muXAV SRATS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | REFERENCES | ISSUE | DATE |
| logoIRT-StExupery |  |  |  |  |
|  |  |  |  |  |
| logoAirbus |  |  |  |  |
|  |  |  |  |  |
| logoONERA |  |  |  |  |
|  |  |  |  |  |
| logoRR |  |  |  |  |
|  |  |  |  |  |
| DASAVIA |  |  |  |  |
| logoSAFRAN |  |  |  |  |
| logoThales |  |  |  |  |
|  |  |  |  |  |

TABLE OF CONTENTS

[1. Introduction 4](#_Toc497989058)

[1.1. Purpose 4](#_Toc497989059)

[1.2. Related Documents 4](#_Toc497989060)

[1.3. Glossary 4](#_Toc497989061)

[1.4. Conventions 5](#_Toc497989062)

[2. Project Overview 8](#_Toc497989063)

[2.1. Context 8](#_Toc497989064)

[2.2. MMS Overview 8](#_Toc497989065)

[3. System Requirements Allocated to Software MMS 9](#_Toc497989066)

[3.1. MMS Functions (top-level) 9](#_Toc497989067)

[3.2. Function Payload Transport (F\_PT) 12](#_Toc497989068)

[3.3. Function Mission Management (F\_MM) 14](#_Toc497989069)

[3.4. Function Flight Control (F\_FC) 19](#_Toc497989070)

[3.5. Function Emergency Landing (F\_EL) 28](#_Toc497989071)

List of Figures

[Figure 1 – MMS Component in its environment 10](#_Toc503445479)

[Figure 2: Internal Block Diagram - RF\_MissionManagementSystem\_Diagram 18](#_Toc503445480)

[Figure 4: Internal Block Diagram - RF\_PayloadTransport\_Diagram 24](#_Toc503445481)

# Introduction

## Purpose

This document presents the system requirements allocated to component Mission Management System controller (MMS) of muXAV.

## Related Documents

### System Requirements

|  |  |
| --- | --- |
| [muXAV-SysReq] | muXAV Specification |

### Plans

|  |  |
| --- | --- |
| [muXAV-ProDef] | muXAV Process Definition |

## Glossary

| Abbreviation | Definition |
| --- | --- |
| AV | Air Vehicle |
| AWC | Adverse Wind Conditions |
| BAT | Battery |
| BITE | Built-In Test Equipment |
| CCA | Common Cause Analysis |
| CL | Confidence Level |
| CMA | Common Mode Analysis |
| CPS | Cyber Physical System |
| ECU | Electronic Control Unit |
| EPS | Electrical Propulsion System |
| FB | Failure Bit |
| FCs | Failure Conditions |
| FCS | Flight Control System |
| F\_BC | Function Braking Control |
| F\_CM | Function Communication Management |
| F\_EL | Function Emergency Landing |
| F\_EM | Function Energy Management |
| F\_FC | Function Flight Control |
| F\_MM | Function Mission Management |
| F\_PC | Function Propulsion Control |
| F\_MM | Function Mission Management |
| F\_PC | Function Propulsion Control |
| FDIR | Fault Detection Isolation Recovery |
| FWC | Favourable Wind Conditions |
| GNC | Guidance Navigation Control |
| HBS | Hydraulic Braking System |
| HVCC | High Voltage Continuous Current |
| IS | International System (of physical units) |
| ISO OSI | International Standard Organisation Open Systems Interconnection |
| kt | Knot |
| LED | Light Electroluminescent Diode |
| MMI | Man Machine Interface |
| MS | Maintenance System |
| MMS | Mission Management System |
| MTOW | Maximal Take-Off Weight |
| n.m | Nautical miles |
| PDI | Parameter Data Item |
| PID | Proportional Integral Derivative control law |
| PEPS | Primary Electrical Power Source |
| PID | Proportional Integral Derivative |
| SEPS | Secondary Electrical Power Source |
| SIC | Standard Icing Conditions |
| SWC | Standard Wind Condition |
| UAV | Unmanned Air Vehicle |

## Conventions

### Requirements Identification

A requirement starts with a tag in square brackets (requirement tag syntax is defined below). And ends with the string “End\*\_\*Req”. The text of the requirement is between the start and the end tag. It can start with a summary label in bolcase.

### Functional Requirements Decomposition

Each function is described in a dedicated section of this document. For scalability reasons, functions are described in a linear organization rather than nested. Requirements of function F\_XX are described in sub-sections as follows:

Function F\_XX Description

1. Parent function

*To be able to reconstruct the functional hierarchy, when applicable.*

1. Interface Requirements

*Provides input and outputs of the function in the form of requirements.*

1. Contract

*Assumptions and Guarantees that are addressable at the function interface level*

1. Sub-functions

Simple list of sub functions

1. Functional Architecture

*Description of how sub-functions are connected to F\_XX boundary interface together with any additional sub-functions communications not using F\_XX boundary interface. These constitute the architecture requirements and can be given textually or in the form of architecture diagrams (e.g. SCADE Architect diagrams).*

1. Behavior Requirements

*Any behavior that is not included in a single sub-function. Must include description of F\_XX sub-functions activation and deactivation. It is expected that a parent function F\_XX’s behavior is that of its sub-functions, and their internal communications described in the functional architecture section.*

### Requirements Tags

Requirements are tagged following pattern [MMS\_F\_XX\_YY\_NN], where:

* REQ\_MMS is the acronym of the requirement (cannot be modified);
* XX is the identifier of the function, for example FC for flight control;
* YY is:
  + ARCH for Architecture Requirements
  + INTF for Interface Requirements,
  + F for Functional Requirements (behavior),
  + ASSU for assumptions about the environment of a function
  + GUAR for guarantees expected from a function
  + SF for Safety Requirements;
* NN is a unique two-digit integer requirement number.

Note-1: F\_XX part is omitted for the top-level.

Note-2: Requirements numbers are allocated incrementally started from 01. When a requirement is suppressed, reuse of the requirement number is prohibited.

Note-3: Assume Guaranteed contracts must be expressed at the scope level where all necessary signals/data are defined and must be expressed in terms of signals/data. Unverifiable contracts are not considered as specifying material, they are informative only.

### Requirement example

### Ranges and Units

Ranges are indicated by the following notation: [a:p:b] (u).

Interpretation:

* from minimum value a
* to maximum value b,
* with increments, p,
* with physical unit u.

### Requirements Standard

The requirements are written following the EARS (Easy Approach to Requirements Syntax) guidelines from presentation of John Terzakis, Intel Corporation. The EARS patterns for requirements are:

* **Ubiquitous:** The <system name> shall <system response>
* **Event-Driven:** WHEN <trigger> <optional precondition> the <system name> shall <system response>
* **Unwanted Behavior:** IF <unwanted condition or event>, THEN the <system name> shall <system response>
* **State-Driven:** WHILE <system state>, the <system name> shall <system response>
* **Optional Feature:** WHERE <feature is included>, the <system name> shall <system response>

### Architecture Standard

The SysML architecture and diagrams are designed according to the following rules:

* There is one standard diagram per parent function level (a function that has child sub-functions). It shows parent boundary interface and propagation to/from sub components,
* Each functional level includes an input output interface table showing the ports of the block diagram and the data routed through each port.

### Data Types

* Dimensions and units are taken from the IS
* Types are given from a shared library component to be shared across different models and simulation levels
* Values should be easily exchanged between tools through system-level and item-level models and scenarios (simple types should be used if data structuring is not available on all platforms).
* Types may be abstract (named only), or given a representation enabling modelling and simulation

# Project Overview

## Context

This document provides the AV level and MMS level system requirements allocated to MMS software. These requirements are grouped into functions and sub-functions, in conformance with ARP 4754 and ARP 4761 approach to system safety.

## MMS Overview

In conformance with [1] and [2], increment 1 layer 2 development of µXAV systems is limited to MMS software. It consists in flight control and mission management aspects. The two functions specified in this document and mapped on MMS software are:

* Payload transport: flight control and mission management,
* Emergency landing: mission cancellation and grounding.

The following diagram presents the MMS in its environment:

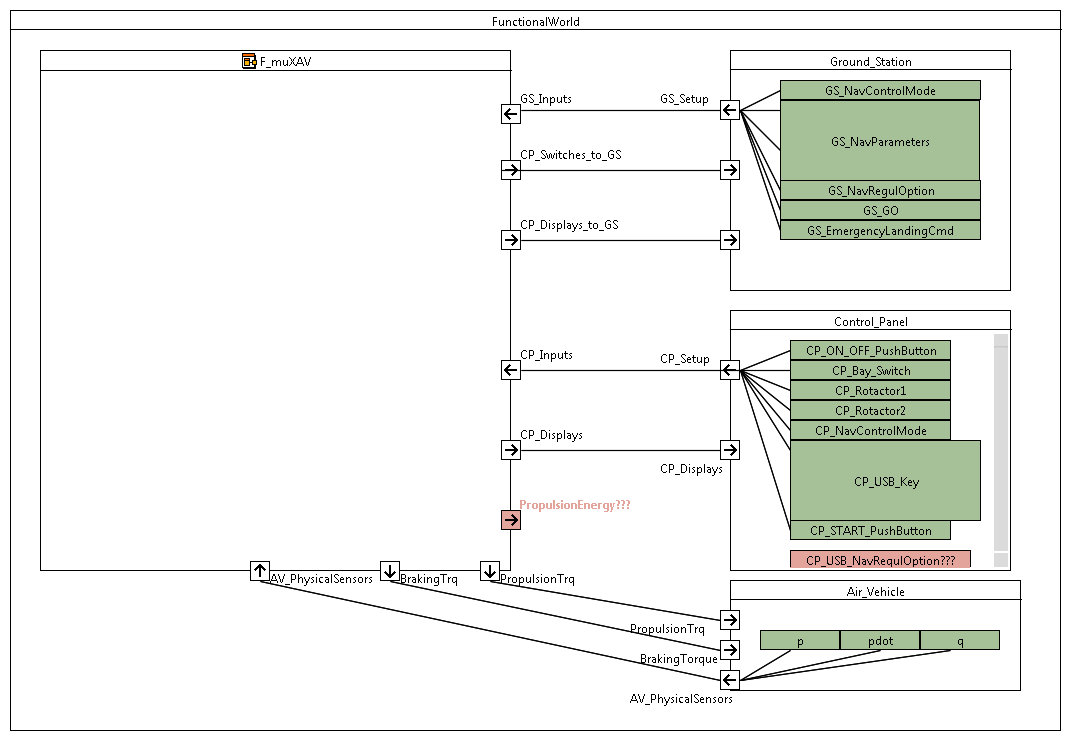


Figure 1 – MMS Component in its environment

# System Requirements Allocated to Software MMS

## MMS Functions (top-level)

### Parent Function

The top-level of MMS has no parent function.

