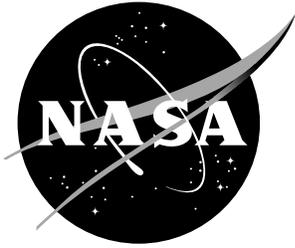


NASA/CR-2003-212426



# Flight Guidance System Requirements Specification

*Steven P. Miller, Alan C. Tribble, Timothy M. Carlson, and Eric J. Danielson  
Rockwell Collins, Cedar Rapids, Iowa*

---

June 2003

## The NASA STI Program Office . . . in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA Scientific and Technical Information (STI) Program Office plays a key part in helping NASA maintain this important role.

The NASA STI Program Office is operated by Langley Research Center, the lead center for NASA's scientific and technical information. The NASA STI Program Office provides access to the NASA STI Database, the largest collection of aeronautical and space science STI in the world. The Program Office is also NASA's institutional mechanism for disseminating the results of its research and development activities. These results are published by NASA in the NASA STI Report Series, which includes the following report types:

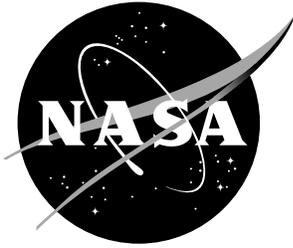
- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counterpart of peer-reviewed formal professional papers, but having less stringent limitations on manuscript length and extent of graphic presentations.
- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.
- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.
- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or co-sponsored by NASA.
- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.
- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services that complement the STI Program Office's diverse offerings include creating custom thesauri, building customized databases, organizing and publishing research results ... even providing videos.

For more information about the NASA STI Program Office, see the following:

- Access the NASA STI Program Home Page at <http://www.sti.nasa.gov>
- E-mail your question via the Internet to [help@sti.nasa.gov](mailto:help@sti.nasa.gov)
- Fax your question to the NASA STI Help Desk at (301) 621-0134
- Phone the NASA STI Help Desk at (301) 621-0390
- Write to:  
NASA STI Help Desk  
NASA Center for AeroSpace Information  
7121 Standard Drive  
Hanover, MD 21076-1320

NASA/CR-2003-212426



# Flight Guidance System Requirements Specification

*Steven P. Miller, Alan C. Tribble, Timothy M. Carlson, and Eric J. Danielson  
Rockwell Collins, Cedar Rapids, Iowa*

National Aeronautics and  
Space Administration

Langley Research Center  
Hampton, Virginia 23681-2199

Prepared for Langley Research Center  
under Cooperative Agreement NCC1-01001

---

June 2003

---

Available from:

NASA Center for AeroSpace Information (CASI)  
7121 Standard Drive  
Hanover, MD 21076-1320  
(301) 621-0390

National Technical Information Service (NTIS)  
5285 Port Royal Road  
Springfield, VA 22161-2171  
(703) 605-6000

## **Abstract**

This report describes a requirements specification written in the RSML<sup>-e</sup> language for the mode logic of a Flight Guidance System of a typical regional jet aircraft. This model was created as one of the first steps in five-year project sponsored by the NASA Langley Research Center, Rockwell Collins Inc., and the Critical Systems Research Group of the University of Minnesota to develop new methods and tools to improve the safety of avionics designs. This model will be used to demonstrate the application of a variety of methods and techniques, including safety analysis of system and subsystem requirements, verification of key properties using theorem provers and model checkers, identification of potential sources mode confusion in system designs, partitioning of applications based on the criticality of system hazards, and autogeneration of avionics quality code. While this model is representative of the mode logic of a typical regional jet aircraft, it does not describe an actual or planned product. Several aspects of a full Flight Guidance System, such as recovery from failed sensors, have been omitted, and no claims are made regarding the accuracy or completeness of this specification.



# Contents

- 1 Introduction** **5**
- 1.1 How Requirements Analysis Relates to Aircraft Safety . . . . . 5
- 1.2 Overview of a Flight Guidance System . . . . . 9
- 1.3 Fundamentals of the Mode Logic . . . . . 12
  
- 2 The FGS Requirements Specification** **15**
- 2.1 Overview of the FGS Specification . . . . . 16
- 2.2 Prioritization of Events . . . . . 20
- 2.3 Basic Definitions . . . . . 23
- 2.4 Flight Director (FD) . . . . . 25
- 2.5 Pilot Flying (PF) . . . . . 29
- 2.6 Independent Mode . . . . . 30
- 2.7 Flight Modes . . . . . 32
  - 2.7.1 Lateral Modes . . . . . 35
    - 2.7.1.1 Roll Hold (ROLL) Mode . . . . . 37
    - 2.7.1.2 Heading Select (HDG) Mode . . . . . 40
    - 2.7.1.3 Navigation (NAV) Mode . . . . . 44
    - 2.7.1.4 Lateral Approach (LAPPR) Mode . . . . . 50
    - 2.7.1.5 Lateral Go Around (LGA) Mode . . . . . 56
  - 2.7.2 Vertical Modes . . . . . 60
    - 2.7.2.1 Pitch Hold (PITCH) Mode . . . . . 62
    - 2.7.2.2 Vertical Speed (VS) Mode . . . . . 65
    - 2.7.2.3 Flight Level Change (FLC) Mode . . . . . 69
    - 2.7.2.4 Altitude Hold (ALT) Mode . . . . . 73
    - 2.7.2.5 Altitude Select (ALTSEL) Mode . . . . . 77
    - 2.7.2.6 Vertical Approach (VAPPR) Mode . . . . . 85
    - 2.7.2.7 Vertical Go Around (VGA) Mode . . . . . 91
- 2.8 Flight Control Laws . . . . . 95
- 2.9 Flight Control Panel (FCP) . . . . . 103
- 2.10 Navigation Sources . . . . . 127
- 2.11 Air Data System . . . . . 128
- 2.12 References . . . . . 129

2.13 Autopilot (AP) . . . . .	130
2.14 Offside FGS . . . . .	133
2.15 FGS Inputs . . . . .	152
2.16 FGS Outputs . . . . .	156
References . . . . .	161
Index . . . . .	163

# Chapter 1

## Introduction

This report describes a requirements specification written in the RSML<sup>-e</sup> [8, 6] language for the mode logic of a Flight Guidance System of a typical regional jet aircraft. This model was created as one of the first steps in a project sponsored by the NASA Langley Research Center, Rockwell Collins, Inc., and the Critical Systems Research Group of the University of Minnesota to develop new methods and tools to improve the safety of avionics designs. This model will be used to demonstrate the application of a variety of methods and techniques, including:

- Safety analysis of system and subsystem requirements
- Verification of key properties using theorem provers and model checkers
- Identification of potential sources mode confusion in system designs
- Partitioning of applications based on the criticality of system hazards
- Autogeneration of avionics quality code

While this model is representative of the mode logic of a typical regional jet aircraft, it does not describe an actual or planned product. Several aspects of a full Flight Guidance System, such as recovery from failed sensors, have been omitted, and no claims are made regarding the accuracy or completeness of this specification.

The report is organized as follows. In this chapter, Section 1.1 discusses why the modeling and analysis of requirements is important to aircraft safety, Section 1.2 provides an overview of a modern Flight Guidance System, and Section 1.3 discusses the the fundamentals of the mode logic found in the FGS. Chapter 2 deals with the requirements specification itself. Section 2.1 describes how the specification is organized. The remaining sections of Chapter 2 are the specification itself.

### 1.1 How Requirements Analysis Relates to Aircraft Safety

Aircraft safety has improved steadily over the last few decades and much of this improvement can be attributed to the introduction of advanced automation in the cockpit. The predicted

Accident Category	Fatalities	Accidents
<b><i>Controlled Flight Into Terrain</i></b>	2111	28
<b><i>Loss of Control in Flight</i></b>	2011	29
In Flight Fire	600	3
<b><i>Midair Collision</i></b>	506	2
Fuel Tank Explosion	238	2
<b><i>Landing</i></b>	~200	14
<b><i>Takeoff</i></b>	~144	3
Ice/Snow	~60	3
Fuel Exhaustion	~40	5
<b><i>Wind Shear</i></b>	~30	2
<b><i>Runway Incursion</i></b>	~10	4
Misc. Fatality	~10	3
<b><i>On Ground</i></b>	~10	3
Refused Takeoff	~10	1
<b><i>Turbulence</i></b>	~10	3
Unknown	482	7
<b>Total</b>	<b>6465</b>	<b>112</b>

Table 1.1: Fatal Accidents - Worldwide Commercial Jet Fleet, 1990-1999 - From [14]

ten-fold increase in air traffic by 2016 will force much greater reliance on automated systems. New technologies, such as: Communication, Navigation, and Surveillance for Air Traffic Management (CNS/ATM); Global Air Traffic Management (GATM); Global Positioning System (GPS); and fly-by-wire, will be introduced to meet enhanced safety and performance goals. Table 1.1 classifies the fatal accidents of the worldwide commercial jet fleet for the period of 1990 to 1999. Categories in which automated systems can play a role in reducing the accident rate still further are emphasized. These account for almost 80 percent of the accidents and fatalities.

However, the growing complexity and integration issues associated with these advanced technologies also increase the potential for errors that could have a direct impact on safety. As systems become more tightly integrated, even the engineers that design them will find it difficult to anticipate every possible interaction. Unfortunately, it is precisely these interactions that will have the greatest impact on safety:

*Today more accidents result from dangerous design characteristics and interactions among components.[10]*

*Complex and highly integrated avionics present greater risk for development error. With non-traditional human-machine interfaces, there is also the potential for operational flight crew errors. Moreover, integration of systems may result in a greater*

*likelihood of undesirable and unintended effects.[2]*

In spite of the increased safety concerns, manufacturers are being asked to design, develop and validate the systems of the future under the budgets of the past. It is imperative that new methods to improve system safety be developed, and that these methods work in concert with the goals of reducing cost and cycle time.

One of the best ways to improve product quality while reducing costs is to develop a complete, consistent, and well organized set of requirement at the start of the product life cycle. As stated by Fred Brooks:

*The hardest single part of building a software system is deciding precisely what to build. No other part of the conceptual work is as difficult as establishing the detailed technical requirements... No other part of the work so cripples the resulting system if done wrong. No other part is as difficult to rectify later.[4]*

The majority of software development errors stem from logic errors made during requirements analysis. Most of these errors are not found until the later phases of a project. The amount of rework that has to be done to fix a requirements error, and the cost of doing so, grows dramatically the later it is detected.[1] In one well-known study, it was found that it costs ten times as much to correct a requirements error during unit testing as during requirements analysis. Correcting a requirements error after a product had been deployed increased the cost by 100 to 200 times.[3] Moreover, requirements errors are often the most serious errors. Investigators focusing on safety-critical systems have found that requirements errors are more likely to affect the safety of an embedded system than errors introduced during design or implementation.[12]

One of the most successful approaches to finding requirements errors is to create a precise model of the system's externally visible behavior that can be executed. Several notations have been developed over the last few years for creating a model of the system's functional behavior that is both readable and mathematically precise. Among the better known are SCR [7], RSML [8, 6], SpecTRM [10], and Statecharts [5]. Creating models based on these notations have been shown to find a wealth of errors in textual specifications [6, 13]. Moreover, such models can be connected to a mock-up of the user interface and executed with the customer in the same way as a simulation. In the best approaches, the underlying notation has been carefully designed to support automated analyses. These make possible a variety of consistency and completeness checks that find many errors, as well as the ability to check for properties specific to the application being modeled. Finally, the requirements model itself becomes a detailed statement of the desired behavior. This enhances design and testing, and makes it far more feasible to outsource the software development.

In short, creation of a precise model of a system's behavior not only finds errors early in the life-cycle when they can be most economically addressed, it enables a variety of downstream activities traditionally associated with the quality of the system, including design, coding, and verification. Particularly relevant to this report, it lays the foundation for improving system safety through analysis:

*Every hazard analysis requires some type of model of the system, which may range from a fuzzy idea in the analyst's mind to a complex and carefully specified mathematical*

*model. The model may range from a high-level abstraction to a low-level and detailed prototype. Nevertheless, information about the system must exist in some form, and that constitutes the system model upon which the analysis is performed.[11]*

The model of a Flight Guidance System described in this report will be used for the demonstration of a wide variety of techniques developed to improve the quality and safety of embedded systems, thereby reducing the accident rate shown in Table 1.1.

## 1.2 Overview of a Flight Guidance System

A Flight Guidance System (FGS) is a component of the overall Flight Control System (FCS). It compares the measured state of an aircraft (position, speed, and attitude) to the desired state and generates pitch and roll guidance commands to minimize the difference between the measured and desired state. A simplified overview of an FCS that emphasizes the role of the FGS is shown in Figure 1.1.

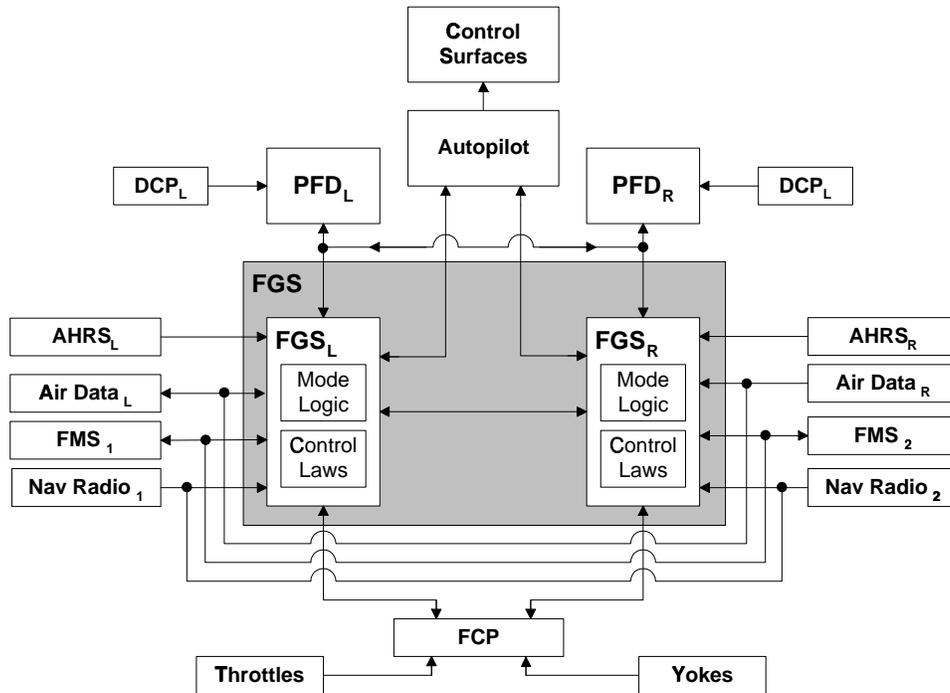


Figure 1.1: Flight Control System Overview

As shown in Figure 1.1, the FGS subsystem accepts input about the aircraft's state from the Attitude Heading Reference System (AHRS), Air Data System (ADS), Flight Management System (FMS), and Navigation Radios. Using this information, it computes pitch and roll guidance commands that are provided to the Autopilot (AP). When engaged, the Autopilot translates these commands into movement of the aircraft's control surfaces necessary to achieve the commanded changes about the lateral and vertical axes.

The flight crew interacts with the FGS primarily through the Flight Control Panel (FCP), shown in more detail in Figure 1.2. The FCP includes switches for turning the Flight Director (FD) on and off, switches for selecting the different flight modes such as vertical speed (VS), lateral navigation (NAV), heading select (HDG), altitude hold (ALT), and approach (APPR), the Vertical Speed/Pitch Wheel, and the autopilot disconnect bar. The FCP also supplies feedback to the crew, indicating selected modes by lighting lamps on either side of a selected mode's button. Figure 1.2 depicts a configuration in which Heading Select (HDG) mode is selected.

