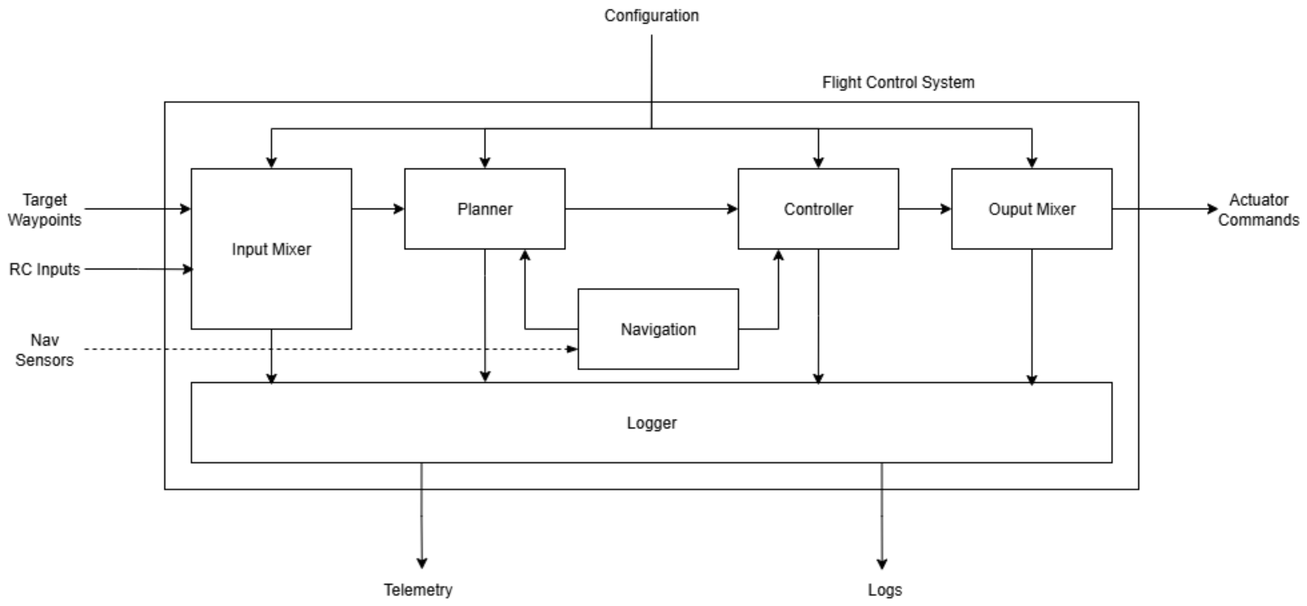

Figure 1. System functional diagram

4.1 System Requirements

The System provides the hardware and software foundation for all onboard modules. It includes processing, hardware capabilities, interfaces, hardware abstraction, and power management. The System is comprised of five main parts: Processing, Built in HW Capabilities, HW Interfaces, HAL, and Power Management.

4.1.1 Interfaces

4.1.1.1 Inputs

4.1.1.1.1 All submodules require configuration parameters that dictate their operation. These parameters will all be contained in a YAML file required for operation start. These configuration parameters will be different dependant on the individual submodule, and that submodule will define the contents of its own configuration. In software, the parameters will be passed as a predefined object, referred to as a struct, though there is no requirement to utilize the C programming language and it's associated structs.

4.1.1.1.2 Waypoints are defined as a time index series of LLA points. Waypoints do not require an orientation or velocity component to avoid overconstraining the Planner submodule.

4.1.1.1.3 RC inputs are defined as the literal signals received from the RC receiver. This includes the raw stick inputs, as well as any auxiliary channels. The layout of these channels will be defined in the configuration parameters for a specific transmitter.

4.1.1.1.4 External sensor data is defined as raw data received from any sensors not directly connected to the onboard PCB. This includes sensors connected via serial communication.

4.1.1.2 Outputs

4.1.1.2.1 Peripheral commands are defined as commands to control onboard motors and servos. These commands are generated by the Controller submodule.

4.1.1.3 Internal

4.1.1.3.1 UAV state estimates are defined as the best estimate of the UAV's current position, orientation, velocity, and angular velocity. This estimate is generated by the Navigation submodule.