### Interface Requirements

The SysML architecture model (the reference) interface tables are given at each level of the functional decomposition. The interface tables show the ports used in the architecture model (as illustrated in the block diagram figures), and for each port all the data propagated through them. The source block is the one producing the data, while the targets are the destination block.

**[MMS\_INTF\_01]**

**Inputs from Ground Station**

|  |  |  |
| --- | --- | --- |
| **Flow Name** | **Content** | **Comment** |
| GS\_NavigationParameters | Distance [1:1:100] (n.m)  Speed [1:1:250] (kt)  Altitude [-500, 1, 3000] (ft) | Only one of Speed or Altitude drives navigation, according to NavigationOption |
| GS\_NavigationControlMode | {‘RP’, ’A’} | Enumerated type RP=Remotely Piloted A=Autonomous |
| GS\_NavigationRegulOption | {‘SPEED’, ’ALTITUDE’, ’ENERGY’} | Enumerated type |
| GS\_GO | event |  |
| GS\_EmergencyLandingCmd | event | *Note: To be added in multi-system specification* |

**End\_Req**

**[MMS\_INTF\_04]**

**Outputs to Ground Station**

|  |  |  |
| --- | --- | --- |
| **Flow Name** | **Content** | **Comment** |
| CP\_Switches | Power: Boolean;  Mode: {‘RP’,’A’};  Bay: {‘OPEN’,’CLOSED’}  START: Boolean;  Rotactor1: [1:1:9] (kg)  Rotactor2: [1:1:9] (kg) |  |
| CP\_Displays | READY: Boolean;  CANCELLED: Boolean;  COMPLETE: Boolean;  ABORTED: Boolean;  PrimarySource [1:1:100] (%)  SecondarySource [1:1:100] (%) | Capacities are relative to full charge |
| PropulsionEnergy | PrimarySourceCapacity [1:1:99999] (J)  SecondarySourceCapacity [1:1:9999] (J) | Capacities are absolute in Joule |

**End\_Req**

|  | Direction | Block Source | Block Target | Comment |
| --- | --- | --- | --- | --- |
| Ports |  |  |  |  |
| **CP\_Displays\_to\_GS** | **in** |  |  |  |
| Related Data |  |  |  |  |
| MissionREADY |  | RF\_MM\_MissionManagement | Ground\_Station |  |
| MissionCOMPLETED |  | RF\_MM\_MissionManagement | Ground\_Station |  |
| MissionCANCELLED |  | RF\_MM\_MissionManagement | Ground\_Station |  |
| MissionABORTED |  | RF\_MM\_MissionManagement | Ground\_Station |  |
| **CP\_Switches\_to\_GS** | **in** |  |  |  |
| **GS\_Setup** | **out** |  |  |  |
| Related Data |  |  |  |  |
| GS\_NavControlMode |  | Ground\_Station | RF\_MM\_MissionManagement |  |
| GS\_NavParameters |  | Ground\_Station | RF\_MM\_MissionManagement |  |
| GS\_NavRegulOption |  | Ground\_Station | RF\_MM\_MissionManagement |  |
| GS\_GO |  | Ground\_Station | RF\_MM\_MissionManagement |  |
| GS\_EmergencyLanding |  | Ground\_Station | RF\_EL |  |

**[MMS\_INTF\_02]**

**Inputs from Control Panel**

|  |  |  |
| --- | --- | --- |
| **Flow Name** | **Content** | **Comment** |
| ON\_OFF\_PushButton | event | Power on, to boot all systems |
| START\_PushButton | event |  |
| ModeSwitch | {‘A’,’RP’} | Two stable states  A=Autonomous, RP=Remotely Piloted |
| BaySwitch | {‘OPEN’,’CLOSED’} | Two stable states |
| PayloadMass | integer | *Note: ROTACTORS ONLY, from CP (GitHub Issue #14)* |
| USB\_Key | Distance [1:1:100] (n.m)  Speed [1:1:250] (kt)  Altitude [-500, 3000] (ft) |  |

**End\_Req**

**[MMS\_INTF\_05]**

**Outputs to Control Panel**

|  |  |  |
| --- | --- | --- |
| **Flow Name** | **Content** | **Comment** |
| MissionCANCELLED | true/false |  |
| MissionCOMPLETE | true/false |  |
| MissionABORTED | true/false |  |
| READY | True/false |  |
| PrimarySource | [1:1:100] (%) |  |
| SecondarySource | [1:1:100] (%) |  |

**End\_Req**

|  | Direction | Block Source | Block Target | Comment |
| --- | --- | --- | --- | --- |
| Ports |  |  |  |  |
| **CP\_Displays** | **in** |  |  |  |
| Related Data |  |  |  |  |
| MissionREADY |  | RF\_MM\_MissionManagement | Control\_Panel |  |
| MissionCOMPLETED |  | RF\_MM\_MissionManagement | Control\_Panel |  |
| MissionCANCELLED |  | RF\_MM\_MissionManagement | Control\_Panel |  |
| MissionABORTED |  | RF\_MM\_MissionManagement | Control\_Panel |  |
| **CP\_Setup** | **out** |  |  |  |
| Related Data |  |  |  |  |
| CP\_START\_PushButton |  | Control\_Panel | RF\_MM\_MissionManagement |  |
| CP\_USB\_Key |  | Control\_Panel | RF\_MM\_MissionManagement |  |
| CP\_NavControlMode |  | Control\_Panel | RF\_MM\_MissionManagement |  |
| CP\_Rotactor2 |  | Control\_Panel | RF\_MM\_MissionManagement |  |
| CP\_Rotactor1 |  | Control\_Panel | RF\_MM\_MissionManagement |  |
| CP\_Bay\_Switch |  | Control\_Panel | RF\_MM\_MissionManagement |  |
| CP\_ON\_OFF\_PushButton |  | Control\_Panel | RF\_MM\_MissionManagement |  |
| CP\_USB\_NavRegulOption??? |  | Control\_Panel | RF\_MM\_MissionManagement |  |

**[MMS\_INTF\_03]**

**Input Physical Parameters**

|  |  |  |
| --- | --- | --- |
| **Flow Name** | **Content** | **Comment** |
| p | float | Drives distance |
| pdot | float | Drives speed |
| q | float | Drives altitude and pitch |

**End\_Req**

**[MMS\_INTF\_06]**

**Output Physical Parameters**

|  |  |  |
| --- | --- | --- |
| **Flow Name** | **Content** | **Comment** |
| PropulsionTorque | [-10e-6:0.0001:+10e-6] (N.m) | When Torque > minTorque , set to zero otherwise |
| BrakingTorque | [-10e-6:0.0001:+10e-6] (N.m) | When Torque < -minTorque, set to zero otherwise. |

**End\_Req**

|  | Direction | Block Source | Block Target | Comment |
| --- | --- | --- | --- | --- |
| Ports |  |  |  |  |
| **PropulsionTrq** | **in** |  |  |  |
| Related Data |  |  |  |  |
| FC\_PropulsionTorque |  | RF\_FC\_FlightControl | Air\_Vehicle |  |
| **BrakingTorque** | **in** |  |  |  |
| Related Data |  |  |  |  |
| FC\_BrakingTorque |  | RF\_FC\_FlightControl | Air\_Vehicle |  |
| EL\_BrakingTorque |  | RF\_EL | Air\_Vehicle |  |
| **AV\_PhysicalSensors** | **out** |  |  |  |
| Related Data |  |  |  |  |
| pdot |  | Air\_Vehicle | RF\_EL, RF\_FC\_FlightControl |  |
| q |  | Air\_Vehicle | RF\_EL, RF\_FC\_FlightControl |  |
| p |  | Air\_Vehicle | RF\_EL, RF\_FC\_FlightControl, RF\_EM\_EnergyViability |  |

### Contract

### Sub-functions

MMS software contains two functions: Payload Transport (F\_PT) and Emergency Landing (F\_EL).

Payload Transport (F\_PT) is the main function. It manages AV’s control and mission until a hazardous situation is detected, in which case the mission is aborted.

Emergency Landing (F\_EL) detects occurrence of hazardous situations not detected by F\_PT. In such situations F\_EL aborts F\_PT (mission cancellation), takes over, and lands the AV.

### Functional Architecture

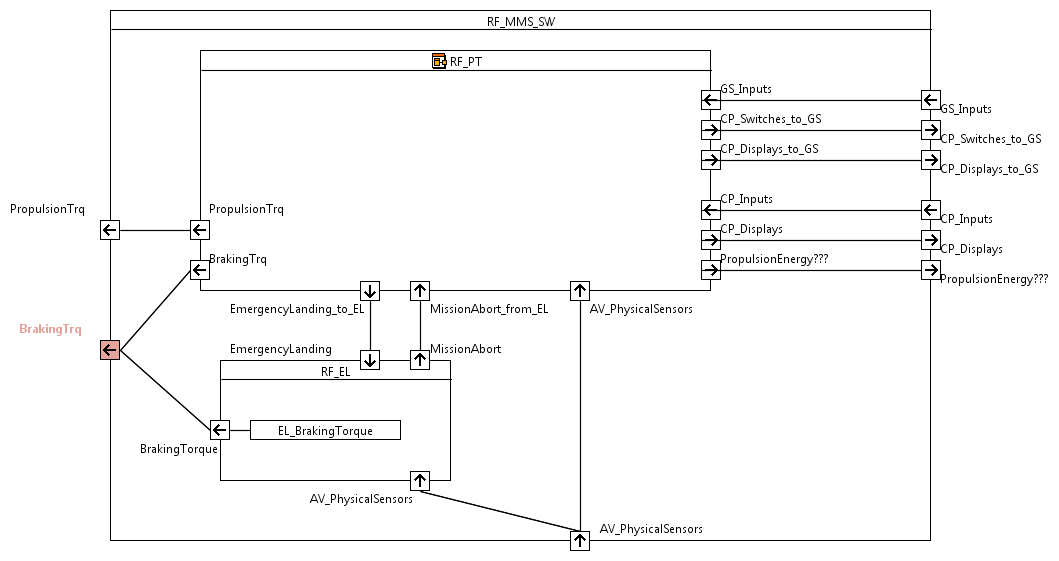


Figure : Internal Block Diagram - F\_MissionManagementSystem\_Diagram

**[MMS\_ARCH\_07]**

**F\_EL aborts F\_PT**

When F\_EL detects occurrence of hazardous situations not detected by F\_PT, F\_EL shall abort F\_PT by sending the signal **MissionAbort** and take over the control to land the Air Vehicle.