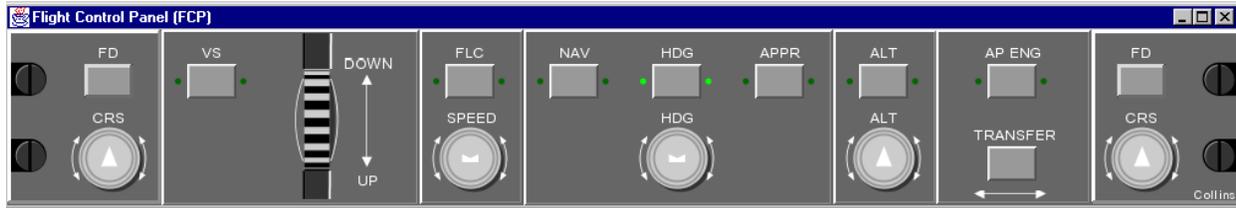


Figure 1.2: Flight Control Panel

A few key controls, such as the Go Around button and the Autopilot Disengage switch, are provided on the control yokes and throttles and routed through the FCP to the FGS. Navigation sources are selected through the Display Control Panel (DCP), with the selected navigation source routed through the PFD to the FGS.

The FGS has two physical sides, or channels, one on the left side and one on the right side of the aircraft (see Figure 1.1). These provide redundant implementations that communicate with each other over a cross-channel bus. Each channel of the FGS can be further broken down into the mode logic and the flight control laws. The flight control laws accept information about the aircraft's current and desired state and compute the pitch and roll guidance commands. The mode logic determines which lateral and vertical modes of operation are active and armed at any given time. These in turn determine which flight control laws are active and armed. These are annunciated, or displayed, on the Primary Flight Displays (PFD) along with a graphical depiction of the flight guidance commands generated by the FGS.

A simplified image of a Primary Flight Display (PFD) is shown in Figure 1.3. The PFDs display essential information about the aircraft, such as airspeed, vertical speed, attitude, the horizon, and heading. The active lateral and vertical modes are displayed (annunciated) at the top of the display. The annunciations in Figure 1.3 indicate that the current active lateral mode is Heading Select (HDG), the active vertical mode is Pitch (PTCH), and that Altitude Select (ALTS) mode is armed.

The large sphere in the center of the PFD is the sky/groundball. The horizontal line across its middle is the artificial horizon. The current pitch and roll of the aircraft is indicated by a white wedge  $\wedge$  representing the aircraft in the middle of the sky/ground ball. Figure 1.3 depicts an aircraft with zero degrees of roll and pitched up approximately five degrees.

The graphical presentation of the pitch and roll guidance commands on the PFD are referred to as the Flight Director (FD).<sup>1</sup> The pitch and roll guidance commands are shown as a magnet wedge  $\wedge$  in the sky/ground ball. When the autopilot is not engaged, these are interpreted as guidance to the pilot. When the autopilot is engaged, these indicate the direction the aircraft is being steered by the autopilot. Figure 1.3 depicts an aircraft in which the autopilot is not engaged and the Flight Director is commanding the pilot to pitch up and roll to the right.

<sup>1</sup>The term Flight Director is also commonly used to refer to the logic that computes the pitch and roll guidance commands.



Figure 1.3: Primary Flight Display

### 1.3 Fundamentals of the Mode Logic

A *mode* is defined by Leveson as a “mutually exclusive set of system behaviors” [9]. Specifically as it relates to an FGS, advisory circular AC/ACJ 25.1329 defines a mode as “a system configuration that corresponds to a single (or set of) FGS behavior(s)” [2]. The primary modes of interest in an FGS are the lateral and vertical modes. The lateral modes control the behavior of the aircraft about the longitudinal, or roll, axis, while the vertical modes control the behavior of the aircraft about the vertical, or pitch, axis. In addition, there are a number of auxillary modes, such as half-bank mode, that control other aspects of the aircraft’s behavior. Examples of FGS modes include Heading Hold (HDG) which holds the aircraft to a selected heading and Vertical Speed (VS) mode which holds the aircraft to selected vertical speed.

A mode is said to be *selected* if it has been manually requested by the flight crew or if it has been automatically requested by a subsystem such as the FMS. The simplest modes have only two states, *cleared* and *selected*, as shown in Figure 1.4. Such a mode becomes active immediately upon selection with it’s associated flight control law providing guidance commands to the flight director and, if engaged, the autopilot. When cleared, a mode’s associated flight control law is non-operational, i.e., it does not generate any outputs.

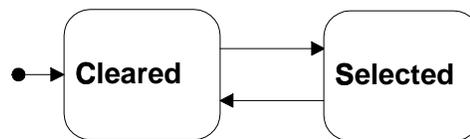


Figure 1.4: A Simple Mode

Some modes can be *armed* to become active when a criterion is met, such as the acquisition of a navigation source or proximity to a target reference such as a desired altitude. Such modes have three states as shown in Figure 1.5. The two states *armed* and *active* are substates of the *selected* state, i.e., when the mode is armed or active, it is also said to be selected. While in the armed state, the mode’s flight control law is not generating guidance commands for the flight director or the autopilot, but it may be accepting inputs, accumulating state information, and helping to determine if the criterion for becoming active is met. Once the criterion is met, the mode transitions to the active state and its flight control law begins generating guidance for the flight director and autopilot. Note that the only way to exit the active state is to deselect the mode, i.e., it is not usually possible to revert directly from the active state to the armed state.

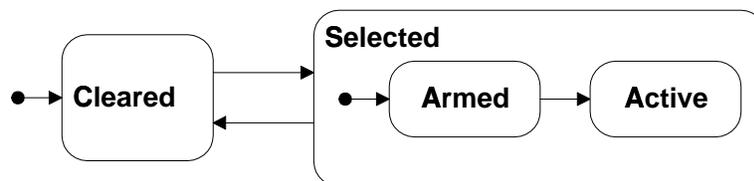


Figure 1.5: A Mode with Armed and Active Substates

Some modes also distinguish between capturing and tracking of the target reference or navigation source. Such a mode is shown in Figure 1.6. Once in the active state, such a mode’s flight control law first *captures* the target by maneuvering the aircraft to align it with the navigation source or reference. Once correctly aligned, the mode transitions to the *tracking* state in which it holds the aircraft on the target. Both the *capture* and *track* states are substates of the *active* state and the mode’s flight control law is active in both states, i.e., generating guidance commands for the flight director and autopilot. Note that the only way to exit the active, track, or capture states is to deselect the mode, i.e., it is not possible to revert directly from the track state to the capture state or from the active state to the armed state.

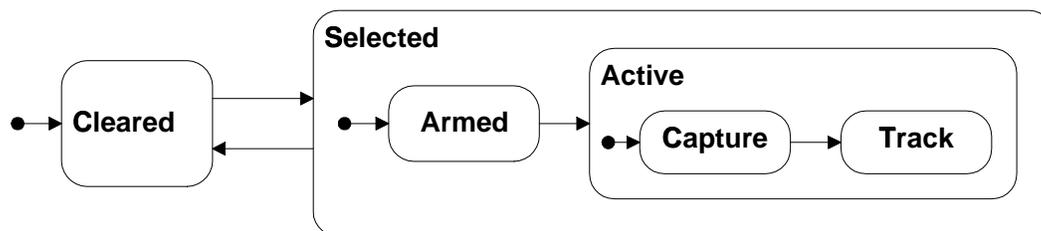


Figure 1.6: A Mode with Capture and Track Substates

The *mode logic* consists of all the available modes and the rules for transitioning between them. Figure 1.7 provides an overview of the modes described in this specification. Traditionally, aircraft modes are associated with a flight control law that determines the guidance provided to the flight director or autopilot. For example, in Figure 1.7, there are lateral modes of Roll Hold, Heading Hold, Navigation, Lateral Approach, and Lateral Go Around. These control the guidance about the longitudinal, or roll, axis. Guidance about the vertical, or pitch, axis is controlled by the vertical modes of Pitch, Vertical Speed, Altitude Hold, Altitude Select, Vertical Approach, and Vertical Go Around. Each of these are associated with one or more control laws.

Also shown in Figure 1.7 are several auxillary modes such as Pilot Flying, Independent Mode, FD, and AP, that describe the status of key state variables. While these have an important affect upon the behavior of the aircraft, they are not directly associated with specific flight control laws, and are not usually viewed as “modes” by system designers in the same way the lateral and vertical modes are.

In order to provide effective guidance of the aircraft, these modes are tightly synchronized so that only a small portion of their total state space is actually reachable. For example, to ensure that meaningful guidance is provided to the flight director and autopilot, only one lateral and one vertical mode can be active at any time. For the same reason, if the autopilot is engaged or the flight director is turned on, at least one lateral and one vertical mode must be active. Other constraints enforce sequencing of modes that are dictated by the characteristics of the aircraft and the airspace. For example, vertical approach mode is not usually allowed to become active until lateral approach mode has become active to ensure that the aircraft is horizontally centered on the localizer before tracking the glideslope. These constraints are clearly important to safe flight, and can become quite complex. The mode logic is responsible for enforcing all these constraints.

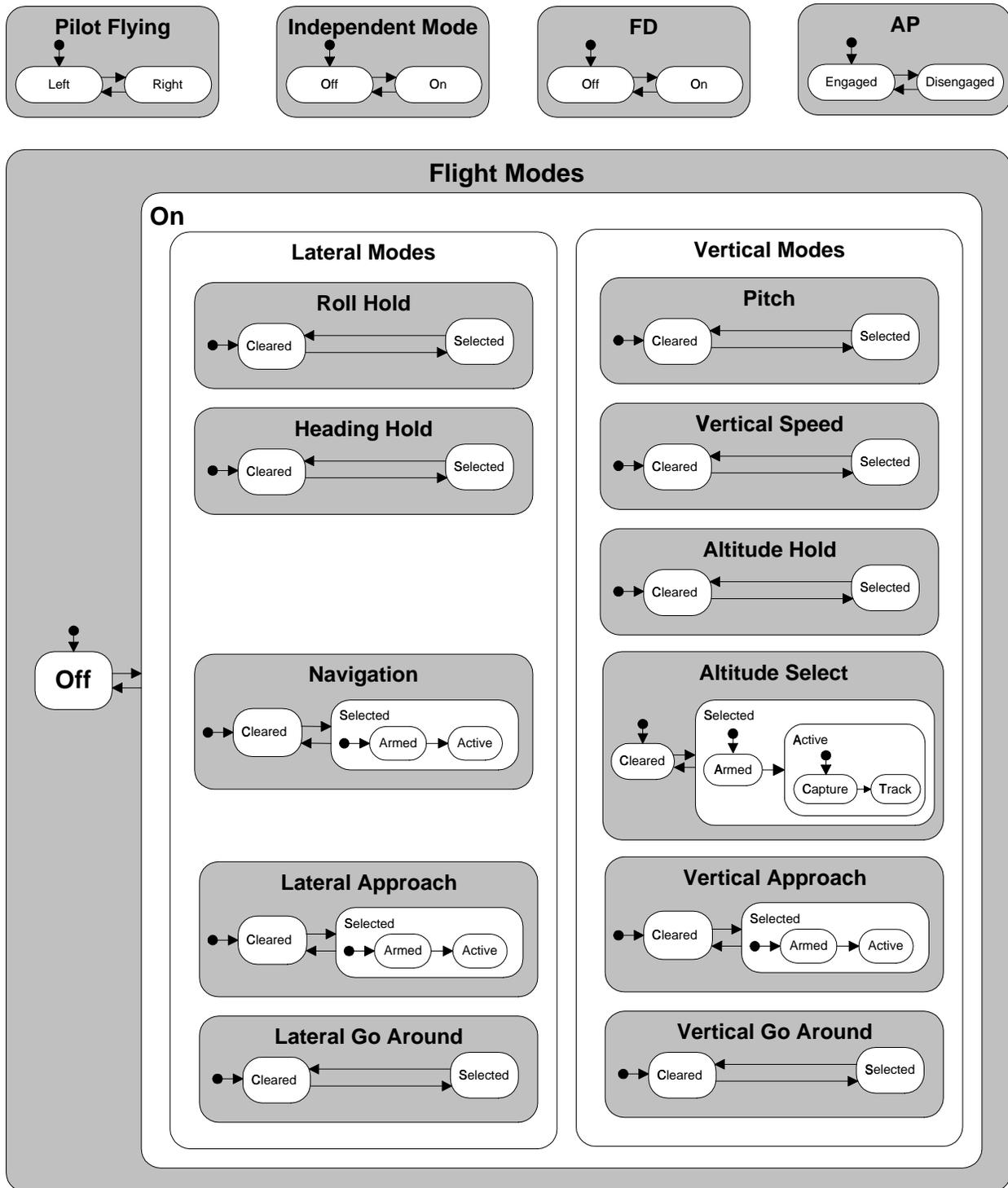


Figure 1.7: FGS Mode Logic

## Chapter 2

# The FGS Requirements Specification

A requirements specification must meet the needs of a diverse set of stakeholders, including but not limited to the customer, end users, program management, systems engineers, software engineers, hardware engineers, test engineers, and regulatory agencies. It must be clear enough that end users can understand it, yet complete and precise enough that the engineers can implement the system correctly and develop a comprehensive set of test cases. It must be robust in the face of change so that small changes only require small efforts. Ideally, the specification should support the development of a family of products, so that it can easily be reused to develop an entire line of similar products.

The FGS requirements specification described in this document is organized to meet these goals. Its structure is hierarchial, allowing the reader to start with the entire subsystem, then examine individual components in greater detail. To enhance readability, it makes heavy use of the tabular formats developed in RSML<sup>-e</sup> to make the complex logic of the Traffic Collision Avoidance System (TCAS) accessible to pilots, engineers, and regulatory authorities.[6]

Some of these goals, particularly those related to robustness in the face of change and support for product families, are best achieved by organizing the specification into pieces that are logically related and likely to change together and by defining interfaces between these components that are unlikely to change. The current version of RSML<sup>-e</sup> does not directly support these notions, though extensions are planned for future versions that will.

However, this specification has been written as though these features were available. Functions that are logically related are grouped together into components, with details that are likely to change encapsulated within these components. Rather than referring to these directly in the other parts of the specification, interfaces (e.g., functions or macros) are defined that should remain stable over time for use in the rest of the specification. Many of the architectural decisions are based on an extensive commonality analysis of several Flight Guidance Systems to determine which features change and which remain stable over a period of time. To facilitate such anticipated changes, the architecture implicitly defines a framework in which common patterns and interfaces are used across similar components.