4.1.1.3.2 Desired states are defined as a time index series of the positions, orientations, velocities, and angular velocities. These desired states are generated by the Planner submodule.

4.1.1.3.3 State mask is defined as a bitmask indicating which states are to be controlled by the Controller submodule, and is generated by the Planner submodule.

4.1.1.3.4 RC adjustments are defined as trims and scaling factors for each control input. Based on the transmitter mapping, the adjusments will be applied to the raw RC inputs and passed to the Logger for logging. These adjustments are generated by the Planner submodule.

4.1.1.3.5 Raw control outputs are defined as the initial values calculated by the Controller for percent thrust, aileron deflection, elevator deflection, and rudder deflection. These values aren't directly used, but are passed to the Logger for logging purposes.

4.1.1.3.6 Error reports are defined as any fault or error detected by the HAL or Planner, Navigation, or Controller submodules during operation. These reports are passed to the Logger for logging and potential transmission. Error reports shall include a timestamp, error code, and description of the error.

4.1.1.3.7 Onboard sensor data is defined as raw data received from any sensors directly connected to the onboard PCB.

4.1.2 Processing

The Processing part provides the computational resources and pipeline for all onboard software modules.

4.1.2.1 Provide sufficient compute for all modules.

4.1.2.2 Support real-time scheduling and isolation.

4.1.2.3 Ensure fault tolerance and recovery.

4.1.3 Built in HW Capabilities

Built-in hardware capabilities include sensors, actuators, and onboard peripherals required for mission execution.

4.1.3.1 Provide required sensors and actuators.

4.1.3.2 Ensure hardware health monitoring.

4.1.3.3 Support hardware expansion where feasible.

4.1.4 HW Interfaces

Hardware interfaces provide connectivity between onboard modules and external devices.

4.1.4.1 Support standard communication protocols.

4.1.4.2 Ensure robust electrical and logical connections.

4.1.4.3 Provide diagnostics for interface health.

4.1.5 HAL

The Hardware Abstraction Layer (HAL) provides a uniform interface to hardware resources, abstracting device specifics from higher-level modules. All configuration of the system will be done through the HAL. All necessary configuration information will be included in the YAML file which will be read in at program initialization.

4.1.5.1 Requirements

4.1.5.1.1 The HAL shall provide a scope and deliverables document detailing the capabilities and resources offered.

4.1.5.1.2 The HAL scope and deliverables document shall include support to at least the following functionalities:

- GPIO pin control
- I2C/SPI bus management
- UART serial communication
- USB 2.0 serial communication
- CAN communication
- Micro SD card read and write support
- ADC support
- Timer control
- Interrupt control
- Generic device interfacing (Requires associated driver)

4.1.5.1.3 The HAL shall provide an ICD outlining the APIs utilized for all functionality described in the scope and deliverables document.

4.1.5.1.4 The HAL ICD shall describe the ownership of all memory for all functions receiving or returning memory pointers.

4.1.5.1.5 The HAL shall provide common status codes which are returned from all HAL functions to avoid silent failures.

4.1.5.1.6 The HAL shall provide a method of gracefully handling faults and reporting those faults.

4.1.5.1.7 The HAL shall adhere to all coding standards described in §X.X.X

4.1.6 Power Management

Power Management ensures efficient and reliable power distribution to all system components.

4.1.6.1 Monitor and regulate power supply.

4.1.6.2 Support low-power and standby modes.

4.1.6.3 Protect against overcurrent and undervoltage.

4.1.7 Sub-Parts

The System is comprised of four functional sub-parts:

4.1.7.1 Planner

4.1.7.2 Controller

4.1.7.3 Navigation

4.1.7.4 Logger

4.2 Planner

The purpose of the Planner is to generate a series of realizable states for the UAV. The planner allows for both autonomous and pilot-directed operation. Inputs from both autonomous and pilot-directed modes are evaluated and potentially adjusted to fit safety and feasibility constraints. The inputs are evaluated based on the current UAV state and the current tolerances from configuration parameters. The Planner is comprised of three submodules: Waypoint Planner, RC Mixer, and State Select. Each has aids in the overall functionality of the Planner.