**End\_Req**

*Note: Although both F\_PT and F\_EL can output a Braking Torque command, there is no need for arbitration or combination since when emergency landing is triggered F\_PT is deactivated.*

**[MMS\_ARCH\_08]**

**F\_PT triggers F\_EL**

WHEN F\_MM part of F\_PT detects hazardous escapes in the flight safety envelopes, F\_PT shall trigger F\_EL Emergency Landing part by sending the signal **EmergencyLanding** to F\_EL and F\_PT shall then be deactivated.

**End\_Req**

### Behavior Requirements

**[MMS\_FUNC\_09]**

**MMS Power On**

When MMS is powered on (Control Panel ON\_OFF\_PushButton), both F\_PT and F\_EL are activated.

**End\_Req**

## Function Payload Transport (F\_PT)

### Parent Function

F\_PT is a sub-function of MMS top-level function.

### Interface Requirements

**[MMS\_F\_PT\_INTF\_10]**

**F\_PT has all MMS inputs**

F\_PT uses all MMS input interface described in [MMS\_INTF\_01] to [MMS\_INTF\_03]

**End\_Req**

**[MMS\_F\_PT\_INTF\_11]**

**F\_PT has all MMS outputs**

F\_PT uses all MMS output interface described in [MMS\_INTF\_04] to [MMS\_INTF\_06]

**End\_Req**

**[MMS\_F\_PT\_INTF\_12]**

**F\_PT input MissionAbort from F\_EL**

|  |  |  |
| --- | --- | --- |
| **Flow Name** | **Content** | **Comment** |
| MissionAbort | true/false | Signal sent by F\_EL when it diagnoses that F\_PT does not detect dangerous behaviour of the AV |

**End\_Req**

**[MMS\_F\_PT\_INTF\_13]**

**F\_PT output EmergencyLanding to F\_EL**

|  |  |  |
| --- | --- | --- |
| **Flow Name** | **Content** | **Comment** |
| EmergencyLanding | true/false | True when F\_PT decides to abort the mission. Leads to soft landing when enough electrical energy is still available, to hard landing otherwise |

**End\_Req**

|  | Direction | Block Source | Block Target | Comment |
| --- | --- | --- | --- | --- |
| Ports |  |  |  |  |
| **GS\_Inputs** | **in** |  |  |  |
| Related Data |  |  |  |  |
| GS\_NavControlMode |  | Ground\_Station | RF\_MM\_MissionManagement |  |
| GS\_NavParameters |  | Ground\_Station | RF\_MM\_MissionManagement |  |
| GS\_NavRegulOption |  | Ground\_Station | RF\_MM\_MissionManagement |  |
| GS\_GO |  | Ground\_Station | RF\_MM\_MissionManagement |  |
| GS\_EmergencyLanding |  | Ground\_Station | RF\_EL |  |
| **CP\_Inputs** | **in** |  |  |  |
| Related Data |  |  |  |  |
| CP\_ON\_OFF\_PushButton |  | Control\_Panel | RF\_MM\_MissionManagement |  |
| CP\_Bay\_Switch |  | Control\_Panel | RF\_MM\_MissionManagement |  |
| CP\_Rotactor1 |  | Control\_Panel | RF\_MM\_MissionManagement |  |
| CP\_Rotactor2 |  | Control\_Panel | RF\_MM\_MissionManagement |  |
| CP\_NavControlMode |  | Control\_Panel | RF\_MM\_MissionManagement |  |
| CP\_USB\_Key |  | Control\_Panel | RF\_MM\_MissionManagement |  |
| CP\_START\_PushButton |  | Control\_Panel | RF\_MM\_MissionManagement |  |
| CP\_USB\_NavRegulOption??? |  | Control\_Panel | RF\_MM\_MissionManagement |  |
| **AV\_PhysicalSensors** | **in** |  |  |  |
| Related Data |  |  |  |  |
| p |  | Air\_Vehicle | RF\_FC\_FlightControl, RF\_EM\_EnergyViability |  |
| pdot |  | Air\_Vehicle | RF\_FC\_FlightControl |  |
| q |  | Air\_Vehicle | RF\_FC\_FlightControl |  |
| **MissionAbort\_from\_EL** | **in** |  |  |  |
| **CP\_Switches\_to\_GS** | **out** |  |  |  |
| **CP\_Displays\_to\_GS** | **out** |  |  |  |
| Related Data |  |  |  |  |
| MissionREADY |  | RF\_MM\_MissionManagement | Ground\_Station, Control\_Panel |  |
| MissionCOMPLETED |  | RF\_MM\_MissionManagement | Ground\_Station, Control\_Panel |  |
| MissionCANCELLED |  | RF\_MM\_MissionManagement | Ground\_Station, Control\_Panel |  |
| MissionABORTED |  | RF\_MM\_MissionManagement | Ground\_Station, Control\_Panel |  |
| **CP\_Displays** | **out** |  |  |  |
| Related Data |  |  |  |  |
| MissionREADY |  | RF\_MM\_MissionManagement | Ground\_Station, Control\_Panel |  |
| MissionCOMPLETED |  | RF\_MM\_MissionManagement | Ground\_Station, Control\_Panel |  |
| MissionCANCELLED |  | RF\_MM\_MissionManagement | Ground\_Station, Control\_Panel |  |
| MissionABORTED |  | RF\_MM\_MissionManagement | Ground\_Station, Control\_Panel |  |
| **PropulsionEnergy???** | **out** |  |  |  |
| **PropulsionTrq** | **out** |  |  |  |
| Related Data |  |  |  |  |
| FC\_PropulsionTorque |  | RF\_FC\_FlightControl | Air\_Vehicle, RF\_EM\_EnergyViability, RF\_CM\_Comm\_MMS |  |
| **BrakingTrq** | **out** |  |  |  |
| Related Data |  |  |  |  |
| FC\_BrakingTorque |  | RF\_FC\_FlightControl | Air\_Vehicle, RF\_EM\_EnergyViability, RF\_CM\_Comm\_MMS |  |
| **EmergencyLanding\_to\_EL** | **out** |  |  |  |
| Related Data |  |  |  |  |
| MM\_EmergencyLanding |  | RF\_MM\_MissionManagement | RF\_EL |  |
| GS\_EmergencyLanding |  | Ground\_Station | RF\_EL |  |

### Contract

#### Assumptions

No assumptions kept (A to E), they all concern the operational specification (scenarios), environment model.

*Note: No assumption is made about mission parameter values entered from ground station or via control panel usb key (they may be invalid input data).*

#### Guarantees

**[MMS\_F\_PT\_GUAR\_67]**

**Viability guarantee**

No take-off if energy aboard is incompatible with mission completion under standard Weather Conditions (SWC) assumption.

**End Req**

**[MMS\_F\_PT\_GUAR\_68]**

**Cancellation Signaling**

Any mission cancelation is signaled on CP and to GS

**End Req**

1. (moved above), kept for alphabetic index stability
2. (moved above), kept for alphabetic index stability
3. Climb safety constraints:
   * max take-off speed: 75kt,
   * climb rate guaranteed in [0.3, 3] m/s,
   * precision of flight level capture: +/- 50ft,
   * precision of speed capture: +/- 5kt.
4. Cruise safety constraints:
   * Minimum flight level: 500ft,
   * Maximum flight level: 1500ft,
   * Flight level precision: +/- 50ft,
   * Maximum speed: 125kt,
5. Descent safety constraints:
   * descent rate guaranteed in [0.1, 1] m/s,
   * maximum landing speed: 25kt.
6. RP mode is identical to A mode when no command is sent to the AV. All control laws are left invariant by ‘A’↔‘RP’ commutations,
7. In ideal weather conditions (no wind, no icing) all initiated missions run to completion,
8. In GWC, at MTOW, when the range is lower than 100 n.m, the mission success ratio is at least 90% (95% confidence level),
9. In AWC, at MTOW, when the range is lower than 50 n.m, the success ratio is at least 60% (95% confidence level),
10. When compensation of external disturbances (keeping the AV within the flight safety envelope) or mission completion are no longer possible, soft landing is ensured (see F\_EL),
11. In J conditions, if propulsion is no longer available, hard landing is ensured (see F\_EL),

### Sub-functions

F\_PT’s behaviour results from the interaction of four sub-functions:

1. F\_MM: **M**ission **M**anagement, in charge of mission preparation and supervision,
2. F\_FC: **F**light **C**ontrol, in charge of mechanical control of the AV’s body,
3. F\_EM: **E**nergy **M**anagement, in charge of the mission’s energetic viability,
4. F\_CM: **C**ommunication **M**anagement, gateway making MMS’ functions communicate with the outer world.

The sub-functions communicate directly to each other.

### Functional Architecture

**[MMS\_F\_PT\_ARCH\_14]**

**F\_PT sub-functions communications**

Sub functions communications illustrated on the following figure should be detailed in this section.