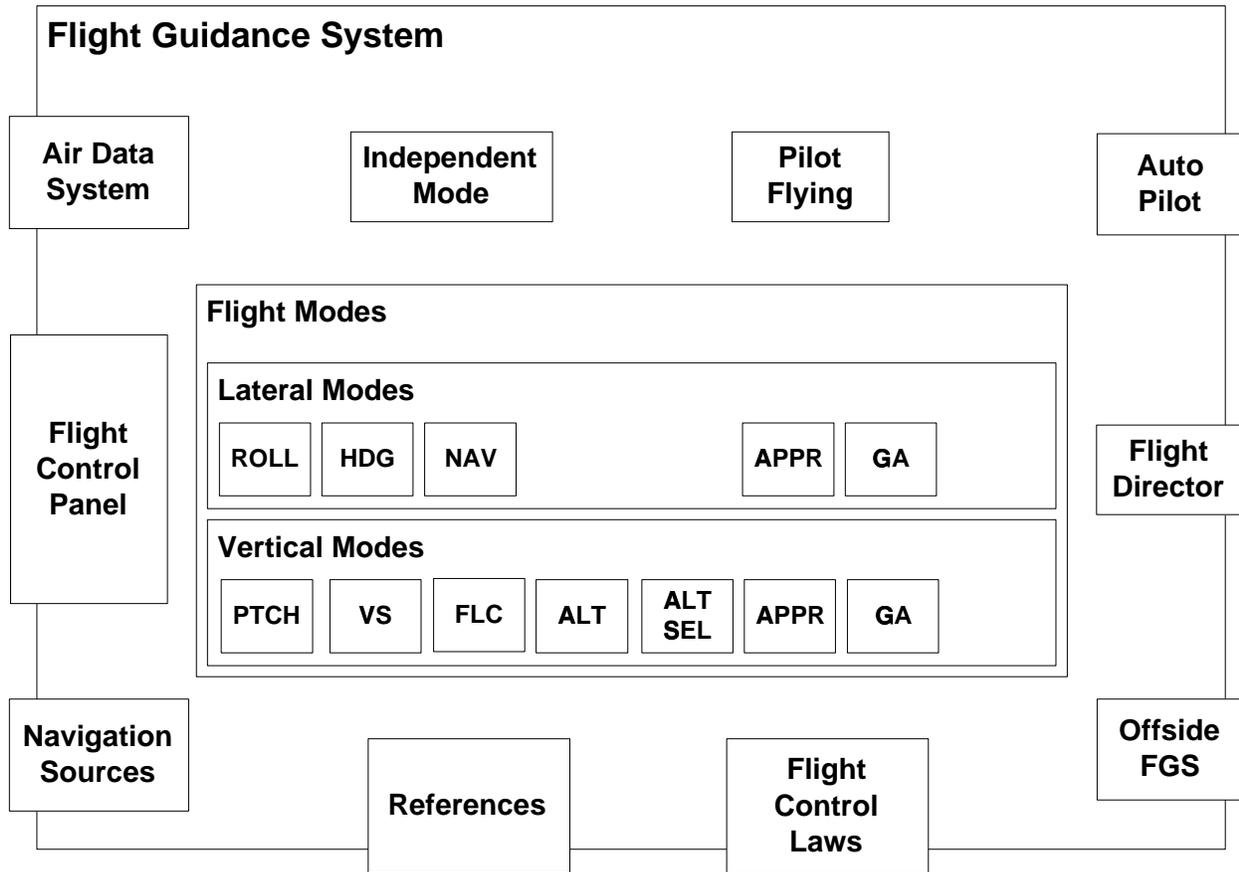


Figure 2.1: Flight Guidance Specification Overview

## 2.1 Overview of the FGS Specification

The top level structure of the FGS is shown in Figure 2.1. The enclosing box labeled *Flight Guidance System* represents the full specification. The components located on the edge of the Flight Guidance System are *boundary* components that define the system's view of its interfaces with the outside world. These do not define the behavior of the actual external subsystem. Rather, they define the behavior of that subsystem that the FGS must be aware of. For example, the component labeled *Flight Control Panel* (FCP) only defines what information is received by and sent to the Flight Control Panel and does not define the full behavior of that actual device.

Each component may define new events or values that are used by other components. For example, the FCP mentioned above defines and exports events for each switch that can be pressed on the panel. In fact, since several of these events can occur at the same time, the FCP component defines a prioritization of these events so that only the highest priority events are seen by the rest of the specification (see Section 2.2).

The heart of the specification is the definition of the mode logic contained in the *Modes*

component. This is further broken down into the *Lateral Modes* and *Vertical Modes*, which are in turn broken down into individual modes. The lateral and vertical modes all present standard interfaces to their parent modules, making it straightforward to add, remove, or substitute modes, a common source of variation in a FGS. In addition, many of the modes share a common behavior that can be specified once, then tailored to define the behavior of each individual mode.

The following sections provide an overview of each individual component. Detailed specifications of the components can be found in the remainder of this chapter.

**Flight Control Panel** (page 103) The Flight Control Panel (FCP) is the primary device with which the flight crew interacts with the FGS. This component defines the FGS’s view of the FCP. This consists of the definition of the monitored variables representing the switches and dials of the FCP and the controlled variables representing the lamps displayed on the FCP. A few additional inputs, such as the SYNC and AP Disengage switches on the control yokes and the GA switches on the throttles, are routed through the FCP and are defined here. This component also defines FCP specific events, such as the pressing of a switch, used in the rest of the specification. These events are prioritized so that if multiple events occur at the same time, only the most significant events are seen by the other parts of the specification.

**Flight Director** (page 25) The Flight Director (FD) displays the pitch and roll guidance commands to the pilot and copilot on the Primary Flight Display. This component defines when the Flight Director guidance cues are turned on and off.

**Independent Mode** (page 30) Normally, the FGS operates in dependent mode in which the pilot flying (PF) side of the FGS provides the mode annunciations and FD guidance commands to both PFDs. While in dependent mode, the pilot not flying (PNF) side of the FGS synchronizes its modes to the pilot flying side. Under final approach conditions and during takeoff or go around modes, the FGS operates in independent mode in which each side operates independently of the other side and provides mode annunciations and FD guidance commands to its own PFD. This component defines when the FGS enters independent mode.

**Pilot Flying** (page 29) The transfer switch on the FCP is used to determine which FGS provides mode annunciations and guidance commands to the PFDs while in dependent mode. This component defines which side is the Pilot Flying side. It also defines the notion of the “active” side of the FGS. A side of the FGS is said to be “active” if it is providing mode annunciations and guidance commands to its PFD.

**Flight Control Laws** (page 95) The Flight Control Laws generate the pitch and roll guidance commands provided to the Flight Director and the Autopilot. Different Flight Control Laws are armed or activated based on the current modes of the FGS. They also provide inputs to the mode logic that determine when transitions are made from the armed to the active state. While the Flight Control Laws are not specified in this model, this component defines the interface between them and the mode logic and defines the events provided by

the Flight Control Laws that the mode logic depends upon. As with events provided by the Flight Control Panel, some of these are prioritized so that if multiple events occur at the same time, only the most significant events are seen by the other parts of the specification.

**Navigation Sources** (page 127) The FGS receives navigation information from several sources, including VHF Omni-Directional Range (VOR) for lateral navigation, Localizer (LOC) for precision lateral approach, and Glideslope (GS) for precision vertical approach. This component defines the FGS's view of its navigation sources. This includes definition of the types of sources available, the currently selected source, and the information available from that source.

**Air Data System** (page 128) The Air Data System provides information about the aircraft state sensed from the surrounding air mass such as pressure altitude and indicated airspeed. This component defines the FGS's view of the Air Data System.

**References** (page 129)

References refer to target values used by several of the modes, such as the Preselected Altitude. These may be set by the flight crew through the Flight Control Panel, or set to the current aircraft value on entry to a mode or a synchronization request. The references may be physically maintained by a variety of systems. This component defines the FGS's view of the references. Details of how these values are obtained from other systems are encapsulated within this component.

**Autopilot** (page 130) The Autopilot (AP) commands the movement the control surfaces based on the pitch and roll commands generated by the FGS. This component defines the FGS's view of the AP.

**Offside FGS** (page 133) The FGS has two physical sides, or channels, on the left and right sides of the aircraft. These provide redundant implementations that communicate with each other over a cross-channel bus. This component defines this side of the FGS's view of the other side.

**Flight Modes** (page 32) The flight modes determine which modes of operation of the FGS are active and armed at any given moment. These in term determine which flight control laws are generating the commands directing the aircraft along the lateral (roll) and vertical (pitch) axes. This component encapsulates the definitions of the lateral and vertical modes and defines how they are synchronized.

**Lateral Modes** (page 35) The lateral modes select the control laws generating commands directing the aircraft along the lateral, or roll, axis. This component describes the specific lateral modes present in this aircraft and defines how they are synchronized.

**Roll Hold (ROLL) Mode** (page 37) In Roll Hold mode the FGS generates guidance commands to hold the aircraft at a fixed bank angle. Roll Hold mode is the basic

lateral mode and is always active when the modes are displayed and no other lateral mode is active.

**Heading Select (HDG) Mode** (page 40) In Heading Select mode, the FGS provides guidance commands to track the *Selected Heading* displayed on the PFD.

**Lateral Navigation (NAV) Mode** (page 40) In Lateral Navigation mode, the FGS provides guidance commands to acquire and track lateral guidance for en route navigation and non-precision approaches. Lateral Navigation is an arming mode, i.e., after being selected, it must remain armed for a period of time before it can become active.

**Lateral Approach (LAPPR) Mode** (page 50) In Lateral Approach mode, the FGS provides guidance commands to acquire and track lateral guidance for precision and non-precision approaches. Lateral Approach is an arming mode, i.e., after being selected, it must remain armed for a period of time before it can become active.

**Go Around (GA) Mode** (page 56) In Go Around mode the FGS generates guidance commands for the pilot after aborting an approach. The AP is never engaged during Go Around mode. Lateral Go Around mode provides guidance to maintain the reference heading.

**Vertical Modes** (page 60) The vertical modes select the control laws generating commands directing the aircraft along the vertical, or pitch, axis. This component describes the specific vertical modes present in this aircraft and defines how they are synchronized.

**Pitch Hold (PITCH) Mode** (page 62) In Pitch Hold mode the FGS generates guidance commands to hold the aircraft at a fixed pitch angle. Pitch Hold mode is the basic vertical mode and is always active when the modes are displayed and no other vertical mode is active.

**Vertical Speed (VS) Mode** (page 65) In Vertical Speed mode, the FGS provides pitch guidance commands to hold the aircraft to the Vertical Speed (VS) reference.

**Flight Level Change (FLC) Mode** (page 69)

In Flight Level Change mode, the FGS provides guidance commands to acquire and track an Indicated Airspeed (IAS) or Mach Reference Airspeed, taking into account the need to climb or descend to bring the aircraft to the PreSelector Altitude (PSA).

**Altitude Hold (ALT) Mode** (page 73)

In Altitude Hold mode, the FGS provides pitch guidance commands to acquire and track the Altitude reference, which is set to the current altitude when the mode is activated or upon a synchronization request by the flight crew.

**Altitude Select (ALTSEL) Mode** (page 77)

In Altitude Select mode, the FGS provides guidance commands to capture and track the Preselected Altitude. Altitude Select is an arming mode, i.e., after being selected, it must remain armed for a period of time before it can become active. Altitude Select mode is normally armed, although it is automatically deselected by the activation of

Vertical Approach, Go Around, or Altitude Hold mode. While in the armed state, the FGS monitors the aircraft closure rate towards the target altitude and determines the optimum capture point to transition to the capture state. In the capture state, the FGS generates vertical guidance commands to perform a smooth capture of the target altitude. Once it acquires the target altitude, it enters the track state, during which it generates vertical guidance commands to maintain the aircraft at the target altitude.

#### **Vertical Approach (VAPPR) Mode** (page 85)

In Vertical Approach mode, the FGS provides guidance commands to track vertical guidance for ILS precision Glideslope approaches. Vertical Approach is an arming mode, i.e., after being selected, it must remain armed for a period of time before it can become active.

#### **Vertical Go Around (VGA) Mode** (page 91)

In Go Around mode, the FGS provides guidance to the pilot after aborting an approach. The AP is never engaged during Go Around mode. Vertical Go Around mode provides guidance to maintain a fixed pitch angle.

## 2.2 Prioritization of Events

An issue that frequently arises in the specification of embedded systems is how to handle multiple events that occur at the same point in time. Here, we assume that an event corresponds to an instantaneous change in some input or state variable. One school argues that since an event occurs at a point in time, it is impossible for two events to occur at precisely the same moment, and simultaneous events should be prohibited in the underlying formal model. While theoretically appealing, many embedded systems are polling systems that periodically sample their inputs. It is difficult to imagine how a polling system could be built that could guarantee that no more than one input changes between any two samples. Even interrupt driven systems are ultimately implemented as digital systems where interrupts are only handled at well defined boundaries, such as the end of an instruction, and multiple interrupts can be observed at these boundaries.

From a practical standpoint, it seems that the required abstraction is to allow one or more events to occur at the same time, and to deal with this within the specification. RSML<sup>-e</sup> makes the *one message assumption* that at most one message can arrive at any given point in time. However, within a message, several fields can change value, allowing more than one input event to occur at the same time. Since we did not know how the inputs to the mode logic would be implemented in a specific system, we decided to group the inputs into large messages in order to make as few assumptions as possible about the disjointness of input events.

In verifying the behavior of the mode logic, it became clear that the behavior of the system when two or more events occurred at the same time would indeed have to be carefully considered if the system was to have the correct behavior and satisfy even the most basic of safety properties (e.g., no more than one lateral mode or vertical mode active at the same time). Our approach was to define a prioritization of the input events such that only the highest priority event is seen by the main body of the specification and the lower priority events are discarded. For example,

it seems reasonable to respond to the pilot pressing the Go Around switch and ignore the copilot pressing the Flight Director switch, rather than the other way around.

This approach is reasonable for most combinations of events. However, a few events, such as the acquisition of a navigation beacon, do not need to be acted upon immediately, but should not be simply discarded. These were given the lowest possible priority, and the condition of their being true, rather than the event of their becoming true, was used to trigger the associated transitions. In this way, processing of the event might be delayed, but would always be completed as soon as no higher priority event blocked it.

Other approaches are of course feasible. For example, one could queue the input events and process them in the order received. This introduces considerable complexity into the model, and the simpler prioritization described here actually appears to be the more appropriate solution. One advantage of this approach is that it makes the solution explicit and easily reviewed.

We also attempted to localize this logic in the specification by introducing macros (predicates) of the form *When\_Event\_Name\_Seen*. These macros are true only when the underlying event occurs and no higher priority event has occurred at the same time. By using only the *When\_Event\_Name\_Seen* macros in the body of the specification, it is straightforward to change the prioritization of events, or even to implement a more complicated approach such as the queueing model discussed earlier.

As formal verification of the model proceeded, it became clear that it was possible to process some combinations of events in parallel. As a result we changed the prioritization from a total to a partial order, as shown in Figure 2.2. In this diagram, an event has a higher priority than all events that can be reached from it by following a downward chain of arrow and will supersede those events. If no path exists between two events, then the two events can be processed concurrently. Thus, pressing the SYNC switch supersedes all events except pressing the AP Disconnect or AP Engage switch, which can be processed in parallel with it. In like manner, pressing the HDG switch will supersede pressing the NAV switch, the Pilot Flying Switch, the Flight Director switch, and any of the capture or track conditions. However, a press of the HDG switch can be processed in parallel with any of the switches associated only with vertical modes, such as the ALT switch.

It is worth noting that this partial order could only have been verified with confidence through the use of formal analysis tools such as model checkers and theorem provers. A full description of this verification activity will be provided in a later report.