4.2.1 Interfaces

4.2.1.1 Inputs

4.2.1.1.1 The planner requires the mission parameters and UAS tolerances when generating waypoints. These tolerances shall include maximum and minimum airspeeds, maximum g-load, maximum climb and descent rates, maximum altitude, maximum bank angle, and minimum turn radius. Additionally, the planner's configuration parameters shall include an operating mode to determine whether RC inputs or generated waypoints are used to generate the desired states.

4.2.1.1.2 Waypoints as defined in 4.1.1.1.X.

4.2.1.1.3 RC inputs as defined in 4.1.1.1.Y.

4.2.1.1.4 UAV state estimates as defined in 4.1.1.3.Z.

4.2.1.2 Outputs

4.2.1.2.1 Desired States as defined in 4.1.1.3.X.

4.2.1.2.2 State mask as defined in 4.1.1.3.Y.

4.2.1.2.3 Logging data as defined in 4.1.1.3.Z.

4.2.1.3 Internal

4.2.1.3.1 Planned waypoint states. These inputs are generated by the Waypoint Planner submodule and consumed by the State Select submodule where they'll be multiplexed based on operating mode.

4.2.2 General Requirements

4.2.2.1 The Planner shall accept the inputs defined in §4.2.1.1 and produce a time-ordered sequence of desired states at no slower than 500 Hz.

4.2.2.2 For all output states:

$$\begin{aligned} V_{\text{min}} \leq v \leq V_{\text{max}}, &\ \text{(airspeed)} \\ |\phi| \leq \phi_{\text{max}}, &\ \text{(bank angle)} \\ |\dot{h}| \leq \dot{h}_{\text{max}}, &\ \text{(vertical speed)} \\ \kappa \leq 1/R_{\text{min}}, &\ \text{(turn rate)} \\ n \leq n_{\text{max}}, &\ \text{(load factor)} \end{aligned}$$

Where V_{min}, V_{max}, ϕ_{max}, $\dot{h}_{\text{max}}$, R_{min}, and n_{max} are the UAV tolerances input to the Planner defined in §4.2.1.1.1.

4.2.2.3 The Planner shall validate input presence, ranges, units, and timestamps. Invalid inputs shall result in rejection of the current input and preservation of last valid output for ≤ 1 cycle.

4.2.2.4 The Planner shall log all input and output data during operation. Additionally, all detected errors will be logged with reason and timestamp.

4.2.2.5 The Planner shall adhere to all coding standards defined in §X.X.X.

4.2.3 Waypoint Planner

The Waypoint Planner submodule is responsible for generating feasible waypoint trajectories during autonomous flight. It consumes high level waypoints, and creates a series of states that are determined to be safe and feasible based on the current UAV state and configuration parameters.

4.2.3.1 The Waypoint Planner shall include each commanded waypoint as a goal in the output sequence when inclusion does not violate any constraint in §4.2.2.2, otherwise, it shall mark the waypoint as unreachable and produce the state that best approximates the commanded waypoint while still satisfying all constraints.

4.2.4 RC Mixer

The RC Mixer submodule is responsible for generating desired states based on pilot RC inputs. These inputs will be adjusted to the level defined in the configuration parameters to aid in preserving the safety of the UAV. The adjusted inputs will then be converted to desired states for the UAV to achieve.

4.2.4.1 The RC Mixer shall always produce a valid output regardless of input validity.

4.2.4.2 The RC Mixer shall converge to zero climb rate and zero turn rate in the absence of valid RC inputs within 2 seconds.

4.2.4.3 The RC Mixer shall log all RC inputs and generated outputs with timestamps, any errors detected, and any adjustments made to the inputs to preserve safety.

4.2.5 State Select

The State Select submodule is responsible for multiplexing between the Waypoint Planner and RC Mixer outputs based on the current operating mode. Additionally, the State Select submodule is responsible for determining the state mask passed to the Controller based on the current operating mode.