**End\_Req**



*Figure 2: Functional decomposition and information flows*

*of function Payload Transport (F\_PT)*

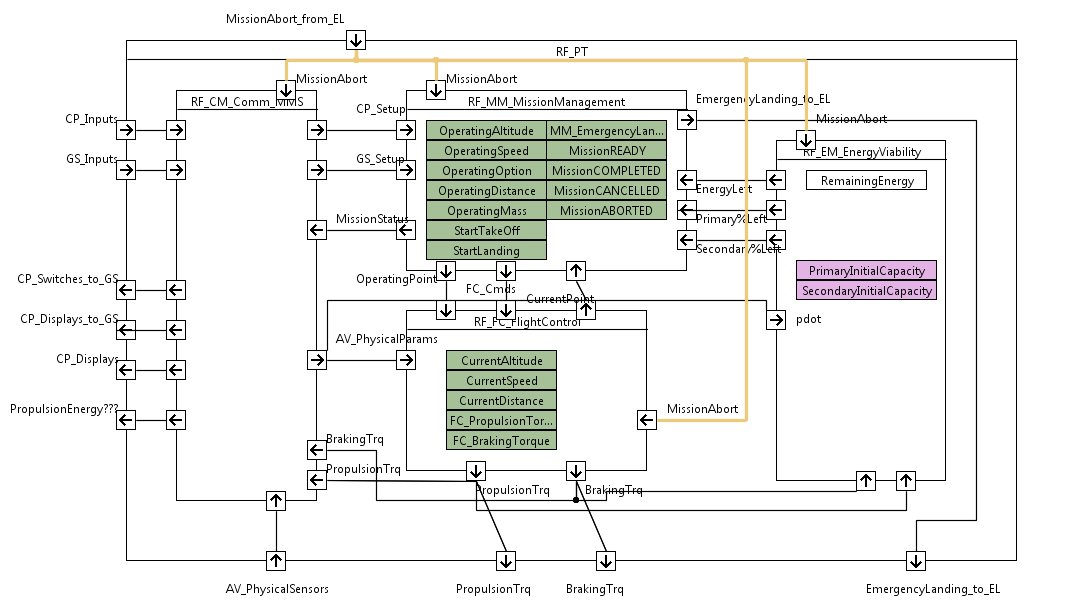


Figure : Internal Block Diagram - RF\_PayloadTransport\_Diagram

### Behavior Requirements

**[MMS\_F\_PT\_FUNC\_15]**

When F\_PT is activated all sub-functions shall be activated.

**End\_Req**

Detailed specification of F\_PT is at sub-function level (F\_MM, F\_FC, F\_EM, and F\_CM).

## Function Mission Management (F\_MM)

### Parent Function

F\_MM is a sub-function of F\_PT.

### Interface Requirements

**[MMS\_F\_PT\_F\_MM\_INTF\_16]**

**F\_MM inputs**

|  |  |  |
| --- | --- | --- |
| **Flow Name** | **Content** | **Comment** |
| NavigationParameters | Distance [1:1:100] (n.m)  Speed [1:1:250] (kt)  Altitude [-500: 1: 3000] (ft) | From F\_CM |
| NavigationMode | {‘RP’,’A’} | From F\_CM |
| NavigationOption | {‘SPEED’,’ENERGY’} | From F\_CM |
| GO | Event | From F\_CM |
| MissionAbort | Event | From F\_FC or F\_EL |
| ON/OFF\_PushButton | Event | From F\_CM |
| START\_PushButton | Event | From F\_CM |
| ModeSwitch | {‘A’,’RP’} | From F\_CM |
| BaySwitch | {‘OPEN’,’CLOSED’} | From F\_CM |
| PayloadMass | [1:0.1:5kg] | From F\_CM |
| USB\_Key | Distance [1:1:100] (n.m)  Speed [1:1:250] (kt)  Altitude [-500:1:3000] (ft) | From F\_CM |
| EstimatedTotalMass | [5,10]kg | **From F\_FC**: instantaneous empty AV mass + payload mass + icing mass |
| CurrentRange | [1:1:1000000](m) | **From F\_FC**: continuously computed estimated ground distance flown since take-off |
| CurrentSpeed | [1:1:500] (km/h) | **From F\_FC**: continuously computed instantaneous air speed |
| CurrentAltitude | [-200:1:1000] (m) | **From F\_FC**: continuously computed instantaneous altitude |
| EnergyLevel | [0:1:500] (kJ) | **From F\_EM**: current energy storage |

**End\_Req**

**[MMS\_F\_PT\_F\_MM\_INTF\_17]**

**F\_MM outputs**

|  |  |  |
| --- | --- | --- |
| **Flow Name** | **Content** | **Comment** |
| MissionCANCELLED | true/false | To F\_CM |
| MissionCOMPLETE | true/false | To F\_CM |
| MissionABORTED | true/false | To F\_CM |
| EmergencyLanding | event | **To F\_EL:** diagnosed catastrophic situation |
| StartTakeOff | event | **To F\_FC** |
| StartLanding | event | **To F\_FC:**  *Note: (This is not used)* |
| OperatingPoint | Altitude and Speed | **To F\_FC**. (same ranges as the ones at interface) |
| OperatingMode | value in {‘SPEED’,’ALTITUDE’,’ENERGY’} | **To F\_FC** |
| MissionRange | Same range as the Distance mission parameter | **To F\_FC** used to trigger landing when the mission range is within reach |

**End\_Req**

|  | Direction | Block Source | Block Target | Comment |
| --- | --- | --- | --- | --- |
| Ports |  |  |  |  |
| EnergyLevel | in |  |  |  |
| PrimarySource | in |  |  |  |
| SecondarySource | in |  |  |  |
| GS\_Setup | in |  |  |  |
| Related Data |  |  |  |  |
| GS\_NavControlMode |  | Ground\_Station | RF\_MM\_MissionManagement |  |
| GS\_NavParameters |  | Ground\_Station | RF\_MM\_MissionManagement |  |
| GS\_NavRegulOption |  | Ground\_Station | RF\_MM\_MissionManagement |  |
| GS\_GO |  | Ground\_Station | RF\_MM\_MissionManagement |  |
| GS\_EmergencyLanding |  | Ground\_Station | RF\_EL |  |
| CP\_Setup | in |  |  |  |
| Related Data |  |  |  |  |
| CP\_ON\_OFF\_PushButton |  | Control\_Panel | RF\_MM\_MissionManagement |  |
| CP\_Bay\_Switch |  | Control\_Panel | RF\_MM\_MissionManagement |  |
| CP\_Rotactor1 |  | Control\_Panel | RF\_MM\_MissionManagement |  |
| CP\_Rotactor2 |  | Control\_Panel | RF\_MM\_MissionManagement |  |
| CP\_NavControlMode |  | Control\_Panel | RF\_MM\_MissionManagement |  |
| CP\_USB\_Key |  | Control\_Panel | RF\_MM\_MissionManagement |  |
| CP\_START\_PushButton |  | Control\_Panel | RF\_MM\_MissionManagement |  |
| CP\_USB\_NavRegulOption??? |  | Control\_Panel | RF\_MM\_MissionManagement |  |
| CurrentPoint | in |  |  |  |
| Related Data |  |  |  |  |
| CurrentAltitude |  | RF\_FC\_FlightControl | RF\_MM\_MissionManagement |  |
| CurrentSpeed |  | RF\_FC\_FlightControl | RF\_MM\_MissionManagement |  |
| CurrentDistance |  | RF\_FC\_FlightControl | RF\_MM\_MissionManagement |  |
| MissionAbort | in |  |  |  |
| MissionStatus | out |  |  |  |
| Related Data |  |  |  |  |
| MissionREADY |  | RF\_MM\_MissionManagement | Ground\_Station, Control\_Panel |  |
| MissionCOMPLETED |  | RF\_MM\_MissionManagement | Ground\_Station, Control\_Panel |  |
| MissionCANCELLED |  | RF\_MM\_MissionManagement | Ground\_Station, Control\_Panel |  |
| MissionABORTED |  | RF\_MM\_MissionManagement | Ground\_Station, Control\_Panel |  |
| EmergencyLanding\_to\_EL | out |  |  |  |
| Related Data |  |  |  |  |
| MM\_EmergencyLanding |  | RF\_MM\_MissionManagement | RF\_EL |  |
| GS\_EmergencyLanding |  | Ground\_Station | RF\_EL |  |
| OperatingPoint | out |  |  |  |
| Related Data |  |  |  |  |
| OperatingAltitude |  | RF\_MM\_MissionManagement | RF\_FC\_FlightControl |  |
| OperatingSpeed |  | RF\_MM\_MissionManagement | RF\_FC\_FlightControl |  |
| OperatingOption |  | RF\_MM\_MissionManagement | RF\_FC\_FlightControl |  |
| OperatingDistance |  | RF\_MM\_MissionManagement | RF\_FC\_FlightControl |  |
| OperatingMass |  | RF\_MM\_MissionManagement | RF\_FC\_FlightControl |  |
| FC\_Cmds | out |  |  |  |
| Related Data |  |  |  |  |
| StartTakeOff |  | RF\_MM\_MissionManagement | RF\_FC\_FlightControl |  |
| StartLanding |  | RF\_MM\_MissionManagement | RF\_FC\_FlightControl |  |

### Contract

#### Assumptions

1. The physical parameters received from F\_EM and F\_FC are accurate or conservative (lower than 10% excess),

*Note:*

*Because of:*

1. *systematic contract-based specification,*
2. *systematic contract-based implementation (system modelling, software and hardware development),*
3. *an assumption on sufficient effectiveness of development assurance,*
4. *assumption A,*

*it has been decided in the use case that F\_MM has no mechanism to mitigate a wrong perception of µXAV’s physical state: dynamics and energy capacity. F\_MM does not try, using its own estimation algorithm, to cross check the physical data it receives from F\_CM, F\_EM and F\_FC.*

*In the use case, there is no functional redundancy motivated by possible development assurance failure. If functional redundancy is introduced, it is on request of the safety group, to cope with F\_PT’s Functional Failure Set (FFS). F\_PT’s FFS takes into account the physical failures (introduced at increment 3) and the development faults (likely to be present since increment 1). The double detection of safety escapes by F\_PT and F\_EL in MMS is a provision whose safety group’s justification appears at increment 3: the “no single component failure” objective.*

*This functional redundancy with (intended later) independence is not some “on the fly"[[1]](#footnote-1) defensive approach to development errors in F\_FC and F\_MM.*

#### Guarantees

1. Missions cancelled for energy reasons can be proven infeasible (under assumption of existence of an embedded MS’ flight data recorder for a posteriori analysis),

### Sub-functions

F\_MM has no sub-functions.