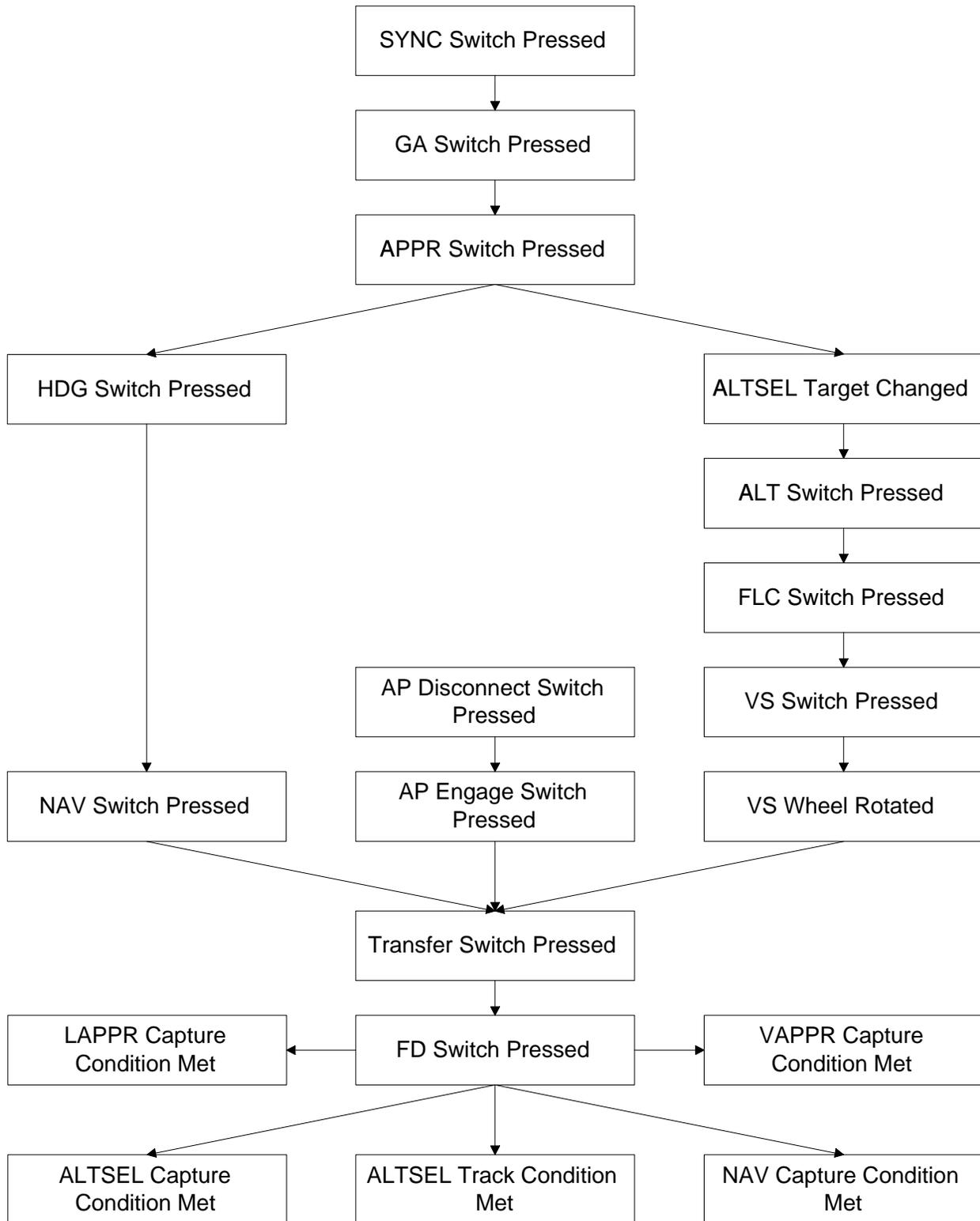


Figure 2.2: The Partial Order of Event Priorities

## 2.3 Basic Definitions

This section defines types and constants that are used throughout the specification.

---

TYPE

---

**Side**

---

**Possible Values:** LEFT, RIGHT

---

TYPE

---

**On\_Off**

---

**Possible Values:** Off, On

---

TYPE

---

**Base\_State**

---

**Possible Values:** Cleared, Selected

---

TYPE

---

**Selected\_State**

---

**Possible Values:** Armed, Active

---

TYPE

---

**Active\_State**

---

**Possible Values:** Capture, Track

---

CONSTANT
----------

---

**THIS\_SIDE**

---

**Units:** N/A

**Value:** LEFT

**Purpose:** This constant defines which side of the FGS (channel) is being specified. The specifications of both sides, left and right, are identical except for this constant.

## 2.4 Flight Director (FD)

The Flight Director (FD) displays the pitch and roll guidance commands to the pilot and copilot on the Primary Flight Display. This component defines when the Flight Director guidance cues are turned on and off.

<b>Definitions of Values to be Imported</b>
---

MACRO
-------

### When\_Turn\_FD\_On

**Condition:**

	<i>OR</i>					
A N D	When_FD_Switch_Pressed_Seen <sub>m-103</sub> ()	T	·	·	·	·
	When(AP <sub>v-132</sub> =Engaged)	·	T	·	·	·
	Overspeed_Condition <sub>m-128</sub> ()	·	·	T	·	·
	When_Lateral_Mode_Manually_Selected <sub>m-26</sub> ()	·	·	·	T	·
	When_Vertical_Mode_Manually_Selected <sub>m-27</sub> ()	·	·	·	·	T
	When_Pilot_Flying_Transfer <sub>m-29</sub> ()	·	·	·	·	T
	Pilot_Flying <sub>v-29</sub> =THIS_SIDE <sub>LEFT</sub>	·	·	·	·	T
	PREVIOUS_STEP(Mode_Annunciations_On <sub>v-33</sub> )	·	·	·	·	T

**Purpose:** This event defines when the onside FD is to be turned on (i.e., displayed on the PFD).

---

MACRO

---

**When\_Turn\_FD\_Off**

---

**Condition:**

A	When_FD_Switch_Pressed_Seen <sub>m-103</sub> ()	T
D	Overspeed_Condition <sub>m-128</sub> ()	F

**Purpose:** This event defines when the onside FD is to be turned off (i.e., removed from the PFD).

---

MACRO

---

**When\_Lateral\_Mode\_Manually\_Selected**

---

**Condition:**

						<i>OR</i>			
	PREVIOUS_STEP(Mode_Annunciations_On <sub>v-33</sub> )	F	F	F	F	·	·	·	·
A	Is_This_Side_Active <sub>v-30</sub>	·	·	·	·	T	T	T	T
N	When_HDG_Switch_Pressed_Seen <sub>m-109</sub> ()	T	·	·	·	T	·	·	·
D	When_NAV_Switch_Pressed_Seen <sub>m-108</sub> ()	·	T	·	·	·	T	·	·
	When_APPR_Switch_Pressed_Seen <sub>m-115</sub> ()	·	·	T	·	·	·	T	·
	When_GA_Switch_Pressed_Seen <sub>m-116</sub> ()	·	·	·	T	·	·	·	T

**Purpose:** This event defines when a lateral mode is manually selected.



<b>Definitions of Values to be Exported</b>
---

---

STATE VARIABLE
----------------

---

**Onside\_FD\_On**

---

**Parent:** NONE

**Type:** BOOLEAN

**Initial Value:** FALSE

**Classified as:** CONTROLLED

**:=** Onside\_FD<sub>v-28</sub> =On **if** TRUE

**Purpose:** Indicates if the FD Guidance cues should be displayed on the PFD.

<b>Definitions of Values to be Encapsulated</b>
---

---

STATE VARIABLE
----------------

---

**Onside\_FD**

---

**Parent:** NONE

**Type:** {Off, On}<sub>ty-23</sub>

**Initial Value:** Off

**Classified as:** State

→ 



**if** When\_Turn\_FD\_On<sub>m-25</sub>()

→ 



**if** When\_Turn\_FD\_Off<sub>m-26</sub>()

**Purpose:** This variable maintains the current state of the onside Flight Director.

## 2.5 Pilot Flying (PF)

The transfer switch on the FCP is used to determine which FGS provides mode annunciations and guidance commands to the PFDs while in dependent mode. This component defines which side is the Pilot Flying side. It also defines the notion of the “active” side of the FGS. A side of the FGS is said to be “active” if it is providing mode annunciations and guidance commands to its PFD.

---

### Definitions of Values to be Exported

---

STATE VARIABLE

---

#### Pilot\_Flying

---

**Parent:** NONE

**Type:** {LEFT, RIGHT}<sub>ty-23</sub>

**Initial Value:** LEFT

**Classified as:** State

**LEFT** → **RIGHT** if When\_Transfer\_Switch\_Pressed\_Seen<sub>m-104</sub>()

**RIGHT** → **LEFT** if When\_Transfer\_Switch\_Pressed\_Seen<sub>m-104</sub>()

**Purpose:** This state variable maintains which side this side of the FGS believes is the Pilot Flying side.

---

MACRO

---

#### When\_Pilot\_Flying\_Transfer

---

**Condition:** Changed(Pilot\_Flying<sub>v-29</sub>)

**Purpose:** This event occurs when the pilot flying side changes to the other side.

## 2.6 Independent Mode

Normally, the FGS operates in dependent mode in which the pilot flying (PF) side of the FGS provides the mode annunciations and FD guidance commands to both PFDs. While in dependent mode, the pilot not flying (PNF) side of the FGS synchronizes its modes to the pilot flying side. Under final approach conditions and during takeoff or go around modes, the FGS operates in independent mode in which each side operates independently of the other side and provides mode annunciations and FD guidance commands to its own PFD. This component defines when the FGS enters independent mode.

<b>Definitions of Values to be Exported</b>
---

STATE VARIABLE
----------------

**Is\_This\_Side\_Active**

**Parent:** NONE

**Type:** BOOLEAN

**Initial Value:** TRUE

**Classified as:** State

**:= TRUE if**

A N D	PREVIOUS STEP(Independent_Mode <sub>v-31</sub> ) =Off	OR	F	T	T
	Pilot_Flying <sub>v-29</sub> =THIS_SIDE <sub>LEFT</sub>		·	T	F
	Offside_Modes <sub>v-135</sub> =On		·	·	F

**:= FALSE if**

A N D	PREVIOUS STEP(Independent_Mode <sub>v-31</sub> ) =Off	T
	Pilot_Flying <sub>v-29</sub> =THIS_SIDE <sub>LEFT</sub>	F
	Offside_Modes <sub>v-135</sub> =On	T

**Purpose:** A side of the FGS is said to be active if it is the source for guidance to the AP its FD. This occurs if the system is in independent mode, if this side of the FGS is the pilot flying side, or if the other side of the FGS is the pilot flying side but its mode annunciations are not turned on.

<b>Definitions of Values to be Encapsulated</b>
---

---

MACRO
-------

---

### Independent\_Mode\_Condition

---

**Condition:**

		<i>OR</i>
A	Is_LAPPR_Active <sub>v-52</sub>	T
N	Is_VAPPR_Active <sub>v-87</sub>	·
D	Is_Offside_LAPPR_Active <sub>v-134</sub>	T
	Is_Offside_VAPPR_Active <sub>v-134</sub>	·
	Is_VGA_Active <sub>v-92</sub>	·
	Is_Offside_VGA_Active <sub>v-135</sub>	T

**Purpose:** This condition defines when the system is in independent mode. This occurs when lateral and vertical approach modes are active on both sides (channels) or vertical Go Around mode is active.

---

STATE VARIABLE
----------------

---

### Independent\_Mode

---

**Parent:** NONE

**Type:** {Off, On}<sub>ty-23</sub>

**Initial Value:** Off

**Classified as:** State

**:= On if** Independent\_Mode\_Condition<sub>m-31</sub>()

**:= Off if** **not** Independent\_Mode\_Condition<sub>m-31</sub>()

**Purpose:** This variable maintains the state of whether this side is in independent mode.

**Comment:** This variable is introduced to make it easier to access its previous value and so that it can be displayed during simulation.

## 2.7 Flight Modes

The flight modes determine which modes of operation of the FGS are active and armed at any given moment. These in term determine which flight control laws are generating the commands directing the aircraft along the lateral (roll) and vertical (pitch) axes. This component describes the lateral and vertical modes and defines how they are synchronized.

<b>Definitions of Values to be Imported</b>
---

---

MACRO
-------

---

### When\_Turn\_Modes\_On

---

**Condition:**

$\begin{matrix} A \\ N \\ D \end{matrix}$	Onside_FD <sub>v-28</sub> =On	<i>OR</i> <table border="1" style="border-collapse: collapse; text-align: center;"> <tr><td>T</td><td>·</td><td>·</td></tr> <tr><td>·</td><td>T</td><td>·</td></tr> <tr><td>·</td><td>·</td><td>T</td></tr> </table>	T	·	·	·	T	·	·	·	T
	T		·	·							
	·		T	·							
·	·	T									
Offside_FD <sub>v-133</sub> =On											
Is_AP_Engaged <sub>v-131</sub>											

**Purpose:** This event defines when the flight modes are to be turned on and displayed on the PFD.

---

MACRO
-------

---

### When\_Turn\_Modes\_Off

---

**Condition:**

$\begin{matrix} A \\ N \\ D \end{matrix}$	Onside_FD <sub>v-28</sub> =Off	<table border="1" style="border-collapse: collapse; text-align: center;"> <tr><td>T</td></tr> <tr><td>T</td></tr> <tr><td>T</td></tr> </table>	T	T	T
	T				
	T				
T					
Offside_FD <sub>v-133</sub> =Off					
AP <sub>v-132</sub> =Disengaged					

**Purpose:** This event defines when the flight modes are to be turned off and removed from the PFD.

Definitions of Values to be Exported
--------------------------------------

---

STATE VARIABLE
----------------

---

Mode\_Annunciations\_On

---

**Parent:** NONE

**Type:** BOOLEAN

**Initial Value:** FALSE

**Classified as:** CONTROLLED

**:=** Modes<sub>v-34</sub> =On **if** TRUE

**Purpose:** Indicates if the mode annunciations should be displayed on the PFD.

Definitions of Values to be Encapsulated
--

---

STATE VARIABLE
----------------

---

### Modes

---

**Parent:** NONE

**Type:** {Off, On}<sub>ty-23</sub>

**Initial Value:** Off

**Classified as:** State

**:=** Offside\_Modes<sub>v-135</sub> **if not** Is\_This\_Side\_Active<sub>v-30</sub>

Off → On **if**

$\frac{A}{N}$	Is_This_Side_Active <sub>v-30</sub>	T
$\frac{D}{D}$	When_Turn_Modes_On <sub>m-32</sub> ()	T

On → Off **if**

$\frac{A}{N}$	Is_This_Side_Active <sub>v-30</sub>	T
$\frac{D}{D}$	When_Turn_Modes_Off <sub>m-32</sub> ()	T

**Purpose:** This variable maintains the current state of whether the mode annunciations are turned on or off.

### 2.7.1 Lateral Modes

The lateral modes select the control laws generating commands directing the aircraft along the lateral, or roll, axis. This component describes the specific lateral modes present in this aircraft and defines how they are synchronized.

<b>Definitions of Values to be Encapsulated</b>
---

MACRO
-------

<b>When_Nonbasic_Lateral_Mode_Activated</b>
---

**Condition:**

		<i>OR</i>			
A	When_HDG_Activated <sub>m-42</sub> ()	T	·	·	·
N	When_NAV_Activated <sub>m-47</sub> ()	·	T	·	·
D	When_LAPPR_Activated <sub>m-53</sub> ()	·	·	T	·
	When_LGA_Activated <sub>m-58</sub> ()	·	·	·	T

**Purpose:** This event occurs when a new lateral mode other than the basic mode becomes active. It is used to deselect active or armed modes.

**Comment:** Basic mode is excluded to avoid a cyclic dependency in the definition of this macro.

---

 MACRO
 

---



---

**Is\_No\_Nonbasic\_Lateral\_Mode\_Active**


---

**Condition:**

A N D	Is_HDG_Active <sub>v-41</sub>	F
	Is_NAV_Active <sub>v-46</sub>	F
	Is_LAPPR_Active <sub>v-52</sub>	F
	Is_LGA_Active <sub>v-57</sub>	F

**Purpose:** This condition indicates if no lateral mode except basic mode is active. It is used to trigger the activation of the basic lateral mode.

**Comment:** Basic mode is excluded to avoid a cyclic dependency in the definition of this macro.

### 2.7.1.1 Roll Hold (ROLL) Mode

In Roll Hold mode the FGS generates guidance commands to hold the aircraft at a fixed bank angle. Roll Hold mode is the basic lateral mode and is always active when the modes are displayed and no other lateral mode is active.

<b>Definitions of Values to be Imported</b>
---

---

MACRO
-------

---

#### Select\_ROLL

---

**Condition:**

A		T
N	Is_No_Nonbasic_Lateral_Mode_Active <sub>m-36</sub> ()	T
D	Modes <sub>v-34</sub> =On	T

**Purpose:** This event defines when Roll Hold mode is to be selected. Roll Hold mode is the basic, or default, mode and is selected whenever the mode annunciations are on and no other lateral mode is active.