4.2.5.1 The State Select shall always produce a valid output regardless of input validity.

4.3 Controller

The Controller computes actuator setpoints from planned trajectories and state estimates. It implements required control algorithms and scheduling, ensuring safe and efficient operation of the UAV actuators. The Controller is comprised of four submodules: Preprocessor, PVA Controller, ATL Controller, and Control Distributor. Each submodule contributes to translating planned trajectories into actuator commands.

4.3.1 Interfaces

4.3.1.1 Inputs

4.3.1.1.1 Planned trajectories and state estimates

4.3.1.2 Outputs

4.3.1.2.1 Actuator setpoints

4.3.1.3 Internal

4.3.1.3.1 Internal control signals

4.3.2 Preprocessor

The Preprocessor submodule is responsible for preparing and conditioning inputs for the control loops, including filtering and validation of sensor data before it is used by the controllers.

4.3.2.1 Consume raw sensor measurements as defined in XXX:location

4.3.2.2 Output filtered sensor streams as defined in XXX:location

4.3.2.3 Execute within allocated CPU and power budgets.

4.3.2.4 Maintain performance across expected environmental conditions.

4.3.2.5 Filtering shall never mask critical fault indicators.

4.3.2.6 Modular design with clear interfaces.

4.3.2.7 Follow applicable software development standards.

4.3.3 PVA Controller

The PVA (Position/Velocity/Acceleration) Controller submodule computes control setpoints to track planned trajectories and maintain the UAV along the desired path.

4.3.3.1 Consume waypoints and state estimates as defined in XXX:location

4.3.3.2 Output actuator-level setpoints as defined in XXX:location

4.3.3.3 Execute within allocated CPU and power budgets.

4.3.3.4 Maintain performance across expected environmental conditions.

4.3.3.5 Ensure bounded errors and safe fallback when estimates are invalid.

4.3.3.6 Modular design with clear interfaces.

4.3.3.7 Follow applicable software development standards.

4.3.4 ATL Controller

The ATL (Axis/Attitude/Torque-Level) Controller submodule translates higher-level commands into low-level actuator demands, respecting actuator limits and constraints.

4.3.4.1 Consume desired attitude and torque references as defined in XXX:location

4.3.4.2 Output low-level actuator commands as defined in XXX:location

4.3.4.3 Execute within allocated CPU and power budgets.

4.3.4.4 Maintain performance across expected environmental conditions.

4.3.4.5 Enforce actuator limits and safe modes under faults.

4.3.4.6 Modular design with clear interfaces.

4.3.4.7 Follow applicable software development standards.

4.3.5 Control Distributor

The Control Distributor submodule routes control outputs to the correct actuator channels, mapping logical control channels to physical hardware outputs.

4.3.5.1 Consume controller output commands as defined in XXX:location

4.3.5.2 Output hardware actuator signals as defined in XXX:location

4.3.5.3 Execute within allocated CPU and power budgets.

4.3.5.4 Maintain performance across expected environmental conditions.

4.3.5.5 Prevent misrouting; provide diagnostics and traceability.

4.3.5.6 Modular design with clear interfaces.

4.3.5.7 Follow applicable software development standards.

4.4 Navigation

The Navigation module provides accurate state estimation and sensor selection to support planning and control by running estimators, selecting appropriate sensors, and publishing state estimates. The Navigation module is comprised of two submodules: Sensor Selector and Estimator. Each submodule contributes to robust and accurate state estimation from available sensor inputs.

4.4.1 Interfaces

4.4.1.1 Inputs

4.4.1.1.1 Sensor measurements and configuration parameters

4.4.1.2 Outputs

4.4.1.2.1 State estimates and validity flags

4.4.1.3 Internal

4.4.1.3.1 Internal sensor selection signals

4.4.2 Sensor Selector

The Sensor Selector submodule evaluates available sensor inputs and selects the best sensors for the current conditions, ensuring robust state estimation even when some sensors fail or degrade.