### Functional Architecture

No internal architecture for F\_MM.

### Behavior Requirements

**[MMS\_F\_PT\_F\_MM\_FUNC\_18]**

**Units: MM converts aeronautical units into S.I.**

GS and CP MMIs display values in aeronautical units (knots, nautical miles, feet etc.) but system and item implementation are based on IS units (m, kg, s).

**End\_Req**

**[MMS\_F\_PT\_F\_MM\_FUNC\_19]**

**Mission Setup activation**

When AV is powered ON, F\_MM shall start the initialization phase.

**End\_Req**

**[MMS\_F\_PT\_F\_MM\_FUNC\_20]**

**Mission Setup sequence**

The initialization phase shall perform sequentially:

* Payload boarding
* Mission initialization

**End\_Req**

**[MMS\_F\_PT\_F\_MM\_FUNC\_21]**

**Payload boarding**

A payload boarding sequence consists in:

1. The bay is opened using the BAY SWITCH (BaySwitch is set to OPEN)
2. The payload is boarded in the bay (which is not sensed by the Air Vehicle)
3. The bay is closed using the BAY SWITCH (BaySwitch is set to CLOSE)

**End\_Req**

**[MMS\_F\_PT\_F\_MM\_FUNC\_22]**

**Mission initialization:**

When Payload Boarding has completed successfully, Mission Set Up is started. Mission Set Up consists in:

* Record Payload mass according to input channel precedence logic
* Record Mission parameters according to input channel precedence logic
* Compute mission viability
* Turn on the READY light in control panel or

**End\_Req**

**[MMS\_F\_PT\_F\_MM\_FUNC\_23]**

**Mission initialization:** *Note (APP): Alternative record mass (APP) seems over specified from System point of view*

When the bay is closed, the mass must be recorded from the CP’s rotator or the USB key.

**End\_Req**

**[MMS\_F\_PT\_F\_MM\_FUNC\_24]**

**Mission initialization: parameters input channels precedence.**

The rotators supersede any other means, unless value 99 is configured on the two wheels. In RP mode, GS supersedes the USB key, whereas in A mode the key supersedes any GS input.

**End\_Req**

**[MMS\_F\_PT\_F\_MM\_FUNC\_25]**

**Mission initialization: initial viability**

When mission viability is computed:

* If the mission is viable, READY is sent to F\_CM (to be forwarded to control panel)
* If the mission is not viable, CANCEL stuff

**End\_Req**

**[MMS\_F\_PT\_F\_MM\_FUNC\_26]**

**Mission setup: completion**

When START PushButton from control panel or GO from Ground Station is received, if mission initialization has successfully completed, StartTakeOff is sent to F\_FC to start the flight. This completes and terminates the Setup phase and the Air Vehicle enters the flight mode (starting with TakeOff phase).

**End\_Req**

**OPERATING POINT UPDATES**

The operating point is the reference value F\_FC regulates on at a given instant. When it regulates on an altitude value or on a speed value, these values are sent by F\_MM to F\_FC. When F\_FC is in energy mode, the altitude or the speed reference value is computed by F\_FC.

**[MMS\_F\_PT\_F\_MM\_FUNC\_27]**

While regulation mode is on altitude or speed, F\_MM shall compute the operating point reference value and sends it to F\_FC to regulate on.

**End\_Req**

**[MMS\_F\_PT\_F\_MM\_FUNC\_28]**

While regulation mode is on energy, F\_FC shall compute the operating point reference value by itself to regulate on.

**End\_Req**

**VIABILITY COMPUTATION (initial and updates)**

Viability analysis is energetic assessment of the mission’s feasibility as defined at initial conditions. It is done before take-off to prevent launching the drone on unfeasible missions. This analysis is then periodically updated during flight (low rate, 0.1 Hz) as bad flight weather conditions may invalidate the standardized assumptions used at set-up phase.

*In-flight updates*

Viability computation in-flight updates are using the same algorithm as initial computation before TakeOff, with distance equated to the remaining distance to fly, is repeated at periodic rate (f\_viability = 0.1Hz) and with two other response surfaces (similar structures, but discount of a flat rate climb energy).

**[MMS\_F\_PT\_F\_MM\_FUNC\_29]**

**Mission Viability: Drone Data for Needed Energy**

Four surrogate models[[2]](#footnote-2) are loaded and updated independently of the software. The models are tabulated functions or response surfaces that give the estimated needed energy from the mission parameters:

* Viability\_Amode\_initial,
* Viability\_Amode\_cruise,
* Viability\_RPmode\_initial,
* Viability\_RPmode\_cruise.

These four PDIs are not mission dependent. They are drone dependent. They depend on the drone’s empty mass, engines, and energy capacity. Each PDI is a table in the form:

EnergyNeeded = ViabilityTable [ PayloadMass x Distance x Altitude x Speed ]

**End\_Req**

**[MMS\_F\_PT\_F\_MM\_FUNC\_30]**

**Mission Viability: interpolation tables**

From the tables, F\_MM shall select the 16 nearest neighbors and extract energy levels for each of these neighbors. F\_MM shall compute the energy needed as an inverse distance based interpolation of the 16 neighboring values navigation parameter values and PayloadMass.

**End\_Req**

**[MMS\_F\_PT\_F\_MM\_FUNC\_31]**

**Mission Viability: needed energy computation**

Compute the Euclidian distance disti between the selected drone parameters (mdrone, speedsetpoint, distancesetpoint, altitudesetpoint) to its 4 neighbors in viability table rowindex (mindex, sindex, dindex, aindex), for index in 1..16.

Compute the needed energy from the 16 energy levels needed for each index with:

If a disti is equal to 0 (within an epsilon tolerance) the energy at that point Ei = E(mi, si, di, ai)

Otherwise compute interpolation with:

Where

**End\_Req**

***Note:*** *This algorithm is documented in* [*https://en.wikipedia.org/wiki/Inverse\_distance\_weighting*](https://en.wikipedia.org/wiki/Inverse_distance_weighting)*. It is suggested unless other recommendations to use the algorithm parameter p value of 1*

*Note: We assume tables are sorted. The neighbors search can be sequential. From the 4 parameters, 2 neighbors in each dimension, a vector of 16 neighboring quadruplets is constructed.*

**[MMS\_F\_PT\_F\_MM\_FUNC\_32]**

**Mission Viability: on-board available energy capacity**

Before TakeOff, comes directly from inputs?

**End\_Req**

**[MMS\_F\_PT\_F\_MM\_FUNC\_33]**

**Mission Viability: decision**

F\_MM compares the result of needed energy level computation to the energy capacity available on board and makes the viability decision with 10% energy margin in RP mode, and 30% in A mode.

**End\_Req**

**MISSION TERMINATION**

**[MMS\_F\_PT\_F\_MM\_FUNC\_34]**

F\_MM monitors CurrentRange (received from F\_FC).

When (Distance – CurrentRange) < GlideDistance (CurrentAltitude), F\_MM shall activate landing.

GlideDistance is a tabulated function that gives the ground distance needed to reach altitude zero from CurrentAltitude.

**End\_Req**

**[MMS\_F\_PT\_F\_MM\_FUNC\_35]**

When F\_MM activates landing, F\_MM shall freeze any change on the operating point. Careful if mode is Energy, when the point is fixed by F\_FC directly. Freezing must supersede F\_FC operating point computation.  
**End\_Req**

## Function Flight Control (F\_FC)

### Parent Function

This function is at the top-level of MMS software (together with Emergency Landing F\_EL).

### Interface Requirements

**[MMS\_F\_PT\_F\_FC\_INTF\_36]**

**F\_FC physical parameter inputs**

|  |  |  |
| --- | --- | --- |
| **Flow Name** | **Content** | **Comment** |
| P | float | Drives distance |
| Pdot | float | Drives speed |
| Q | float | Drives altitude and pitch |

**End\_Req**

**[MMS\_F\_PT\_F\_FC\_INTF\_37]**

**F\_FC navigation set up**

|  |  |  |
| --- | --- | --- |
| **Flow Name** | **Content** | **Comment** |
| OperatingPoint | Altitude and Speed |  |
| OperatingMode | value in {‘SPEED’, ’ALTITUDE’, ’ENERGY’} | OPERATINGREGULOPTION |
| MissionRange | Same range as the Distance mission parameter |  |

**End\_Req**

**[MMS\_F\_PT\_F\_FC\_INTF\_38]**

**F\_FC outputs**

|  |  |  |
| --- | --- | --- |
| **Flow Name** | **Content** | **Comment** |
| EstimatedTotalMass | Altitude and Speed |  |
| CurrentRange | value in {‘SPEED’, ’ALTITUDE’, ’ENERGY’} |  |
| CurrentSpeed | Same range as the Distance mission parameter |  |
| CurrentAltitude | [-200:1:1000] (m) |  |
| PropulsionTorque | [0:0.1:100000] (N.m) | **To AV** |
| BrakingTorque | [0:0.1:100000] (N.m) | **To AV** |

F\_FC need to output some emergency landing activation

**End\_Req**

**[MMS\_F\_PT\_F\_FC\_INTF\_39]**

Similar to F\_MM’s tabulated viability functions, three association tables, one per phase (climb, cruise, descent), give the means to select the appropriate PID gain triple (Kd, Kp, Ki) applicable to AV’s current sate vector.

The three tables are named:

* ClimbGains
* CruiseGains
* DescentGains

They all have the same three indexing parameters: *Note: (viability energy tables seem to have distance as additional parameter)*

* Mass
* Altitude
* Speed

They are used to compute the interpolated (Kd, Kp, Ki) triple from the triples of the nearest neighbours to the current state.