**Comment:** To avoid cyclic dependencies, the only way to select Roll Hold mode is to deselect the active lateral mode, which will automatically activate Roll Hold.

---

MACRO
-------

---

#### Deselect\_ROLL

---

**Condition:**

		<i>OR</i>	
A		T	·
N	When_Nonbasic_Lateral_Mode_Activated <sub>m-35</sub> ()	·	T
D	When(Modes <sub>v-34</sub> =Off)	T	·

**Purpose:** The event defines when Roll Hold mode is to be deselected. This occurs when a new lateral mode is activated or the modes are turned off.

---

Definitions of Values to be Exported
--------------------------------------

---

STATE VARIABLE
----------------

---

**Is\_ROLL\_Selected**

---

**Parent:** NONE

**Type:** BOOLEAN

**Initial Value:** FALSE

**Classified as:** CONTROLLED

**:=** (ROLL<sub>v-39</sub> =Selected) **if** TRUE

**Purpose:** Indicates if ROLL mode is selected.

---

STATE VARIABLE
----------------

---

**Is\_ROLL\_Active**

---

**Parent:** NONE

**Type:** BOOLEAN

**Initial Value:** FALSE

**Classified as:** CONTROLLED

**:=** (ROLL<sub>v-39</sub> =Selected) **if** TRUE

**Purpose:** Indicates if ROLL mode is selected.

**Comment:** Even though ROLL Selected and ROLL Active are the same value, this is defined to provide a common set of controlled variables across all modes.

<b>Definitions of Values to be Encapsulated</b>
---

---

STATE VARIABLE
----------------

---

**ROLL**

---

**Parent:** Modes<sub>v-34</sub>.On

**Type:** {Cleared, Selected}<sub>ty-23</sub>

**Initial Value:** UNDEFINED

**Classified as:** State

**:=** Offside\_ROLL<sub>v-136</sub> **if not** Is\_This\_Side\_Active<sub>v-30</sub>

UNDEFINED → Cleared **if**

$\begin{matrix} A \\ N \\ D \end{matrix}$	Select_ROLL <sub>m-37</sub> ()	F
	Is_This_Side_Active <sub>v-30</sub>	T

UNDEFINED → Selected **if**

$\begin{matrix} A \\ N \\ D \end{matrix}$	Select_ROLL <sub>m-37</sub> ()	T
	Is_This_Side_Active <sub>v-30</sub>	T

Cleared → Selected **if**

$\begin{matrix} A \\ N \\ D \end{matrix}$	Select_ROLL <sub>m-37</sub> ()	T
	Is_This_Side_Active <sub>v-30</sub>	T

Selected → Cleared **if**

$\begin{matrix} A \\ N \\ D \end{matrix}$	Deselect_ROLL <sub>m-37</sub> ()	T
	Is_This_Side_Active <sub>v-30</sub>	T

**Purpose:** This variable maintains the current base state of Roll Hold mode, i.e., whether it is cleared or selected. Note that if this side of the FGS is not active, it obtains its value from the off side. The normal transitions occur only when this side is active.

**2.7.1.2 Heading Select (HDG) Mode**

In Heading Select mode, the FGS provides guidance commands to track the Selected Heading displayed on the PFD.

**Definitions of Values to be Imported**

MACRO

**Select\_HDG**

**Condition:** When\_HDG\_Switch\_Pressed\_Seen<sub>m-109</sub>()

**Purpose:** This event defines when Heading Select mode is to be selected.

MACRO

**Deselect\_HDG**

**Condition:**

	<i>OR</i>			
<i>A</i>	When_HDG_Switch_Pressed_Seen <sub>m-109</sub> ()	T	·	·
<i>N</i>	When_Nonbasic_Lateral_Mode_Activated <sub>m-35</sub> ()	·	T	·
<i>D</i>	When_Pilot_Flying_Transfer <sub>m-29</sub> ()	·	·	T
	When(Modes <sub>v-34</sub> =Off)	·	·	T

**Purpose:** This event defines when Heading Select mode is to be deselected.

Definitions of Values to be Exported
--------------------------------------

---

STATE VARIABLE
----------------

---

**Is\_HDG\_Selected**

---

**Parent:** NONE

**Type:** BOOLEAN

**Initial Value:** FALSE

**Classified as:** CONTROLLED

**:=** (HDG<sub>v-43</sub> =Selected) **if** TRUE

**Purpose:** Indicates if HDG mode is selected.

---

STATE VARIABLE
----------------

---

**Is\_HDG\_Active**

---

**Parent:** NONE

**Type:** BOOLEAN

**Initial Value:** FALSE

**Classified as:** CONTROLLED

**:=** (HDG<sub>v-43</sub> =Selected) **if** TRUE

**Purpose:** Indicates if HDG mode is active.

**Comment:** Even though HDG Selected and HDG Active are the same value, this is defined to provide a common set of controlled variables across all modes.

---

MACRO
-------

---

### When\_HDG\_Activated

---

**Condition:**

A	Select_HDG <sub>m-40</sub> ()	T
N	PREVIOUS_STEP(HDG <sub>v-43</sub> ) = Selected	F
D	Is_This_Side_Active <sub>v-30</sub>	T

**Purpose:** This signal occurs when Heading Select mode is activated.

**Comment:** This event is defined this way to avoid circular dependencies. It would be preferable to define it as When(HDG = Selected).

<b>Definitions of Values to be Encapsulated</b>
---

---

STATE VARIABLE
----------------

---

## HDG

---

**Parent:** Modes<sub>v-34</sub>.On

**Type:** {Cleared, Selected}<sub>ty-23</sub>

**Initial Value:** UNDEFINED

**Classified as:** State

**Purpose:** This variable maintains the current base state of Heading Select mode, i.e., whether it is cleared or selected. Note that if this side of the FGS is not active, it obtains its value from the off side. The normal transitions occur only when this side is active.

**:=** Offside\_HDG<sub>v-136</sub> **if** **not** Is\_This\_Side\_Active<sub>v-30</sub>

UNDEFINED → Cleared **if**

$\begin{matrix} A \\ N \\ D \end{matrix}$	Select_HDG <sub>m-40</sub> ()	F
	Is_This_Side_Active <sub>v-30</sub>	T

UNDEFINED → Selected **if**

$\begin{matrix} A \\ N \\ D \end{matrix}$	Select_HDG <sub>m-40</sub> ()	T
	Is_This_Side_Active <sub>v-30</sub>	T

Cleared → Selected **if**

$\begin{matrix} A \\ N \\ D \end{matrix}$	Select_HDG <sub>m-40</sub> ()	T
	Is_This_Side_Active <sub>v-30</sub>	T

Selected → Cleared **if**

$\begin{matrix} A \\ N \\ D \end{matrix}$	Deselect_HDG <sub>m-40</sub> ()	T
	Is_This_Side_Active <sub>v-30</sub>	T

### 2.7.1.3 Navigation (NAV) Mode

In Lateral Navigation mode, the FGS provides guidance commands to acquire and track lateral guidance for en route navigation and non-precision approaches. Lateral Navigation is an arming mode, i.e., after being selected, it must remain armed for a period of time before it can become active.

<b>Definitions of Values to be Imported</b>
---

---

MACRO
-------

---

#### Select\_NAV

---

**Condition:** When\_NAV\_Switch\_Pressed\_Seen<sub>m-108</sub>()

**Purpose:** This event defines when Lateral Navigation mode is to be selected, i.e., to become armed.

---

MACRO
-------

---

#### Activate\_NAV

---

**Condition:** When\_NAV\_Track\_Cond\_Met\_Seen<sub>m-95</sub>()

**Purpose:** This event defines when Lateral Navigation mode is to transition from the armed to the active state.

---

 MACRO
 

---

### Deselect\_NAV

---

**Condition:**

A N D	When_NAV_Switch_Pressed_Seen <sub>m-108</sub> ()	T	.	.	.	.
	When_Selected_Nav_Source_Changed <sub>v-127</sub>	.	T	.	.	.
	When_Selected_Nav_Frequency_Changed <sub>v-127</sub>	.	.	T	.	.
	When_Pilot_Flying_Transfer <sub>m-29</sub> ()	.	.	.	T	.
	When(Modes <sub>v-34</sub> =Off)	.	.	.	.	T

OR

**Purpose:** This event defines when Lateral Navigation mode is to be deselected, i.e., to be cleared regardless of whether it is armed or active.

---

 MACRO
 

---

### Dearm\_NAV

---

**Condition:** When\_LAPPR\_Armed<sub>m-53</sub>()

**Purpose:** This event defines when Lateral Navigation mode is to be cleared if it is armed, but to remain active if it is active.

---

 MACRO
 

---

### Deactivate\_NAV

---

**Condition:** When\_Nonbasic\_Lateral\_Mode\_Activated<sub>m-35</sub>()

**Purpose:** This event defines when Lateral Navigation mode is to be cleared if it is active, but to remain armed if it is armed.

Definitions of Values to be Exported
--------------------------------------

---

STATE VARIABLE
----------------

---

**Is\_NAV\_Selected**

---

**Parent:** NONE

**Type:** BOOLEAN

**Initial Value:** FALSE

**Classified as:** CONTROLLED

**:=** (NAV<sub>v-48</sub> =Selected) **if** TRUE

**Purpose:** Indicates if NAV mode is selected.

---

STATE VARIABLE
----------------

---

**Is\_NAV\_Active**

---

**Parent:** NONE

**Type:** BOOLEAN

**Initial Value:** FALSE

**Classified as:** CONTROLLED

**:=** (NAV\_Selected<sub>v-49</sub> =Active) **if** TRUE

**Purpose:** Indicates if NAV mode is active.

---

 MACRO
 

---

**When\_NAV\_Armed**


---

**Condition:**

A	Select_NAV <sub>m-44</sub> ()	T
N	PREVIOUS_STEP(NAV <sub>v-48</sub> ) =Cleared	T
D	Is_This_Side_Active <sub>v-30</sub>	T

**Purpose:** This signal occurs when Lateral NAV mode is armed.

**Comment:** This event is defined this way to avoid circular dependencies. It would be preferable to define it as When(NAV\_Selected = Armed).

---

 MACRO
 

---

**When\_NAV\_Activated**


---

**Condition:**

A	Activate_NAV <sub>m-44</sub> ()	T
N	PREVIOUS_STEP(NAV_Selected <sub>v-49</sub> ) =Armed	T
D	Is_This_Side_Active <sub>v-30</sub>	T

**Purpose:** This signal occurs when Lateral Navigation mode is activated.

**Comment:** This event is defined this way to avoid circular dependencies. It would be preferable to define it as When(NAV\_Selected = Active).

<b>Definitions of Values to be Encapsulated</b>
---

---

STATE VARIABLE
----------------

---

**NAV**

---

**Parent:** Modes<sub>v-34</sub>.On

**Type:** {Cleared, Selected}<sub>ty-23</sub>

**Initial Value:** UNDEFINED

**Classified as:** State

**:=** Offside\_NAV<sub>v-137</sub> **if not** Is\_This\_Side\_Active<sub>v-30</sub>

UNDEFINED → Cleared **if**

$\begin{matrix} A \\ N \\ D \end{matrix}$	Select_NAV <sub>m-44</sub> ()	F
	Is_This_Side_Active <sub>v-30</sub>	T

UNDEFINED → Selected **if**

$\begin{matrix} A \\ N \\ D \end{matrix}$	Select_NAV <sub>m-44</sub> ()	T
	Is_This_Side_Active <sub>v-30</sub>	T

Cleared → Selected **if**

$\begin{matrix} A \\ N \\ D \end{matrix}$	Select_NAV <sub>m-44</sub> ()	T
	Is_This_Side_Active <sub>v-30</sub>	T

Selected → Cleared **if**

$\begin{matrix} A \\ N \\ D \end{matrix}$	Deselect_NAV <sub>m-45</sub> ()	T	·	·
	Dearm_NAV <sub>m-45</sub> ()	·	T	·
	PREVIOUS STEP(NAV_Selected <sub>v-49</sub> ) = Armed	·	T	·
	Deactivate_NAV <sub>m-45</sub> ()	·	·	T
	PREVIOUS STEP(NAV_Selected <sub>v-49</sub> ) = Active	·	·	T
	Is_This_Side_Active <sub>v-30</sub>	T	T	T

*OR*

**Purpose:** This variable maintains the current base state of Lateral Navigation mode, i.e., whether it is cleared or selected. Note that if this side of the FGS is not active, it obtains its value from the off side. The normal transitions occur only when this side is active.

---

STATE VARIABLE

---

## NAV\_Selected

---

**Parent:** Modes<sub>v-34</sub>.On  $\triangleright$  NAV<sub>v-48</sub>.Selected

**Type:** {Armed, Active}<sub>ty-23</sub>

**Initial Value:** UNDEFINED

**Classified as:** State

**:=** Offside\_NAV\_Selected<sub>v-137</sub> **if** **not** Is\_This\_Side\_Active<sub>v-30</sub>

UNDEFINED  $\rightarrow$  Armed **if** TRUE

Armed  $\rightarrow$  Active **if**

A	Activate_NAV <sub>m-44</sub> ()	T
D	Is_This_Side_Active <sub>v-30</sub>	T

**Purpose:** This variable maintains the current state of Lateral Navigation mode when it is selected, i.e., whether it is armed or active. Note that if this side of the FGS is not active, it obtains its value from the off side. The normal transitions occur only when this side is active.

#### 2.7.1.4 Lateral Approach (LAPPR) Mode

In Lateral Approach mode, the FGS provides guidance commands to acquire and track lateral guidance for precision and non-precision approaches. Lateral Approach is an arming mode, i.e., after being selected, it must remain armed for a period of time before it can become active.

<b>Definitions of Values to be Imported</b>
---

---

MACRO
-------

---

#### Select\_LAPPR

---

**Condition:** When\_APPR\_Switch\_Pressed\_Seen<sub>m-115</sub>()

**Purpose:** This event defines when Lateral Approach mode is to be selected, i.e., to become armed.

---

MACRO
-------

---

#### Activate\_LAPPR

---

**Condition:** When\_LAPPR\_Track\_Cond\_Met\_Seen<sub>m-98</sub>()

**Purpose:** This event defines when Lateral Approach mode is to transition from the armed to the active state.

---

 MACRO
 

---

### Deselect\_LAPPR

---

**Condition:**

$\begin{matrix} A \\ N \\ D \end{matrix}$	When_APPR_Switch_Pressed_Seen <sub>m-115</sub> ()	T	.	.	.	.
	When_Selected_Nav_Source_Changed <sub>v-127</sub>	.	T	.	.	.
	When_Selected_Nav_Frequency_Changed <sub>v-127</sub>	.	.	T	.	.
	When_Pilot_Flying_Transfer <sub>m-29</sub> ()	.	.	.	T	.
	When(Modes <sub>v-34</sub> =Off)	.	.	.	.	T

OR

**Purpose:** This event defines when Lateral Approach mode is to be deselected, i.e., to be cleared regardless of whether it is armed or active.

---

 MACRO
 

---

### Dearm\_LAPPR

---

**Condition:** When\_NAV\_Armed<sub>m-47</sub>()

**Purpose:** This event defines when Lateral Approach mode is to be cleared if it is armed, but to remain active if it is active.

---

 MACRO
 

---

### Deactivate\_LAPPR

---

**Condition:** When\_Nonbasic\_Lateral\_Mode\_Activated<sub>m-35</sub>()

**Purpose:** This event defines when Lateral Approach mode is to be cleared if it is active, but to remain armed if it is armed.

Definitions of Values to be Exported
--------------------------------------

---

STATE VARIABLE
----------------

---

**Is\_LAPPR\_Selected**

---

**Parent:** NONE**Type:** BOOLEAN**Initial Value:** FALSE**Classified as:** CONTROLLED**:=** (LAPPR<sub>v-54</sub> =Selected) **if** TRUE**Purpose:** Indicates if LAPPR mode is selected.