4.4.2.1 Provide prioritized sensor streams and fallback choices.

4.4.2.2 Consume all available sensor feeds and health indicators as defined in XXX:location

4.4.2.3 Output selected sensor feeds for estimator as defined in XXX:location

4.4.2.4 Execute within allocated CPU and power budgets.

4.4.2.5 Maintain performance across expected environmental conditions.

4.4.2.6 Ensure safe fallbacks when preferred sensors are unavailable.

4.4.2.7 Modular design with clear interfaces.

4.4.2.8 Follow applicable software development standards.

4.4.3 Estimator

The Estimator submodule fuses selected sensor inputs using sensor fusion algorithms to produce the best estimate of the system state (position, velocity, attitude, etc.).

4.4.3.1 Provide pose, velocity and other required state variables with uncertainty metrics.

4.4.3.2 Consume selected sensor streams and configuration as defined in XXX:location

4.4.3.3 Output state estimates with timestamps as defined in XXX:location

4.4.3.4 Execute within allocated CPU and power budgets.

4.4.3.5 Maintain performance across expected environmental conditions.

4.4.3.6 Provide safe outputs when inputs are inconsistent or stale.

4.4.3.7 Modular design with clear interfaces.

4.4.3.8 Follow applicable software development standards.

4.5 Logger

The Logger records mission data and provides mechanisms for transmission or storage by collecting telemetry, writing to storage, and/or sending data via communications links. The Logger is comprised of two submodules: Storage and Transmitt. Each submodule handles data persistence and transmission for mission telemetry and diagnostic information.

4.5.1 Interfaces

4.5.1.1 Inputs

4.5.1.1.1 The Logger requires configuration parameters to determine where data is logged and sent, transmission rates, and data routing policies. These parameters will inclue file storage locations and transmission rates and protocols.

4.5.1.1.2 The Logger receives input waypoints from the Companion Computer as defined in 4.1.1.1.2.

4.5.1.1.3 The Logger receives desired states from the Planner as defined in 4.1.1.3.2.

4.5.1.1.4 The Logger receives the state mask from the Planner as defined in 4.1.1.3.3.

4.5.1.1.5 The Logger receives RC inputs from the Pilot as defined in 4.1.1.1.3.

4.5.1.1.6 The Logger receives adjustments to RC inputs from the Planner. These adjustments include trims and scaling factors of each control input.

4.5.1.1.7 The Logger receives raw sensor data from any onboard sensors as defined in 4.1.1.3.7 or external sensors as defined in 4.1.1.1.4.

4.5.1.1.8 The Logger receives state estimates from the Navigation as defined in 4.1.1.3.1.

4.5.1.1.9 The Logger receives raw control outputs from the Controller as defined in 4.1.1.3.5.

4.5.1.1.10 The Logger receives peripheral commands from the Controller as defined in 4.1.1.2.1.

4.5.1.1.11 The Logger receives any error reports from the Planner, Navigation, and Controller as defined in 4.1.1.3.6.

4.5.1.2 Outputs

4.5.1.2.1 The Logger outputs stored log files to the location defined in the configuration parameters.

4.5.1.2.2 The Logger outputs telemetry packets transmitted via communications link as defined in the configuration parameters.

4.5.2 General Requirements

4.5.2.1 The Logger shall support logging of all generated data from onboard modules.

4.5.2.2 The Logger shall support rate limiting for all logged data.

4.5.2.3 The Logger shall adhere to all coding standards defined in §X.X.X.

4.5.3 Storage

The Storage submodule is responsible for persisting telemetry and logs to onboard non-volatile memory, ensuring data is retained for later retrieval and analysis. The location and format of stored data can be defined via configuration parameters. However, if no configuration is provided, the Storage submodule will default to storing data in a predefined location in a standard format as defined in the proceeding requirements.

4.5.3.1 The Storage submodule shall have default storage location of '/logs/' on onboard non-volatile memory.

4.5.3.2 The Storage submodule shall default to storing data from each data stream in a '.log' file with timestamps for each entry.

4.5.3.3 The Storage submodule shall support configuration parameters to define custom storage locations and file formats for all logged data.

4.5.3.4 The Storage submodule shall support a sliding window of log files, based on either time duration or storage size limits defined in configuration parameters. The default shall be 4 hours and 10MB respectively.

4.5.3.5 The Storage submodule shall implement a verification mechanism such as checksums or hashes for all stored data.

4.5.4 Transmitt

The Transmitt submodule handles transmission of selected telemetry data to ground stations or other external systems via available communications links.