**End\_Req**

|  | Direction | Block Source | Block Target | Comment |
| --- | --- | --- | --- | --- |
| Ports |  |  |  |  |
| **AV\_PhysicalParams** | **in** |  |  |  |
| Related Data |  |  |  |  |
| p |  | Air\_Vehicle | RF\_FC\_FlightControl |  |
| pdot |  | Air\_Vehicle | RF\_FC\_FlightControl |  |
| q |  | Air\_Vehicle | RF\_FC\_FlightControl |  |
| **OperatingPoint** | **in** |  |  |  |
| Related Data |  |  |  |  |
| OperatingAltitude |  | RF\_MM\_MissionManagement | RF\_FC\_FlightControl |  |
| OperatingSpeed |  | RF\_MM\_MissionManagement | RF\_FC\_FlightControl |  |
| OperatingOption |  | RF\_MM\_MissionManagement | RF\_FC\_FlightControl |  |
| OperatingDistance |  | RF\_MM\_MissionManagement | RF\_FC\_FlightControl |  |
| OperatingMass |  | RF\_MM\_MissionManagement | RF\_FC\_FlightControl |  |
| **MM\_Cmds** | **in** |  |  |  |
| Related Data |  |  |  |  |
| StartTakeOff |  | RF\_MM\_MissionManagement | RF\_FC\_FlightControl |  |
| StartLanding |  | RF\_MM\_MissionManagement | RF\_FC\_FlightControl |  |
| **MissionAbort** | **in** |  |  |  |
| **PropulsionTrq** | **out** |  |  |  |
| Related Data |  |  |  |  |
| FC\_PropulsionTorque |  | RF\_FC\_FlightControl | RF\_EM\_EnergyViability, Air\_Vehicle, RF\_CM\_Comm\_MMS |  |
| **BrakingTrq** | **out** |  |  |  |
| Related Data |  |  |  |  |
| FC\_BrakingTorque |  | RF\_FC\_FlightControl | RF\_EM\_EnergyViability, Air\_Vehicle, RF\_CM\_Comm\_MMS |  |
| **CurrentPoint** | **out** |  |  |  |
| Related Data |  |  |  |  |
| CurrentAltitude |  | RF\_FC\_FlightControl | RF\_MM\_MissionManagement |  |
| CurrentSpeed |  | RF\_FC\_FlightControl | RF\_MM\_MissionManagement |  |
| CurrentDistance |  | RF\_FC\_FlightControl | RF\_MM\_MissionManagement |  |

### Contract

#### Assumptions

1. The measurements of (p, pdot, q) are noiseless and unbiased,
2. The values of the operating points (speed, altitude) are safe. Since security is out of the scope of the use case, no integrity check is applied to the values received from F\_CM,
3. Every mission is viable or cancelled, at any time. *Note: Cancelled, Viable or Aborted ? Cancelled means “not started”?*

#### Guarantees

1. The regulatory behavioural properties are met (see the specific document “µXAV regulatory functional safety properties”, intentionally separated from this one[[3]](#footnote-3)),
2. AV’s body is always stable (under SWC assumption),

**[MMS\_F\_PT\_F\_FC\_GUAR\_69]**

**F\_FC keeps AV in flight safety envelope**

1. AV is kept in its flight safety envelope, i.e meets phase-dependent constraints on p, pdot, q, qdot:
   1. Climb: (qdot\_minCl < qdot < qdot\_maxCl) and (q < q\_maxCl)
   2. Cruise: (qdot\_minCr < qdot < qdot\_maxCr) and (q > q\_minCr) and (pdot < pdot\_maxCr)
   3. Descent: (qdot\_minDs < qdot < qdot\_maxDs) and (q < q\_maxDs)

The constants bounding the safety envelope are given at model level. (see also 6.3.2, clause C of F\_PT’s contract),

**End\_Req**

**[MMS\_F\_PT\_F\_FC\_GUAR\_69]**

**No simultaneous propulsion and braking**

F\_FC guarantees mutual exclusion of PropulsionTorque and BrakingTorque. When one is positive, the other is set to zero.

**End\_Req**

**[MMS\_F\_PT\_F\_FC\_GUAR\_70]**

**Mission abortion when available propulsion capacity**

In-flight mission cancellation with remaining propulsion capacity implies[[4]](#footnote-4) occurrence of safety escapes for more than EscapeTime consecutive seconds.

**End\_Req**

### Sub-functions

No sub functions.

### Functional Architecture

No functional decomposition of flight control.

### Behavior Requirements

**[MMS\_F\_PT\_F\_FC\_FUNC\_41]**

**Constants for flight control safety envelope**

|  |  |  |
| --- | --- | --- |
| **Constant Name** | **Unit** | **Value** |
| qdot\_minCl | Angle.s-1 | TBD |
| qdot\_maxCl | Angle.s-1 | TBD |
| q\_maxCl | Angle | TBD |
| qdot\_minCr | Angle.s-1 | TBD |
| qdot\_maxCr | Angle.s-1 | TBD |
| q\_minCr | Angle | TBD |
| pdot\_maxCr | Angle.s-1 | TBD |
| qdot\_minDs | Angle.s-1 | TBD |
| qdot\_maxDs | Angle.s-1 | TBD |
| q\_maxDs | Angle | TBD |

**End\_Req**

**SAFETY BREAKING**

When safety escapes are pending because of wind gusts, F\_FC strives to keep the AV’s body safe by stopping propulsion and preparing the brakes. This transition is necessary for physically effective mutual exclusion between the antagonist torque actions. Safety and/or lifetime of the drone is at stake: the engines are damaged when powered during braking action. There is a need to take into account inertia and delays on actuation lines (hydraulic and electrical, defined at increment 2).

**[MMS\_F\_PT\_F\_FC\_FUNC\_42]**

**Propulsion Braking temporal separation**

The parameter

|  |  |  |
| --- | --- | --- |
| **Name** | **Unit** | **Value** |
| CommutationDuration | s | TBD |

is defined to separate the incompatible propulsion and breaking actuation phases. The numerical value of CommutationDuration is defined at model level.

**End\_Req**

**[MMS\_F\_PT\_F\_FC\_FUNC\_43]**

**Propulsion Braking commutation**

The gain scheduling policy applies to propulsion and braking control. There is a gain table swap at each Propulsion ↔ Braking swap. *Note: That seems to indicate there are actually 6 Gain tables, not three as said earlier…*

**End\_Req**

**[MMS\_F\_PT\_F\_FC\_FUNC\_44]**

**Safety braking activation**

In case of persistency in the hazardous region more than HazardDuration consecutive seconds or in case of fast evolution to a safety escape[[5]](#footnote-5), braking is activated after propulsion has been stopped for CommutationDuration seconds.

**End\_Req**

*Note: Needs a derived software REQ here to propose an interpretation of “fast” safety escape?*

**[MMS\_F\_PT\_F\_FC\_FUNC\_45]**

Braking is stopped when AV’s dynamics re-enters the safety domain. The same CommutationDuration timeout is applied before resuming propulsion to cope with physical latencies in both ways.

**End\_Req**

**REFERENCE TRAJECTORY GENERATION**

The piecewise linear approach may need small increments of the set-point. The set-point is either an altitude value or a speed value. For landing, a distance objective is added to the zero-altitude objective. Landing must occur exactly at range completion.

The reference trajectory is the discrete evolution curve of intermediate set-points generated and added between the set-points defined by µXAV’s user.

Reference trajectory generation is in charge of splitting the amplitude of the set-point updates into smaller steps, to facilitate prevention of overshoot.

**[MMS\_F\_PT\_F\_FC\_FUNC\_46]**

**Intermediate set points**

Instead of giving the true set-point on altitude or speed to Propulsion Control, it gives half of the change amplitude, i.e. the mid-value between the current state and the set-point.

IntermediateSetPoint\_forX(t) = ½ \* (CurrentX(t)+ SetPoint\_X(t))

where X stands for Speed or Altitude.

It can be computed at each cycle. Slower rates are possible, but not too slow. When the current intermediate set-point is reached the next one is computed, and so on until the user-defined set-point is reached.

The propulsion and braking controllers use IntermediateSetPoint\_X instead of SetPoint\_X. IntermediateSetPoint\_X tends to SetPoint\_X over time (zero-like aspects to be addressed when convergence).

**End\_Req**

**[MMS\_F\_PT\_F\_FC\_FUNC\_47]**

**Reference trajectory reset**

This module is reset by any navigation parameter change.

**End\_Req**

**PROPULSION CONTROL**

In this version, the energy minimization option is not actually implemented as an optimal control algorithm (will be supplied later). When “Energy” option is active, the “Altitude” or “Speed” option is selected, depending on the respective energy cost induced by the two pre-set Altitude and Speed values. The selected option is the less energetically demanding one.

**[MMS\_F\_PT\_F\_FC\_FUNC\_48]**

**Energy is Altitude or Speed**

To determine which to select, two systems of two equations in (pdotA, qA), (pdotS, qS) are to be solved.

First:

* g/L= (pdotA)2 . cos(qA)
* Altitude = L.(1 - cos(qA))

Second :

* g/L= (pdotS)2 . cos(qS)
* Speed= pdotS.L.sin(qS)

If pdotA < pdotS, option “Altitude” shall be selected otherwise “Speed” shall be selected.

**End\_Req**

**[MMS\_F\_PT\_F\_FC\_FUNC\_49]**

**Error value for PID regulation**

The navigation option conditions the computation of variable error(t) used by the PID algorithms shall be computed as:

* While in Speed mode, error(t) = Speed(t) – OperatingPoint.Speed(t)
* While in Altitude mode, error(t) = Altitude(t) – OperatingPoint.Altitude(t)

**End\_Req**

**[MMS\_F\_PT\_F\_FC\_FUNC\_50]**

The PID controller computes PropulsionTorque(t) as a sum of three torques, each one dependent on error(t), but on different time horizons:

* instantaneous error at time t: error(t),
* cumulative past error during the last *Ti* seconds:
  + integral\_error(t)=
* error variation at time t:
  + derivative\_error(t)=

Hence the three letters P, I, D.

PropulsionTorque(t) = Kp(t)\*error(t) + Ki(t)\*integral\_error(t) + Kd(t)\*derivative\_error(t).

with Kp(t), Ki(t), and Kd(t) being the scheduled gains for current state (CurrentSpeed(t), CurrentAltitude(t)).

The coefficients Ki also depend on phase, and whether we are in propulsion or braking...