---

STATE VARIABLE
----------------

---

**Is\_LAPPR\_Active**

---

**Parent:** NONE**Type:** BOOLEAN**Initial Value:** FALSE**Classified as:** CONTROLLED**:=** (LAPPR\_Selected<sub>v-55</sub> =Active) **if** TRUE**Purpose:** Indicates if LAPPR mode is active.

---

 MACRO
 

---

### When\_LAPPR\_Armed

---

**Condition:**

A	Select_LAPPR <sub>m-50</sub> ()	T
N	PREVIOUS_STEP(LAPPR <sub>v-54</sub> ) =Cleared	T
D	Is_This_Side_Active <sub>v-30</sub>	T

**Purpose:** This signal occurs when Lateral Approach mode is armed.

**Comment:** This event is defined this way to avoid circular dependencies. It would be preferable to define it as When(LAPPR.Selected = Armed).

---

 MACRO
 

---

### When\_LAPPR\_Activated

---

**Condition:**

A	Activate_LAPPR <sub>m-50</sub> ()	T
N	PREVIOUS_STEP(LAPPR_Selected <sub>v-55</sub> ) =Armed	T
D	Is_This_Side_Active <sub>v-30</sub>	T

**Purpose:** This signal occurs when Lateral Approach mode is activated.

**Comment:** This event is defined this way to avoid circular dependencies. It would be preferable to define it as When(LAPPR.Selected = Active).

<b>Definitions of Values to be Encapsulated</b>
---

---

STATE VARIABLE
----------------

---

## LAPPR

---

**Parent:** Modes<sub>v-34</sub>.On

**Type:** {Cleared, Selected}<sub>ty-23</sub>

**Initial Value:** UNDEFINED

**Classified as:** State

**:=** Offside\_LAPPR<sub>v-138</sub> **if** **not** Is\_This\_Side\_Active<sub>v-30</sub>

UNDEFINED → Cleared **if**

$\begin{matrix} A \\ N \\ D \end{matrix}$	Select_LAPPR <sub>m-50</sub> ()	F
	Is_This_Side_Active <sub>v-30</sub>	T

UNDEFINED → Selected **if**

$\begin{matrix} A \\ N \\ D \end{matrix}$	Select_LAPPR <sub>m-50</sub> ()	T
	Is_This_Side_Active <sub>v-30</sub>	T

Cleared → Selected **if**

$\begin{matrix} A \\ N \\ D \end{matrix}$	Select_LAPPR <sub>m-50</sub> ()	T
	Is_This_Side_Active <sub>v-30</sub>	T

Selected → Cleared **if**

$\begin{matrix} A \\ N \\ D \end{matrix}$	Deselect_LAPPR <sub>m-51</sub> ()	T	·	·
	Dearm_LAPPR <sub>m-51</sub> ()	·	T	·
	PREVIOUS_STEP(LAPPR_Selected <sub>v-55</sub> ) = Armed	·	T	·
	Deactivate_LAPPR <sub>m-51</sub> ()	·	·	T
	PREVIOUS_STEP(LAPPR_Selected <sub>v-55</sub> ) = Active	·	·	T
	Is_This_Side_Active <sub>v-30</sub>	T	T	T

*OR*

**Purpose:** This variable maintains the current base state of Lateral Approach mode, i.e., whether it is cleared or selected. Note that if this side of the FGS is not active, it obtains its value from the off side. The normal transitions occur only when this side is active.

---

STATE VARIABLE

---

## LAPPR\_Selected

---

**Parent:** Modes<sub>v-34</sub>.On  $\triangleright$  LAPPR<sub>v-54</sub>.Selected

**Type:** {Armed, Active}<sub>ty-23</sub>

**Initial Value:** UNDEFINED

**Classified as:** State

**:=** Offside\_LAPPR\_Selected<sub>v-138</sub> **if** **not** Is\_This\_Side\_Active<sub>v-30</sub>

UNDEFINED  $\rightarrow$  Armed **if** TRUE

Armed  $\rightarrow$  Active **if**

$\begin{matrix} A \\ N \\ D \end{matrix}$	Activate_LAPPR <sub>m-50</sub> ()	T
	Is_This_Side_Active <sub>v-30</sub>	T

**Purpose:** This variable maintains the current state of Lateral Approach mode when it is selected, i.e., whether it is armed or active. Note that if this side of the FGS is not active, it obtains its value from the off side. The normal transitions occur only when this side is active.

**2.7.1.5 Lateral Go Around (LGA) Mode**

In Go Around mode the FGS generates guidance commands for the pilot after aborting an approach. The AP is never engaged during Go Around mode. Lateral Go Around mode provides guidance to maintain the reference heading.

**Definitions of Values to be Imported**

MACRO

**Select\_LGA**

**Condition:**

$\frac{A}{N}$	When_GA_Switch_Pressed_Seen <sub>m-116</sub> ()	T
$\frac{D}{}$	Overspeed_Condition <sub>m-128</sub> ()	F

**Purpose:** This event defines when Lateral Go Around mode is to be selected.

MACRO

**Deselect\_LGA**

**Condition:**

		<i>OR</i>					
$\frac{A}{N}$ $\frac{D}{}$	When(AP <sub>v-132</sub> =Engaged)	T	·	·	·	·	·
	When_Nonbasic_Lateral_Mode_Activated <sub>m-35</sub> ()	·	T	·	·	·	·
	When_Nonbasic_Vertical_Mode_Activated <sub>m-60</sub> ()	·	·	T	·	·	·
	When_SYNC_Switch_Pressed_Seen <sub>m-117</sub> ()	·	·	·	T	·	·
	When_Pilot_Flying_Transfer <sub>m-29</sub> ()	·	·	·	·	T	·
	When(Modes <sub>v-34</sub> =Off)	·	·	·	·	·	T

**Purpose:** This event defines when Lateral Go Around mode is to be deselected.

---

Definitions of Values to be Exported
--------------------------------------

---

STATE VARIABLE
----------------

---

**Is\_LGA\_Selected**

---

**Parent:** NONE**Type:** BOOLEAN**Initial Value:** FALSE**Classified as:** CONTROLLED**:=** (LGA<sub>v-59</sub> =Selected) **if** TRUE**Purpose:** Indicates if LGA mode is selected.

---

STATE VARIABLE
----------------

---

**Is\_LGA\_Active**

---

**Parent:** NONE**Type:** BOOLEAN**Initial Value:** FALSE**Classified as:** CONTROLLED**:=** (LGA<sub>v-59</sub> =Selected) **if** TRUE**Purpose:** Indicates if LGA mode is active.**Comment:** Even though LGA Selected and LGA Active are the same value, this is defined to provide a common set of controlled variables across all modes.

---

MACRO
-------

---

### When\_LGA\_Activated

---

**Condition:**

A	Select_LGA <sub>m-56</sub> ()	T
N	PREVIOUS_STEP(LGA <sub>v-59</sub> ) =Selected	F
D	Is_This_Side_Active <sub>v-30</sub>	T

**Purpose:** This signal occurs when Lateral Go Around mode is activated.

**Comment:** This event is defined this way to avoid circular dependencies. It would be preferable to define it as When(LGA = Selected).

<b>Definitions of Values to be Encapsulated</b>
---

---

STATE VARIABLE
----------------

---

**LGA**

---

**Parent:** Modes<sub>v-34</sub>.On

**Type:** {Cleared, Selected}<sub>ty-23</sub>

**Initial Value:** UNDEFINED

**Classified as:** State

**:=** Offside\_LGA<sub>v-139</sub> **if** **not** Is\_This\_Side\_Active<sub>v-30</sub>

UNDEFINED → Cleared **if**

$\begin{matrix} A \\ N \\ D \end{matrix}$	Select_LGA <sub>m-56</sub> ()	F
	Is_This_Side_Active <sub>v-30</sub>	T

UNDEFINED → Selected **if**

$\begin{matrix} A \\ N \\ D \end{matrix}$	Select_LGA <sub>m-56</sub> ()	T
	Is_This_Side_Active <sub>v-30</sub>	T

Cleared → Selected **if**

$\begin{matrix} A \\ N \\ D \end{matrix}$	Select_LGA <sub>m-56</sub> ()	T
	Is_This_Side_Active <sub>v-30</sub>	T

Selected → Cleared **if**

$\begin{matrix} A \\ N \\ D \end{matrix}$	Deselect_LGA <sub>m-56</sub> ()	T
	Is_This_Side_Active <sub>v-30</sub>	T

**Purpose:** This variable maintains the current state of Lateral Go Around mode. Note that if this side is not active, it obtains its value from the off side. The normal transitions for Lateral Go Around mode occur only when this side is active.

### 2.7.2 Vertical Modes

The vertical modes select the control laws generating commands directing the aircraft along the vertical, or pitch, axis. This component describes the specific vertical modes present in this aircraft and defines how they are synchronized.

<b>Definitions of Values to be Encapsulated</b>
---

MACRO
-------

---

#### When\_Nonbasic\_Vertical\_Mode\_Activated

---

**Condition:**

		<i>OR</i>								
A	N	D	When_VS_Activated <sub>m-67</sub> ()	T	·	·	·	·	·	·
			When_FLC_Activated <sub>m-71</sub> ()	·	T	·	·	·	·	·
			When_ALT_Activated <sub>m-75</sub> ()	·	·	T	·	·	·	·
			When_ALTSEL_Activated <sub>m-81</sub> ()	·	·	·	T	·	·	·
			When_VAPPR_Activated <sub>m-88</sub> ()	·	·	·	·	T	·	·
			When_VGA_Activated <sub>m-93</sub> ()	·	·	·	·	·	T	·

**Purpose:** This event indicates when a new vertical mode other than the basic mode becomes active. It is used to deselect active or armed modes.

**Comment:** Basic mode is excluded to avoid a cyclic dependency in the definition of this macro.

---

 MACRO
 

---

 Is\_No\_Nonbasic\_Vertical\_Mode\_Active
 

---

**Condition:**

A N D	Is_VS_Active <sub>v-66</sub>	F
	Is_FLC_Active <sub>v-71</sub>	F
	Is_ALT_Active <sub>v-74</sub>	F
	Is_ALTSEL_Active <sub>v-80</sub>	F
	Is_VAPPR_Active <sub>v-87</sub>	F
	Is_VGA_Active <sub>v-92</sub>	F

**Purpose:** This condition indicates if no vertical mode except basic mode is active. It is used to trigger the activation of the basic lateral mode.

**Comment:** Basic mode is excluded to avoid a cyclic dependency in the definition of this macro.

### 2.7.2.1 Pitch Hold (PITCH) Mode

In Pitch Hold mode the FGS generates guidance commands to hold the aircraft at a fixed pitch angle. Pitch Hold mode is the basic vertical mode and is always active when the modes are displayed and no other vertical mode is active.

<b>Definitions of Values to be Imported</b>
---

---

MACRO
-------

---

#### Select\_PITCH

---

**Condition:**

$\begin{matrix} A \\ N \\ D \end{matrix}$	Is_No_Nonbasic_Vertical_Mode_Active <sub>m-61</sub> ()	T
	Modes <sub>v-34</sub> =On	T

**Purpose:** Pitch Hold mode is the basic, or default, mode and is selected whenever the mode annunciations are on and no other vertical mode is active.

**Comment:** To avoid cyclic dependencies, the only way to select Pitch Hold mode is to deselect the active vertical mode, which will automatically activate Pitch Hold mode.

---

MACRO
-------

---

#### Deselect\_PITCH

---

**Condition:**

$\begin{matrix} A \\ N \\ D \end{matrix}$	When_Nonbasic_Vertical_Mode_Activated <sub>m-60</sub> ()	T	OR	·
	When(Modes <sub>v-34</sub> =Off)	·		T

**Purpose:** Pitch Hold mode is deselected when: a new vertical mode is activated or the modes are turned off.

---

**Definitions of Values to be Exported**

---

---

**STATE VARIABLE**

---

**Is\_PITCH\_Selected**

---

**Parent:** NONE**Type:** BOOLEAN**Initial Value:** FALSE**Classified as:** CONTROLLED**:=** (PITCH<sub>v-64</sub> =Selected) **if** TRUE**Purpose:** Indicates if PITCH mode is selected.

---

**STATE VARIABLE**

---

**Is\_PITCH\_Active**

---

**Parent:** NONE**Type:** BOOLEAN**Initial Value:** FALSE**Classified as:** CONTROLLED**:=** (PITCH<sub>v-64</sub> =Selected) **if** TRUE**Purpose:** Indicates if PITCH mode is active.**Comment:** Even though PITCH Selected and PITCH Active are the same value, this is defined to provide a common set of controlled variables across all modes.

<b>Definitions of Values to be Encapsulated</b>
---

---

STATE VARIABLE
----------------

---

**PITCH**

---

**Parent:** Modes<sub>v-34</sub>.On

**Type:** {Cleared, Selected}<sub>ty-23</sub>

**Initial Value:** UNDEFINED

**Classified as:** State

**:=** Offside\_PITCH<sub>v-139</sub> **if** **not** Is\_This\_Side\_Active<sub>v-30</sub>

UNDEFINED → Cleared **if**

$\begin{matrix} A \\ N \\ D \end{matrix}$	Select_PITCH <sub>m-62</sub> ()	F
	Is_This_Side_Active <sub>v-30</sub>	T

UNDEFINED → Selected **if**

$\begin{matrix} A \\ N \\ D \end{matrix}$	Select_PITCH <sub>m-62</sub> ()	T
	Is_This_Side_Active <sub>v-30</sub>	T

Cleared → Selected **if**

$\begin{matrix} A \\ N \\ D \end{matrix}$	Select_PITCH <sub>m-62</sub> ()	T
	Is_This_Side_Active <sub>v-30</sub>	T

Selected → Cleared **if**

$\begin{matrix} A \\ N \\ D \end{matrix}$	Deselect_PITCH <sub>m-62</sub> ()	T
	Is_This_Side_Active <sub>v-30</sub>	T

**Purpose:** This variable maintains the current base state of Pitch Hold mode, i.e., whether it is cleared or selected. Note that if this side of the FGS is not active, it obtains its value from the off side. The normal transitions occur only when this side is active.

### 2.7.2.2 Vertical Speed (VS) Mode

In Vertical Speed mode, the FGS provides pitch guidance commands to hold the aircraft to the Vertical Speed (VS) reference.

<b>Definitions of Values to be Imported</b>
---

---

MACRO
-------

---

#### Select\_VS

---

**Condition:**

A	When_VS_Switch_Pressed_Seen <sub>m-111</sub> ()	T
N	Overspeed_Condition <sub>m-128</sub> ()	F
D	PREVIOUS_STEP(Is_VAPPR_Active <sub>v-87</sub> )	F

**Purpose:** This event defines when Vertical Speed mode is to be selected.

---

MACRO
-------

---

#### Deselect\_VS

---

**Condition:**

		<i>OR</i>			
A	When_VS_Switch_Pressed_Seen <sub>m-111</sub> ()	T	·	·	·
N	When_Nonbasic_Vertical_Mode_Activated <sub>m-60</sub> ()	·	T	·	·
D	When_Pilot_Flying_Transfer <sub>m-29</sub> ()	·	·	T	·
	When(Modes <sub>v-34</sub> =Off)	·	·	·	T

**Purpose:** This event defines when Vertical Speed mode is to be deselected.

Definitions of Values to be Exported
--------------------------------------

---

STATE VARIABLE
----------------

---

**Is\_VS\_Selected**

---

**Parent:** NONE

**Type:** BOOLEAN

**Initial Value:** FALSE

**Classified as:** CONTROLLED

**:=** (VS<sub>v-68</sub> =Selected) **if** TRUE

**Purpose:** Indicates if VS mode is selected.

---

STATE VARIABLE
----------------

---

**Is\_VS\_Active**

---

**Parent:** NONE

**Type:** BOOLEAN

**Initial Value:** FALSE

**Classified as:** CONTROLLED

**:=** (VS<sub>v-68</sub> =Selected) **if** TRUE

**Purpose:** Indicates if VS mode is active.

**Comment:** Even though VS Selected and VS Active are the same value, this is defined to provide a common set of controlled variables across all modes.