4.5.4.1 The Transmitt submodule shall support transmission to up to three external systems simultaneously.

4.5.4.2 The Transmitt submodule shall be support both UDP and TCP transmission protocols.

5 VERIFICATION OF REQUIREMENTS

5.1 System Verification

5.1.1 Processing

5.1.1.1 Verify sufficient compute for all modules.

5.1.1.2 Verify support for real-time scheduling and isolation.

5.1.1.3 Verify fault tolerance and recovery.

5.1.2 Built in HW Capabilities

5.1.2.1 Verify provision of required sensors and actuators.

5.1.2.2 Verify hardware health monitoring.

5.1.2.3 Verify support for hardware expansion where feasible.

5.1.3 HW Interfaces

5.1.3.1 Verify support for standard communication protocols.

5.1.3.2 Verify robust electrical and logical connections.

5.1.3.3 Verify diagnostics for interface health.

5.1.4 HAL

5.1.4.1 Verify exposure of consistent APIs for hardware access.

5.1.4.2 Verify isolation of hardware changes from application logic.

5.1.4.3 Verify support for safe fallback on hardware faults.

5.1.5 Power Management

5.1.5.1 Verify monitoring and regulation of power supply.

5.1.5.2 Verify support for low-power and standby modes.

5.1.5.3 Verify protection against overcurrent and undervoltage.

5.2 Planner Verification

5.2.1 Waypoint Planner

5.2.1.1 Verify consumption of waypoints and configuration parameters.

5.2.1.2 Verify output of planned waypoints.

5.2.1.3 Verify execution within allocated CPU and power budgets.

5.2.1.4 Verify performance across expected environmental conditions.

5.2.1.5 Verify adherence to safety constraints.

5.2.1.6 Verify modular design and clear interfaces.

5.2.1.7 Verify compliance with applicable software development standards.

5.2.2 RC Mixer

5.2.2.1 Verify consumption of required items.

5.2.2.2 Verify output of required items.

5.2.2.3 Verify execution within allocated CPU and power budgets.

5.2.2.4 Verify performance across expected environmental conditions.

5.2.2.5 Verify adherence to safety constraints.

5.2.2.6 Verify modular design and clear interfaces.

5.2.2.7 Verify compliance with applicable software development standards.

5.2.3 State Select

5.2.3.1 Verify consumption of required items.

5.2.3.2 Verify output of required items.

5.2.3.3 Verify execution within allocated CPU and power budgets.

5.2.3.4 Verify performance across expected environmental conditions.

5.2.3.5 Verify adherence to safety constraints.

5.2.3.6 Verify modular design and clear interfaces.

5.2.3.7 Verify compliance with applicable software development standards.

5.3 Controller Verification

5.3.1 Preprocessor

5.3.1.1 Verify consumption of raw sensor measurements.

5.3.1.2 Verify output of filtered sensor streams.

5.3.1.3 Verify execution within allocated CPU and power budgets.

5.3.1.4 Verify performance across expected environmental conditions.

5.3.1.5 Verify that filtering does not mask critical fault indicators.

5.3.1.6 Verify modular design and clear interfaces.

5.3.1.7 Verify compliance with applicable software development standards.

5.3.2 PVA Controller

5.3.2.1 Verify consumption of waypoints and state estimates.

5.3.2.2 Verify output of actuator-level setpoints.

5.3.2.3 Verify execution within allocated CPU and power budgets.

5.3.2.4 Verify performance across expected environmental conditions.

5.3.2.5 Verify bounded errors and safe fallback when estimates are invalid.

5.3.2.6 Verify modular design and clear interfaces.

5.3.2.7 Verify compliance with applicable software development standards.

5.3.3 ATL Controller

5.3.3.1 Verify consumption of desired attitude and torque references.

5.3.3.2 Verify output of low-level actuator commands.

5.3.3.3 Verify execution within allocated CPU and power budgets.

5.3.3.4 Verify performance across expected environmental conditions.

5.3.3.5 Verify enforcement of actuator limits and safe modes under faults.

5.3.3.6 Verify modular design and clear interfaces.

5.3.3.7 Verify compliance with applicable software development standards.

5.3.4 Control Distributor

5.3.4.1 Verify consumption of controller output commands.

5.3.4.2 Verify output of hardware actuator signals.

5.3.4.3 Verify execution within allocated CPU and power budgets.