**End\_Req**

**[MMS\_F\_PT\_F\_FC\_FUNC\_51]**

**Propulsion PID coefficients determination**

Like in viability computation, CurrentSpeed(t), CurrentAltitude(t) define the current position of AV for selecting its nearest neighbors in the 2D FlightDomainMesh. The distances to the 4 nearest neighbors (d1(t), d2(t), d3(t), d4(t)) are computed, the 4 weights wi(t)=1/di(t) i = 1..4, are computed, and finally the four neighbors’ gain triplets are extracted (limit cases di(t)=0 to be properly handled).

Then, Kp(t), Ki(t), and Kd(t) are the wi(t)-weighted averages of the 4 neighbors’ corresponding coefficients.

*Note: Explanation to be continued*

**End\_Req**

**[MMS\_F\_PT\_F\_FC\_FUNC\_52]**

**Braking PID coefficients determination**

The algorithm is a simpler version of the propulsion one: the Speed/Altitude option has no longer influence. error(t) is always computed from the speed.

Braking is an emergency phase triggered to quickly pull back the AV’s dynamics in the safety region. It consists in sharp speed decrease (short duration, drop of pdot and q).

A constant speed level, named RecoverySpeed, is used as set point.

|  |  |  |
| --- | --- | --- |
| **Name** | **Unit** | **Value** |
| RecoverySpeed | m/s | TBD |

*Note: Explanation to be continued*

**End\_Req**

**[MMS\_F\_PT\_F\_FC\_FUNC\_53]**

**Air vehicle state vector**

The AV’s state vector is defined by:

* Speed(t) = L.sin(q(t)).pdot(t)
* Altitude(t) = L.(1 – cos(q(t)))
* Distance(t) =
* Mass(t) = M0 + Mpayload + Micing(t), where Micing is an unknown perturbation to be estimated, and Mpayload is given by the two rotators of the control panel.

**End\_Req**

*Note: Micing(t) is neglected in increment 1, therefore Mass(t) is constant as calculated an=t mission initialisation.*

**[MMS\_F\_PT\_F\_FC\_FUNC\_54]**

**AV mechanical parameters**

The AV’s mechanical body is dimensioned by means of the following parameters:

|  |  |  |
| --- | --- | --- |
| Name | Unit | Value |
| J0 | kg.m2 | TBD |
| L | m | TBD |
| M0 | kg | TBD |

The parameter values will be supplied with the layer 0 model and the associated simulation scenarios conformant to [3].

**End\_Req**

## Function Energy Management (F\_EM)

Energy management is in charge of estimating the remaining energy available and by estimating the dispended energy removed from initial capacity. The estimation is performed using a pdot integration computation.

*Note: A complementary energy-related function performed by Mission Management consists in estimating the viability of missions.*

### Parent Function

The parent function of F\_EM is F\_PT.

### Interface Requirements

**[MMS\_F\_PT\_F\_EM \_INTF\_55]**

**F\_EM inputs**

|  |  |  |
| --- | --- | --- |
| **Flow Name** | **Content** | **Comment** |
| pdot | Angle.s-1 | From F\_MM |
| PropulsionTorque | N.m | From F\_FC |
| BrakingTorque | N.m | From F\_FC |

**End\_Req**

**[MMS\_F\_PT\_F\_EM \_INTF\_56]**

**F\_EM outputs**

|  |  |  |
| --- | --- | --- |
| **Flow Name** | **Content** | **Comment** |
| EnergyLevel | J | Sums the remaining capacities of the two sources. |
| PrimarySource | % | Relative capacity, for CP display |
| SecondarySource | % | Relative capacity, for CP display |

**End\_Req**

|  | Direction | Block Source | Block Target | Comment |
| --- | --- | --- | --- | --- |
| Ports |  |  |  |  |
| **pdot** | **in** |  |  |  |
| Related Data |  |  |  |  |
| p1 |  | Air\_Vehicle | RF\_EM\_EnergyViability |  |
| **PropulsionTrq** | **in** |  |  |  |
| Related Data |  |  |  |  |
| FC\_PropulsionTorque |  | RF\_FC\_FlightControl | RF\_EM\_EnergyViability |  |
| **BrakingTrq** | **in** |  |  |  |
| Related Data |  |  |  |  |
| FC\_BrakingTorque |  | RF\_FC\_FlightControl | RF\_EM\_EnergyViability |  |
| **MissionAbort** | **in** |  |  |  |
| **EnergyLeft** | **out** |  |  |  |
| **Primary%Left** | **out** |  |  |  |
| **Secondary%Left** | **out** |  |  |  |

### Contract

#### Assumptions

1. The mechanical body is actuated and reacts exactly as commanded (i.e the computed digital torques are exactly converted into physical torques),
2. The mechanical body behaves exactly as sensed (i.e the digital p, pdot, q state vector coincides with the AV’s actual physical state vector, at any time).

#### Guarantees

1. Under SWC assumption the mission remains viable at any time,
2. EnergyLevel (energy left) is accurate (equates to the actual embedded energy)
3. The CP-displayed primary and secondary relative capacities are accurate.

### Sub-functions

No sub functions.

### Functional Architecture

No functional decomposition of F\_EM.

### Behavior Requirements

Parameters:

• PrimaryInitialCapacity (J)

• SecondaryInitialCapacity (J)

At t=0, start of mission:

InitialCapacity=PrimaryInitialCapacity + SecondaryInitialCapacity.

**[MMS\_F\_PT\_F\_EM\_FUNC\_57]**

**Dispended Energy estimation**

DispendedEnergy(t)= PropulsionEnergy(t) + BrakingEnergy(t).

PropulsionEnergy(t)=

BrakingEnergy(t)=

**End\_Req**

*Note: In this version (increment 1) of the specification, the dispended energy estimation is considered as part of MMS because the Electrical Propulsion System is not present. In final version, this function is performed by the EPS.*

**[MMS\_F\_PT\_F\_EM\_FUNC\_58]**

**Remaining Primary Energy left**

While (PrimarySource(t) > 0)

PrimaryEnergyLeft(t)=PrimaryInitialCapacity-DispendedEnergy(t)

PrimarySource(t)=100\*(PrimaryInitialCapacity-DispendedEnergy)/

PrimaryInitialCapacity.

**End\_Req**

**[MMS\_F\_PT\_F\_EM\_FUNC\_59]**

**Remaining Primary Energy left**

When (PrimarySource(t) <= 0)

PrimarySource(t)= 0 and Primary Source frozen

**End\_Req**

**[MMS\_F\_PT\_F\_EM\_FUNC\_60]**

**Remaining Secondary Energy left**

When (PrimarySource(t) = 0)

SecondaryEnergyLeft(t)=SecondaryInitialCapacity - DispendedEnergy(t).

SecondarySource(t)= 100\*(SecondaryInitialCapacity - DispendedEnergy)/

SecondaryInitialCapacity.

**End\_Req**

## Function Communication Management (F\_CM)

For increment 1, F\_CM has two roles:

1. It manages possible inconsistencies between GS-supplied and AV-supplied navigation parameters, *Note: No, this is F\_MM responsibility*

2. It dispatches the software inputs to the functions,

3. It collects the functions’ outputs and dispatches them to software outputs

### Parent Function

The parent function of F\_CM is F\_PT.

### Interface Requirements

**[MMS\_F\_PT\_F\_CM \_INTF\_61]**

**F\_CM inputs**

|  |  |  |
| --- | --- | --- |
| **Flow Name** | **Content** | **Comment** |
| MissionSetUp (  [Distance:float;  Speed:float;  Altitude:float]) |  | From GS |
| NavigationMode ({‘RP’,’A’}) |  | From GS |
| NavigationOption ({‘SPEED’,’ALTITUDE’,’ENERGY’}) |  | From GS |
| GO |  | From GS |
| EmergencyLanding |  | From GS |
| AV\_CP\_Switches (  [Power: Boolean;  Mode: {‘RP’,’A’};  Bay:{‘OPEN’,’CLOSED’};  START:Boolean;  Rotactor1:Integer;  Rotactor2:Integer;}]) |  | From CP |
| (p, pdot, q) |  | From AV |
|  |  |  |
|  |  |  |

**End\_Req**

**[MMS\_F\_PT\_F\_CM \_INTF\_62]**

**F\_CM outputs**

|  |  |  |
| --- | --- | --- |
| **Flow Name** | **Content** | **Comment** |
| CP\_Displays(  [READY:Boolean;  CANCELLED:Boolean;  COMPLETE:Boolean;  ABORTED:Boolean;  PSRC\_CAPACITY:Integer;  SSRC\_CAPACITY:Integer  ]) |  | To CP |
| LEDs (bitarray[20]) |  | To CP |
| MissionCompleted ([PRIMARY\_SOURCE:Integer;  SECONDARY\_SOURCE:Integer; ]) |  | To GS |
| PropulsionTorque |  | To AV |
| BrakingTorque |  | To AV |