---

MACRO
-------

---

### When\_VS\_Activated

---

**Condition:**

A	Select_VS <sub>m-65</sub> ()	T
N	PREVIOUS_STEP(VS <sub>v-68</sub> ) = Selected	F
D	Is_This_Side_Active <sub>v-30</sub>	T

**Purpose:** This signal occurs when Vertical Speed mode is activated.

**Comment:** This event is defined this way to avoid circular dependencies. It would be preferable to define it as When(VS = Selected).

<b>Definitions of Values to be Encapsulated</b>
---

---

STATE VARIABLE
----------------

---

**VS**

---

**Parent:** Modes<sub>v-34</sub>.On

**Type:** {Cleared, Selected}<sub>ty-23</sub>

**Initial Value:** UNDEFINED

**Classified as:** State

**:=** Offside\_VS<sub>v-140</sub> **if** **not** Is\_This\_Side\_Active<sub>v-30</sub>

UNDEFINED → Cleared **if**

$\begin{matrix} A \\ N \\ D \end{matrix}$	Select_VS <sub>m-65</sub> ()	F
	Is_This_Side_Active <sub>v-30</sub>	T

UNDEFINED → Selected **if**

$\begin{matrix} A \\ N \\ D \end{matrix}$	Select_VS <sub>m-65</sub> ()	T
	Is_This_Side_Active <sub>v-30</sub>	T

Cleared → Selected **if**

$\begin{matrix} A \\ N \\ D \end{matrix}$	Select_VS <sub>m-65</sub> ()	T
	Is_This_Side_Active <sub>v-30</sub>	T

Selected → Cleared **if**

$\begin{matrix} A \\ N \\ D \end{matrix}$	Deselect_VS <sub>m-65</sub> ()	T
	Is_This_Side_Active <sub>v-30</sub>	T

**Purpose:** This variable maintains the current base state of Vertical Speed mode, i.e., whether it is cleared or selected. Note that if this side of the FGS is not active, it obtains its value from the off side. The normal transitions occur only when this side is active.

### 2.7.2.3 Flight Level Change (FLC) Mode

In Flight Level Change mode, the FGS provides guidance commands to acquire and track an Indicated Airspeed (IAS) or Mach Reference Airspeed, taking into account the need to climb or descend to bring the aircraft to the PreSelector Altitude (PSA).

<b>Definitions of Values to be Imported</b>
---

---

MACRO
-------

---

#### Select\_FLC

---

**Condition:**

AND	When_FLC_Switch_Pressed_Seen <sub>m-112</sub> ()	OR	T	·
	PREVIOUS_STEP(Is_VAPPR_Active <sub>v-87</sub> )		F	·
	Overspeed_Condition <sub>m-128</sub> ()		·	T
	PREVIOUS_STEP(Is_ALT_Active <sub>v-74</sub> )		·	F
	When_ALT_Activated <sub>m-75</sub> ()		·	F
	PREVIOUS_STEP(Is_ALTSEL_Active <sub>v-80</sub> )		·	F
	When_ALTSEL_Activated <sub>m-81</sub> ()		·	F

**Purpose:** This event defines when Flight Level Change mode is to be selected.

---

 MACRO
 

---

 Deselect\_FLC
 

---

**Condition:**

	When_FLC_Switch_Pressed_Seen <sub>m-112</sub> ()	T	.	.	.	.
	When_VS_Pitch_Wheel_Rotated_Seen <sub>m-110</sub> ()	.	T	.	.	.
<i>A</i> <i>N</i> <i>D</i>	Overspeed_Condition <sub>m-128</sub> ()	F	F	.	.	.
	When_Nonbasic_Vertical_Mode_Activated <sub>m-60</sub> ()	.	.	T	.	.
	When_Pilot_Flying_Transfer <sub>m-29</sub> ()	.	.	.	T	.
	When(Modes <sub>v-34</sub> =Off)	.	.	.	.	T

*OR*

**Purpose:** This event defines when Flight Level Change mode is to be deselected.

---

 Definitions of Values to be Exported
 

---



---

 STATE VARIABLE
 

---

 Is\_FLC\_Selected
 

---

**Parent:** NONE**Type:** BOOLEAN**Initial Value:** FALSE**Classified as:** CONTROLLED**:=** (FLC<sub>v-72</sub> =Selected) **if** TRUE

**Purpose:** Indicates if FLC mode is selected.

---

STATE VARIABLE
----------------

---

### Is\_FLC\_Active

---

**Parent:** NONE

**Type:** BOOLEAN

**Initial Value:** FALSE

**Classified as:** CONTROLLED

**:=** (FLC<sub>v-72</sub> =Selected) **if** TRUE

**Purpose:** Indicates if FLC mode is active.

**Comment:** Even though FLC Selected and FLC Active are the same value, this is defined to provide a common set of controlled variables across all modes.

---

MACRO
-------

---

### When\_FLC\_Activated

---

**Condition:**

A	Select_FLC <sub>m-69</sub> ()	T
N	PREVIOUS STEP(FLC <sub>v-72</sub> ) =Selected	F
D	Is_This_Side_Active <sub>v-30</sub>	T

**Purpose:** This signal occurs when Flight Level Change mode is activated.

**Comment:** This event is defined this way to avoid circular dependencies. It would be preferable to define it as When(FLC = Selected).

<b>Definitions of Values to be Encapsulated</b>
---

---

STATE VARIABLE
----------------

---

**FLC**

---

**Parent:** Modes<sub>v-34</sub>.On

**Type:** {Cleared, Selected}<sub>ty-23</sub>

**Initial Value:** UNDEFINED

**Classified as:** State

**:=** Offside\_FLC<sub>v-140</sub> **if not** Is\_This\_Side\_Active<sub>v-30</sub>

UNDEFINED → Cleared **if**

$\begin{matrix} A \\ N \\ D \end{matrix}$	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="padding: 2px;">Select_FLC<sub>m-69</sub>()</td> <td style="width: 20px;"></td> <td style="text-align: center; vertical-align: middle;">F</td> </tr> <tr> <td style="padding: 2px;">Is_This_Side_Active<sub>v-30</sub></td> <td></td> <td style="text-align: center; vertical-align: middle;">T</td> </tr> </table>	Select_FLC <sub>m-69</sub> ()		F	Is_This_Side_Active <sub>v-30</sub>		T
Select_FLC <sub>m-69</sub> ()		F					
Is_This_Side_Active <sub>v-30</sub>		T					

UNDEFINED → Selected **if**

$\begin{matrix} A \\ N \\ D \end{matrix}$	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="padding: 2px;">Select_FLC<sub>m-69</sub>()</td> <td style="width: 20px;"></td> <td style="text-align: center; vertical-align: middle;">T</td> </tr> <tr> <td style="padding: 2px;">Is_This_Side_Active<sub>v-30</sub></td> <td></td> <td style="text-align: center; vertical-align: middle;">T</td> </tr> </table>	Select_FLC <sub>m-69</sub> ()		T	Is_This_Side_Active <sub>v-30</sub>		T
Select_FLC <sub>m-69</sub> ()		T					
Is_This_Side_Active <sub>v-30</sub>		T					

Cleared → Selected **if**

$\begin{matrix} A \\ N \\ D \end{matrix}$	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="padding: 2px;">Select_FLC<sub>m-69</sub>()</td> <td style="width: 20px;"></td> <td style="text-align: center; vertical-align: middle;">T</td> </tr> <tr> <td style="padding: 2px;">Is_This_Side_Active<sub>v-30</sub></td> <td></td> <td style="text-align: center; vertical-align: middle;">T</td> </tr> </table>	Select_FLC <sub>m-69</sub> ()		T	Is_This_Side_Active <sub>v-30</sub>		T
Select_FLC <sub>m-69</sub> ()		T					
Is_This_Side_Active <sub>v-30</sub>		T					

Selected → Cleared **if**

$\begin{matrix} A \\ N \\ D \end{matrix}$	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="padding: 2px;">Deselect_FLC<sub>m-70</sub>()</td> <td style="width: 20px;"></td> <td style="text-align: center; vertical-align: middle;">T</td> </tr> <tr> <td style="padding: 2px;">Is_This_Side_Active<sub>v-30</sub></td> <td></td> <td style="text-align: center; vertical-align: middle;">T</td> </tr> </table>	Deselect_FLC <sub>m-70</sub> ()		T	Is_This_Side_Active <sub>v-30</sub>		T
Deselect_FLC <sub>m-70</sub> ()		T					
Is_This_Side_Active <sub>v-30</sub>		T					

**Purpose:** This variable maintains the current state of Flight Level Change mode. Note that if this side is not active, it obtains its value from the off side. The normal transitions for Heading Select mode occur only when this side is active.

### 2.7.2.4 Altitude Hold (ALT) Mode

In Altitude Hold mode, the FGS provides pitch guidance commands to to acquire and track the Altitude reference, which is set to the current altitude when the mode is activated or upon a synchronization request by the flight crew.

#### Definitions of Values to be Imported

MACRO

#### Select\_ALT

**Condition:**

A N D	When_ALT_Switch_Pressed_Seen <sub>m-113</sub> ()	OR	T	·
	PREVIOUS_STEP(Is_VAPPR_Active <sub>v-87</sub> )		F	F
	When_ALTSEL_Target_Altitude_Changed_Seen <sub>m-114</sub> ()		·	T
	PREVIOUS_STEP(Is_ALTSEL_Track <sub>v-80</sub> )		·	T

**Purpose:** This event defines when Altitude Hold mode is to be selected.

MACRO

#### Deselect\_ALT

**Condition:**

A N D	When_ALT_Switch_Pressed_Seen <sub>m-113</sub> ()	T	·	·	·	·
	When_VS_Pitch_Wheel_Rotated_Seen <sub>m-110</sub> ()	·	T	·	·	·
	When_Nonbasic_Vertical_Mode_Activated <sub>m-60</sub> ()	·	·	T	·	·
	When_Pilot_Flying_Transfer <sub>m-29</sub> ()	·	·	·	T	·
	When(Modes <sub>v-34</sub> =Off)	·	·	·	·	T

**Purpose:** This event defines when Altitude Hold mode is to be deselected.

Definitions of Values to be Exported
--------------------------------------

---

STATE VARIABLE
----------------

---

**Is\_ALT\_Selected**

---

**Parent:** NONE

**Type:** BOOLEAN

**Initial Value:** FALSE

**Classified as:** CONTROLLED

**:=** (ALT<sub>v-76</sub> =Selected) **if** TRUE

**Purpose:** Indicates if ALT mode is selected.

---

STATE VARIABLE
----------------

---

**Is\_ALT\_Active**

---

**Parent:** NONE

**Type:** BOOLEAN

**Initial Value:** FALSE

**Classified as:** CONTROLLED

**:=** (ALT<sub>v-76</sub> =Selected) **if** TRUE

**Purpose:** Indicates if ALT mode is active.

**Comment:** Even though ALT Selected and ALT Active are the same value, this is defined to provide a common set of controlled variables across all modes.

---

MACRO
-------

---

### When\_ALT\_Activated

---

**Condition:**

A	Select_ALT <sub>m-73</sub> ()	T
N	PREVIOUS_STEP(ALT <sub>v-76</sub> ) =Selected	F
D	Is_This_Side_Active <sub>v-30</sub>	T

**Purpose:** This signal occurs when Altitude Hold mode is activated.

**Comment:** This event is defined this way to avoid circular dependencies. It would be preferable to define it as When(ALT = Selected).

<b>Definitions of Values to be Encapsulated</b>
---

---

STATE VARIABLE
----------------

---

**ALT**

---

**Parent:** Modes<sub>v-34</sub>.On

**Type:** {Cleared, Selected}<sub>ty-23</sub>

**Initial Value:** UNDEFINED

**Classified as:** State

**:=** Offside\_ALT<sub>v-141</sub> **if** **not** Is\_This\_Side\_Active<sub>v-30</sub>

UNDEFINED → Cleared **if**

$\begin{matrix} A \\ N \\ D \end{matrix}$	Select_ALT <sub>m-73</sub> ()	F
	Is_This_Side_Active <sub>v-30</sub>	T

UNDEFINED → Selected **if**

$\begin{matrix} A \\ N \\ D \end{matrix}$	Select_ALT <sub>m-73</sub> ()	T
	Is_This_Side_Active <sub>v-30</sub>	T

Cleared → Selected **if**

$\begin{matrix} A \\ N \\ D \end{matrix}$	Select_ALT <sub>m-73</sub> ()	T
	Is_This_Side_Active <sub>v-30</sub>	T

Selected → Cleared **if**

$\begin{matrix} A \\ N \\ D \end{matrix}$	Deselect_ALT <sub>m-73</sub> ()	T
	Is_This_Side_Active <sub>v-30</sub>	T

**Purpose:** This variable maintains the current base state of Altitude Hold mode, i.e., whether it is cleared or selected. Note that if this side of the FGS is not active, it obtains its value from the off side. The normal transitions occur only when this side is active.

### 2.7.2.5 Altitude Select (ALTSEL) Mode

In Altitude Select mode, the FGS provides guidance commands to capture and track the Preselected Altitude. Altitude Select is an arming mode, i.e., after being selected, it must remain armed for a period of time before it can become active. Altitude Select mode is normally armed, although it is automatically deselected by the activation of Vertical Approach, Go Around, or Altitude Hold mode. While in the armed state, the FGS monitors the aircraft closure rate towards the target altitude and determines the optimum capture point to transition to the capture state. In the capture state, the FGS generates vertical guidance commands to perform a smooth capture of the target altitude. Once it acquires the target altitude, it enters the track state, during which it generates vertical guidance commands to maintain the aircraft at the target altitude.

<b>Definitions of Values to be Imported</b>
---

---

MACRO
-------

---

**Select\_ALTSEL**

---

**Condition:**

A	Is_VAPPR_Active <sub>v-87</sub>	F
N	Is_VGA_Active <sub>v-92</sub>	F
D	Is_ALT_Active <sub>v-74</sub>	F
	Modes <sub>v-34</sub> =On	T

**Purpose:** This event defines when Altitude Select mode is to be selected, i.e., to become armed. Altitude Select mode is armed or active except when Vertical Approach, Go Around, or Altitude Hold are active.

---

 MACRO
 

---

### Capture\_ALTSEL

---

**Condition:** When\_ALTSEL\_Capture\_Cond\_Met\_Seen<sub>m-97</sub>()

**Purpose:** This event defines when Altitude Select mode is to transition from the armed to the capture state.

---

 MACRO
 

---

### Track\_ALTSEL

---

**Condition:** When\_ALTSEL\_Track\_Cond\_Met\_Seen<sub>m-96</sub>()

**Purpose:** This event defines when Altitude Select mode is to transition from the capture to the track state.

---

 MACRO
 

---

### Deselect\_ALTSEL

---

**Condition:**

A N D	Is_VAPPR_Active <sub>v-87</sub>	T	·	·	·
	Is_VGA_Active <sub>v-92</sub>	·	T	·	·
	Is_ALT_Active <sub>v-74</sub>	·	·	T	·
	When(Modes <sub>v-34</sub> =Off)	·	·	·	T

*OR*

**Purpose:** This event defines when Altitude Select mode is to be deselected, i.e., to be cleared regardless of whether it is armed or active. Altitude Select mode is armed or active except when Vertical Approach, Go Around, or Altitude Hold are active.