5.3.4.4 Verify performance across expected environmental conditions.

5.3.4.5 Verify prevention of misrouting and provision of diagnostics and traceability.

5.3.4.6 Verify modular design and clear interfaces.

5.3.4.7 Verify compliance with applicable software development standards.

5.4 Navigation Verification

5.4.1 Sensor Selector

5.4.1.1 Verify prioritized sensor streams and fallback choices.

5.4.1.2 Verify consumption of all available sensor feeds and health indicators.

5.4.1.3 Verify output of selected sensor feeds for estimator.

5.4.1.4 Verify execution within allocated CPU and power budgets.

5.4.1.5 Verify performance across expected environmental conditions.

5.4.1.6 Verify safe fallbacks when preferred sensors are unavailable.

5.4.1.7 Verify modular design and clear interfaces.

5.4.1.8 Verify compliance with applicable software development standards.

5.4.2 Estimator

5.4.2.1 Verify provision of pose, velocity, and other required state variables with uncertainty metrics.

5.4.2.2 Verify consumption of selected sensor streams and configuration.

5.4.2.3 Verify output of state estimates with timestamps.

5.4.2.4 Verify execution within allocated CPU and power budgets.

5.4.2.5 Verify performance across expected environmental conditions.

5.4.2.6 Verify safe outputs when inputs are inconsistent or stale.

5.4.2.7 Verify modular design and clear interfaces.

5.4.2.8 Verify compliance with applicable software development standards.

5.5 Logger Verification

5.5.1 Storage

5.5.1.1 Verify support for retrieval by timestamp and event markers.

5.5.1.2 Verify persistence of telemetry streams and storage configuration.

5.5.1.3 Verify retrieval of stored log files or records by time/index.

5.5.1.4 Verify execution within allocated CPU and power budgets.

5.5.1.5 Verify performance across expected environmental conditions.

5.5.1.6 Verify protection of critical logs and prevention of data loss on faults.

5.5.1.7 Verify modular design and clear interfaces.

5.5.1.8 Verify compliance with applicable software development standards.

5.5.2 Transmitt

5.5.2.1 Verify telemetry prioritization and retransmission.

5.5.2.2 Verify handling of selected telemetry streams and transmission policy.

5.5.2.3 Verify transmission of telemetry packets via communications link.

5.5.2.4 Verify execution within allocated CPU and power budgets.

5.5.2.5 Verify performance across expected environmental conditions.

5.5.2.6 Verify prioritization of safety/health telemetry under degraded links.

5.5.2.7 Verify modular design and clear interfaces.

5.5.2.8 Verify compliance with applicable software development standards.

5.6 Verification Methods

Possible verification methods include:

5.6.1 Inspection:

Inspection is a method of verification consisting of investigation, without the use of special laboratory appliances or procedures, to determine compliance with requirements. Inspection is generally nondestructive and includes (but is not limited to) visual examination, manipulation, gauging, and measurement.

5.6.2 Demonstration:

Demonstration is a method of verification that is limited to readily observable functional operation to determine compliance with requirements. This method shall not require the use of special equipment or sophisticated instrumentation.

5.6.3 Analysis:

Analysis is a method of verification, taking the form of the processing of accumulated results and conclusions, intended to provide proof that verification of a requirement has been accomplished. The analytical results may be based on engineering study, compilation or interpretation of existing information, similarity to previously verified requirements, or derived from lower level examinations, tests, demonstrations, or analyses.

5.6.4 Direct Test:

Test is a method of verification that employs technical means, including (but not limited to) the evaluation of functional characteristics by use of special equipment or instrumentation, simulation techniques, and the application of established principles and procedures to determine compliance with requirements.

5.7 Verify Coverage of Stakeholder Requirements

Paragraph Number	Test Type	Tester's Name	Pass/Fail	Date