**End\_Req**

|  | Direction | Block Source | Block Target | Comment |
| --- | --- | --- | --- | --- |
| Ports |  |  |  |  |
| GS\_Inputs | in |  |  |  |
| Related Data |  |  |  |  |
| GS\_NavControlMode |  | Ground\_Station | RF\_MM\_MissionManagement |  |
| GS\_NavParameters |  | Ground\_Station | RF\_MM\_MissionManagement |  |
| GS\_NavRegulOption |  | Ground\_Station | RF\_MM\_MissionManagement |  |
| GS\_GO |  | Ground\_Station | RF\_MM\_MissionManagement |  |
| GS\_EmergencyLanding |  | Ground\_Station | RF\_EL |  |
| CP\_Inputs | in |  |  |  |
| Related Data |  |  |  |  |
| CP\_ON\_OFF\_PushButton |  | Control\_Panel | RF\_MM\_MissionManagement |  |
| CP\_Bay\_Switch |  | Control\_Panel | RF\_MM\_MissionManagement |  |
| CP\_Rotactor1 |  | Control\_Panel | RF\_MM\_MissionManagement |  |
| CP\_Rotactor2 |  | Control\_Panel | RF\_MM\_MissionManagement |  |
| CP\_NavControlMode |  | Control\_Panel | RF\_MM\_MissionManagement |  |
| CP\_USB\_Key |  | Control\_Panel | RF\_MM\_MissionManagement |  |
| CP\_START\_PushButton |  | Control\_Panel | RF\_MM\_MissionManagement |  |
| CP\_USB\_NavRegulOption??? |  | Control\_Panel | RF\_MM\_MissionManagement |  |
| AV\_PhysicalSensors | in |  |  |  |
| Related Data |  |  |  |  |
| p |  | Air\_Vehicle | RF\_FC\_FlightControl, RF\_EM\_EnergyViability |  |
| pdot |  | Air\_Vehicle | RF\_FC\_FlightControl |  |
| q |  | Air\_Vehicle | RF\_FC\_FlightControl |  |
| PropulsionTrq | in |  |  |  |
| Related Data |  |  |  |  |
| FC\_PropulsionTorque |  | RF\_FC\_FlightControl | RF\_CM\_Comm\_MMS |  |
| BrakingTrq | in |  |  |  |
| Related Data |  |  |  |  |
| FC\_BrakingTorque |  | RF\_FC\_FlightControl | RF\_CM\_Comm\_MMS |  |
| MissionStatus | in |  |  |  |
| Related Data |  |  |  |  |
| MissionREADY |  | RF\_MM\_MissionManagement | Ground\_Station, Control\_Panel |  |
| MissionCOMPLETED |  | RF\_MM\_MissionManagement | Ground\_Station, Control\_Panel |  |
| MissionCANCELLED |  | RF\_MM\_MissionManagement | Ground\_Station, Control\_Panel |  |
| MissionABORTED |  | RF\_MM\_MissionManagement | Ground\_Station, Control\_Panel |  |
| MissionAbort | in |  |  |  |
| CP\_Switches\_to\_GS | out |  |  |  |
| CP\_Displays\_to\_GS | out |  |  |  |
| Related Data |  |  |  |  |
| MissionREADY |  | RF\_MM\_MissionManagement | Ground\_Station, Control\_Panel |  |
| MissionCOMPLETED |  | RF\_MM\_MissionManagement | Ground\_Station, Control\_Panel |  |
| MissionCANCELLED |  | RF\_MM\_MissionManagement | Ground\_Station, Control\_Panel |  |
| MissionABORTED |  | RF\_MM\_MissionManagement | Ground\_Station, Control\_Panel |  |
| CP\_Displays | out |  |  |  |
| Related Data |  |  |  |  |
| MissionREADY |  | RF\_MM\_MissionManagement | Ground\_Station, Control\_Panel |  |
| MissionCOMPLETED |  | RF\_MM\_MissionManagement | Ground\_Station, Control\_Panel |  |
| MissionCANCELLED |  | RF\_MM\_MissionManagement | Ground\_Station, Control\_Panel |  |
| MissionABORTED |  | RF\_MM\_MissionManagement | Ground\_Station, Control\_Panel |  |
| PropulsionEnergy??? | out |  |  |  |
| GS\_Setup | out |  |  |  |
| Related Data |  |  |  |  |
| GS\_NavControlMode |  | Ground\_Station | RF\_MM\_MissionManagement |  |
| GS\_NavParameters |  | Ground\_Station | RF\_MM\_MissionManagement |  |
| GS\_NavRegulOption |  | Ground\_Station | RF\_MM\_MissionManagement |  |
| GS\_GO |  | Ground\_Station | RF\_MM\_MissionManagement |  |
| GS\_EmergencyLanding |  | Ground\_Station | RF\_EL |  |
| CP\_Setup | out |  |  |  |
| Related Data |  |  |  |  |
| CP\_ON\_OFF\_PushButton |  | Control\_Panel | RF\_MM\_MissionManagement |  |
| CP\_Bay\_Switch |  | Control\_Panel | RF\_MM\_MissionManagement |  |
| CP\_Rotactor1 |  | Control\_Panel | RF\_MM\_MissionManagement |  |
| CP\_Rotactor2 |  | Control\_Panel | RF\_MM\_MissionManagement |  |
| CP\_NavControlMode |  | Control\_Panel | RF\_MM\_MissionManagement |  |
| CP\_USB\_Key |  | Control\_Panel | RF\_MM\_MissionManagement |  |
| CP\_START\_PushButton |  | Control\_Panel | RF\_MM\_MissionManagement |  |
| CP\_USB\_NavRegulOption??? |  | Control\_Panel | RF\_MM\_MissionManagement |  |
| AV\_PhysicalParams | out |  |  |  |
| Related Data |  |  |  |  |
| p |  | Air\_Vehicle | RF\_FC\_FlightControl, RF\_EM\_EnergyViability |  |
| pdot |  | Air\_Vehicle | RF\_FC\_FlightControl |  |
| q |  | Air\_Vehicle | RF\_FC\_FlightControl |  |

### Contract

#### Assumptions

1. The data-link with GS is not lossy,
2. Transfer preserves integrity of contents (no security aspects),
3. Ideal digital to mechanical conversion of torques,
4. Ideal mechanical to digital conversion of the sensed mechanical state variables (p, pdot, q),
5. All communications are continuous time signals (including the messages that will become event-driven at increment 4).

#### Guarantees

1. Outer to inner information mapping is correct and complete,
2. Inner to outer information mapping is correct and complete,
3. When A mode is set on CP, the navigation options/parameters are that of the USB-Key or mission is cancelled.

### Sub-functions

No sub functions.

### Functional Architecture

No functional decomposition of F\_CM.

### Behavior Requirements

**[MMS\_F\_PT\_F\_CM\_FUNC\_63]**

**Navigation Modes/Options/Parameters set up.** *Note: To be moved to MM*

When the two channels are active and provide conflicting values for the same entity, the following selection logic is applied:

1. In case of conflict on the navigation mode, CP prevails over GS,
2. In case of RP mode, GS prevails over CP,
3. In case of A mode, CP prevails over GS.

**End\_Req**

## Function Emergency Landing (F\_EL)

F\_EL is fully defined at increment 1. It has two roles:

• Performing the hard landings activated by F\_MM,

• Monitoring the safety escapes not diagnosed by F\_MM which lead to self-activation of hard landing

*Note: Since in this version of specification (increment 1 of muXAV) the Hydraulic Braking System does not exist emergency landing function is considered as performed by the MMS.*

Hard landing is named so because it is brake only control. It may lead to terrain contact at excessive speed with ensued drone damage.

*Note: F\_EL detects on its own flight safety envelope escapes in case F\_MM malfunctions or is lost. The flight safety envelopes detection escapes detection uses the same algorithm as the one used in F\_MM.*

For the braking part, F\_EL is essentially a PID controller without gain scheduling.

### Parent Function

F\_EL parent function is F\_MMS top-level.

### Interface Requirements

**[MMS\_F\_EL \_INTF\_64]**

**F\_EL inputs**

|  |  |  |
| --- | --- | --- |
| **Flow Name** | **Content** | **Comment** |
| p | float | Drives distance |
| pdot | float | Drives speed |
| q | float | Drives altitude and pitch |
| EmergencyLanding | event | Trigger from F\_PT.F\_MM |

**End\_Req**

**[MMS\_F\_EL \_INTF\_65]**

**F\_EL outputs**

|  |  |  |
| --- | --- | --- |
| **Flow Name** | **Content** | **Comment** |
| BrakingTorque | [-10e-6:0.0001:+10e-6] (N.m) |  |
| MissionAbort | event | To F\_PT.F\_MM |

**End\_Req**

F\_EL is not aware of any remaining Electrical energy and never attempts soft landing, only hard landing with hydraulic energy is performed, only BrakingTorque command is sent.

|  | Direction | Block Source | Block Target | Comment |
| --- | --- | --- | --- | --- |
| Ports |  |  |  |  |
| **EmergencyLanding** | **in** |  |  |  |
| Related Data |  |  |  |  |
| MM\_EmergencyLanding |  | RF\_MM\_MissionManagement | RF\_EL |  |
| GS\_EmergencyLanding |  | Ground\_Station | RF\_EL |  |
| **AV\_PhysicalSensors** | **in** |  |  |  |
| Related Data |  |  |  |  |
| p |  | Air\_Vehicle | RF\_EL |  |
| pdot |  | Air\_Vehicle | RF\_EL |  |
| q |  | Air\_Vehicle | RF\_EL |  |
| **MissionAbort** | **out** |  |  |  |
| **BrakingTorque** | **out** |  |  |  |
| Related Data |  |  |  |  |
| EL\_BrakingTorque |  | RF\_EL | Air\_Vehicle |  |

### Contract

#### Assumptions

No assumption. Since F\_EL is the ultimate control process to avoid crash, its robustness has to be maximal; it has to be operative in any condition.

#### Guarantees

1. Under assumption of continued braking power and SWC, there is no damage to the AV (defined as |qdot| < MaxTouchSpeed when q=0; MaxTouchSpeed=0.5 m/s).
2. Under AWC assumption, damage may occur in at most 25% of the cases (95% confidence level).

### Sub-functions

No sub functions.

### Functional Architecture

No functional decomposition of F\_EL.

### Behavior Requirements

**[MMS\_F\_EL\_FUNC\_66]**

A linear reference trajectory of q and qdot are defined from q and qdot at F\_EL activation time, down-to q=0, qdot=0 at grounding time. These trajectories are defined with safety margins wt. descent safety envelope. (Safety envelope max gains xxxDs parameters).

**End\_Req**

1. i.e an initiative of the development team to enhance safety in addition to the safety requirements issued by the safety process. [↑](#footnote-ref-1)
2. Surrogate models are common in engineering but may be unfamiliar in system and software engineering. They approximate the results of time-consuming computation (detailed PDE physical models, results of design of experiments, etc.). They use tables of parameters and interpolation functions in suitable bases (splines, radial basis functions, polynomials etc.). [↑](#footnote-ref-2)
3. Designers are assumed to consult FAR 2x / CS 2x regulation, and to derive the behavioral or airframe requirements that ensure conformance to it. There is a form of implicitly inherited regulatory contract. [↑](#footnote-ref-3)
4. needs the MS flight data recorder option, [↑](#footnote-ref-4)
5. Intentionally left unformal. Example of a safety critical system-software transition issue. [↑](#footnote-ref-5)