## MACRO

## Deactivate\_ALTSEL

**Condition:**

		<i>OR</i>			
A N D	When_ALTSEL_Target_Altitude_Changed_Seen <sub>m-114</sub> ()	T	·	·	·
	When_VS_Pitch_Wheel_Rotated_Seen <sub>m-110</sub> ()	·	T	·	·
	When_Pilot_Flying_Transfer <sub>m-29</sub> ()	·	·	T	·
	When_Nonbasic_Vertical_Mode_Activated <sub>m-60</sub> ()	·	·	·	T

**Purpose:** This event defines when Altitude Select mode is to be cleared if it is active, but to remain armed if it is armed.

## Definitions of Values to be Exported

## STATE VARIABLE

## Is\_ALTSEL\_Selected

**Parent:** NONE

**Type:** BOOLEAN

**Initial Value:** FALSE

**Classified as:** CONTROLLED

**:=** (ALTSEL<sub>v-82</sub> =Selected) **if** TRUE

**Purpose:** Indicates if ALTSEL mode is selected.

---

STATE VARIABLE
----------------

---

### Is\_ALTSEL\_Active

---

**Parent:** NONE

**Type:** BOOLEAN

**Initial Value:** FALSE

**Classified as:** CONTROLLED

**:=** (ALTSEL\_Selected<sub>v-83</sub> =Active) **if** TRUE

**Purpose:** Indicates if ALTSEL mode is active.

---

STATE VARIABLE
----------------

---

### Is\_ALTSEL\_Track

---

**Parent:** NONE

**Type:** BOOLEAN

**Initial Value:** FALSE

**Classified as:** CONTROLLED

**:=** (ALTSEL\_Active<sub>v-84</sub> =Track) **if** TRUE

**Purpose:** Indicates if ALTSEL mode is tracking.

---

MACRO
-------

---

### When\_ALTSEL\_Activated

---

**Condition:**

A	Capture_ALTSEL <sub>m-78</sub> ()	T
N	PREVIOUS STEP(ALTSEL_Selected <sub>v-83</sub> ) = Armed	T
D	Is_This_Side_Active <sub>v-30</sub>	T

**Purpose:** This signal occurs when Altitude Select mode is activated, i.e., it enters the capture state.

**Comment:** This event is defined this way to avoid circular dependencies. It would be preferable to define it as When(ALTSEL\_Selected = Active).

<b>Definitions of Values to be Encapsulated</b>
---

---

STATE VARIABLE
----------------

---

**ALTSEL**

---

**Parent:** Modes<sub>v-34</sub>.On

**Type:** {Cleared, Selected}<sub>ty-23</sub>

**Initial Value:** UNDEFINED

**Classified as:** State

**:=** Offside\_ALTSEL<sub>v-141</sub> **if** **not** Is\_This\_Side\_Active<sub>v-30</sub>

UNDEFINED → Cleared **if**

$\begin{matrix} A \\ N \\ D \end{matrix}$	Select_ALTSEL <sub>m-77</sub> ()	F
D	Is_This_Side_Active <sub>v-30</sub>	T

UNDEFINED → Selected **if**

$\begin{matrix} A \\ N \\ D \end{matrix}$	Select_ALTSEL <sub>m-77</sub> ()	T
D	Is_This_Side_Active <sub>v-30</sub>	T

Cleared → Selected **if**

$\begin{matrix} A \\ N \\ D \end{matrix}$	Select_ALTSEL <sub>m-77</sub> ()	T
D	Is_This_Side_Active <sub>v-30</sub>	T

Selected → Cleared **if**

$\begin{matrix} A \\ N \\ D \end{matrix}$	Deselect_ALTSEL <sub>m-78</sub> ()	T
D	Is_This_Side_Active <sub>v-30</sub>	T

**Purpose:** This variable maintains the current base state of Altitude Select mode, i.e., whether it is cleared or selected. Note that if this side of the FGS is not active, it obtains its value from the off side. The normal transitions occur only when this side is active.

## STATE VARIABLE

## ALTSEL\_Selected

**Parent:** Modes<sub>v-34</sub>.On  $\triangleright$  ALTSEL<sub>v-82</sub>.Selected

**Type:** {Armed, Active}<sub>ty-23</sub>

**Initial Value:** UNDEFINED

**Classified as:** State

**:=** Offside\_ALTSEL\_Selected<sub>v-142</sub> **if** **not** Is\_This\_Side\_Active<sub>v-30</sub>

UNDEFINED  $\longrightarrow$  Armed **if** TRUE

Armed  $\longrightarrow$  Active **if**

A	Capture_ALTSEL <sub>m-78</sub> ()	T
D	Is_This_Side_Active <sub>v-30</sub>	T

Active  $\longrightarrow$  Armed **if**

A	Deactivate_ALTSEL <sub>m-79</sub> ()	T
D	Is_This_Side_Active <sub>v-30</sub>	T

**Purpose:** This variable maintains the current state of Altitude Select mode when it is selected, i.e., whether it is armed or active. Note that if this side of the FGS is not active, it obtains its value from the off side. The normal transitions occur only when this side is active.

---

STATE VARIABLE

---



---

**ALTSEL\_Active**


---

**Parent:** Modes<sub>v-34</sub>.On ▷ ALTSEL<sub>v-82</sub>.Selected ▷ ALTSEL\_Selected<sub>v-83</sub>.Active

**Type:** {Capture, Track}<sub>ty-23</sub>

**Initial Value:** UNDEFINED

**Classified as:** State

**:=** Offside\_ALTSEL\_Active<sub>v-142</sub> **if not** Is\_This\_Side\_Active<sub>v-30</sub>

UNDEFINED → Capture **if** TRUE

Capture → Track **if** Track\_ALTSEL<sub>m-78</sub>()

**Purpose:** This variable maintains the current state of Altitude Select mode when it is active, i.e., whether it is captured or tracking. Note that if this side of the FGS is not active, it obtains its value from the off side. The normal transitions occur only when this side is active.

### 2.7.2.6 Vertical Approach (VAPPR) Mode

In Vertical Approach mode, the FGS provides guidance commands to track vertical guidance for ILS precision Glideslope approaches. Vertical Approach is an arming mode, i.e., after being selected, it must remain armed for a period of time before it can become active.

<b>Definitions of Values to be Imported</b>
---

---

MACRO
-------

---

#### Select\_VAPPR

---

**Condition:** When\_APPR\_Switch\_Pressed\_Seen<sub>m-115</sub>()

**Purpose:** This event defines when Vertical Approach mode is to be selected.

---

MACRO
-------

---

#### Activate\_VAPPR

---

**Condition:**

A	When_VAPPR_Track_Cond_Met_Seen <sub>m-99</sub> ()	T
N	Is_LAPPR_Active <sub>v-52</sub>	T
D	Overspeed_Condition <sub>m-128</sub> ()	F

**Purpose:** This event defines when Vertical Approach mode is to transition from the armed to the active state.

---

 MACRO
 

---

### Deselect\_VAPPR

---

**Condition:**

		<i>OR</i>					
A N D	When_APPR_Switch_Pressed_Seen <sub>m-115</sub> ()	T	·	·	·	·	·
	When(Is_LAPPR_Selected <sub>v-52</sub> =FALSE)	·	T	·	·	·	·
	When_Selected_Nav_Source_Changed <sub>v-127</sub>	·	·	T	·	·	·
	When_Selected_Nav_Frequency_Changed <sub>v-127</sub>	·	·	·	T	·	·
	When_Pilot_Flying_Transfer <sub>m-29</sub> ()	·	·	·	·	T	·
	When(Modes <sub>v-34</sub> =Off)	·	·	·	·	·	T

**Purpose:** This event defines when Vertical Approach mode is to be deselected, i.e., to be cleared regardless of whether it is armed or active.

---

 MACRO
 

---

### Deactivate\_VAPPR

---

**Condition:**

When_Nonbasic_Vertical_Mode_Activated <sub>m-60</sub> ()	T
--	---

**Purpose:** This event defines when Vertical Approach mode is to be cleared if it is active, but to remain armed if it is armed.

---

Definitions of Values to be Exported
--------------------------------------

---

STATE VARIABLE
----------------

---

**Is\_VAPPR\_Selected**

---

**Parent:** NONE**Type:** BOOLEAN**Initial Value:** FALSE**Classified as:** CONTROLLED**:=** (VAPPR<sub>v-89</sub> =Selected) **if** TRUE**Purpose:** Indicates if VAPPR mode is selected.

---

STATE VARIABLE
----------------

---

**Is\_VAPPR\_Active**

---

**Parent:** NONE**Type:** BOOLEAN**Initial Value:** FALSE**Classified as:** CONTROLLED**:=** (VAPPR\_Selected<sub>v-90</sub> =Active) **if** TRUE**Purpose:** Indicates if VAPPR mode is active.

---

MACRO
-------

---

### When\_VAPPR\_Activated

---

**Condition:**

A	Activate_VAPPR <sub>m-85</sub> ()	T
N	PREVIOUS_STEP(VAPPR_Selected <sub>v-90</sub> ) = Armed	T
D	Is_This_Side_Active <sub>v-30</sub>	T

**Purpose:** This signal occurs when Vertical Approach mode is activated.

**Comment:** This event is defined this way to avoid circular dependencies. It would be preferable to define it as When(VAPPR\_Selected = Active).

<b>Definitions of Values to be Encapsulated</b>
---

---

STATE VARIABLE
----------------

---

### VAPPR

---

**Parent:** Modes<sub>v-34</sub>.On

**Type:** {Cleared, Selected}<sub>ty-23</sub>

**Initial Value:** UNDEFINED

**Classified as:** State

**:=** Offside\_VAPPR<sub>v-143</sub> **if** **not** Is\_This\_Side\_Active<sub>v-30</sub>

UNDEFINED → Cleared **if**

$\begin{matrix} A \\ N \\ D \end{matrix}$	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="padding: 2px;">Select_VAPPR<sub>m-85</sub>()</td> <td style="width: 20px; text-align: center; vertical-align: middle;">F</td> </tr> <tr> <td style="padding: 2px;">Is_This_Side_Active<sub>v-30</sub></td> <td style="width: 20px; text-align: center; vertical-align: middle;">T</td> </tr> </table>	Select_VAPPR <sub>m-85</sub> ()	F	Is_This_Side_Active <sub>v-30</sub>	T
Select_VAPPR <sub>m-85</sub> ()	F				
Is_This_Side_Active <sub>v-30</sub>	T				

UNDEFINED → Selected **if**

$\begin{matrix} A \\ N \\ D \end{matrix}$	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="padding: 2px;">Select_VAPPR<sub>m-85</sub>()</td> <td style="width: 20px; text-align: center; vertical-align: middle;">T</td> </tr> <tr> <td style="padding: 2px;">Is_This_Side_Active<sub>v-30</sub></td> <td style="width: 20px; text-align: center; vertical-align: middle;">T</td> </tr> </table>	Select_VAPPR <sub>m-85</sub> ()	T	Is_This_Side_Active <sub>v-30</sub>	T
Select_VAPPR <sub>m-85</sub> ()	T				
Is_This_Side_Active <sub>v-30</sub>	T				

Cleared → Selected **if**

$\begin{matrix} A \\ N \\ D \end{matrix}$	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="padding: 2px;">Select_VAPPR<sub>m-85</sub>()</td> <td style="width: 20px; text-align: center; vertical-align: middle;">T</td> </tr> <tr> <td style="padding: 2px;">Is_This_Side_Active<sub>v-30</sub></td> <td style="width: 20px; text-align: center; vertical-align: middle;">T</td> </tr> </table>	Select_VAPPR <sub>m-85</sub> ()	T	Is_This_Side_Active <sub>v-30</sub>	T
Select_VAPPR <sub>m-85</sub> ()	T				
Is_This_Side_Active <sub>v-30</sub>	T				

Selected → Cleared **if**

$\begin{matrix} A \\ N \\ D \end{matrix}$	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="padding: 2px;">Deselect_VAPPR<sub>m-86</sub>()</td> <td style="width: 20px; text-align: center; vertical-align: middle;">T</td> <td style="width: 20px; text-align: center; vertical-align: middle;">OR</td> <td style="width: 20px; text-align: center; vertical-align: middle;">·</td> </tr> <tr> <td style="padding: 2px;">Deactivate_VAPPR<sub>m-86</sub>()</td> <td style="width: 20px; text-align: center; vertical-align: middle;">·</td> <td style="width: 20px; text-align: center; vertical-align: middle;">T</td> <td style="width: 20px; text-align: center; vertical-align: middle;">T</td> </tr> <tr> <td style="padding: 2px;">PREVIOUS_STEP(VAPPR_Selected<sub>v-90</sub>) = Active</td> <td style="width: 20px; text-align: center; vertical-align: middle;">·</td> <td style="width: 20px; text-align: center; vertical-align: middle;">T</td> <td style="width: 20px; text-align: center; vertical-align: middle;">T</td> </tr> <tr> <td style="padding: 2px;">Is_This_Side_Active<sub>v-30</sub></td> <td style="width: 20px; text-align: center; vertical-align: middle;">T</td> <td style="width: 20px; text-align: center; vertical-align: middle;">T</td> <td style="width: 20px; text-align: center; vertical-align: middle;">T</td> </tr> </table>	Deselect_VAPPR <sub>m-86</sub> ()	T	OR	·	Deactivate_VAPPR <sub>m-86</sub> ()	·	T	T	PREVIOUS_STEP(VAPPR_Selected <sub>v-90</sub> ) = Active	·	T	T	Is_This_Side_Active <sub>v-30</sub>	T	T	T
Deselect_VAPPR <sub>m-86</sub> ()	T	OR	·														
Deactivate_VAPPR <sub>m-86</sub> ()	·	T	T														
PREVIOUS_STEP(VAPPR_Selected <sub>v-90</sub> ) = Active	·	T	T														
Is_This_Side_Active <sub>v-30</sub>	T	T	T														

**Purpose:** This variable maintains the current base state of Vertical Approach mode, i.e., whether it is cleared or selected. Note that if this side of the FGS is not active, it obtains its value from the off side. The normal transitions occur only when this side is active.

## STATE VARIABLE

## VAPPR\_Selected

**Parent:** Modes<sub>v-34</sub>.On  $\triangleright$  VAPPR<sub>v-89</sub>.Selected

**Type:** {Armed, Active}<sub>ty-23</sub>

**Initial Value:** UNDEFINED

**Classified as:** State

**:=** Offside\_VAPPR\_Selected<sub>v-143</sub> **if** **not** Is\_This\_Side\_Active<sub>v-30</sub>

UNDEFINED  $\longrightarrow$  Armed **if** TRUE

Armed  $\longrightarrow$  Active **if**

A	Activate_VAPPR <sub>m-85</sub> ()	T
D	Is_This_Side_Active <sub>v-30</sub>	T

**Purpose:** This variable maintains the current state of Vertical Approach mode when it is selected, i.e., whether it is armed or active. Note that if this side of the FGS is not active, it obtains its value from the off side. The normal transitions occur only when this side is active.



Definitions of Values to be Exported
--------------------------------------

---

STATE VARIABLE
----------------

---

**Is\_VGA\_Selected**

---

**Parent:** NONE

**Type:** BOOLEAN

**Initial Value:** FALSE

**Classified as:** CONTROLLED

**:=** (VGA<sub>v-94</sub> =Selected) **if** TRUE

**Purpose:** Indicates if VGA mode is selected.

---

STATE VARIABLE
----------------

---

**Is\_VGA\_Active**

---

**Parent:** NONE

**Type:** BOOLEAN

**Initial Value:** FALSE

**Classified as:** CONTROLLED

**:=** (VGA<sub>v-94</sub> =Selected) **if** TRUE

**Purpose:** Indicates if VGA mode is active.

**Comment:** Even though VGA Selected and VGA Active are the same value, this is defined to provide a common set of controlled variables across all modes.

---

MACRO
-------

---

### When\_VGA\_Activated

---

**Condition:**

A	Select_VGA <sub>m-91</sub> ()	T
N	PREVIOUS_STEP(VGA <sub>v-94</sub> ) = Selected	F
D	Is_This_Side_Active <sub>v-30</sub>	T

**Purpose:** This signal occurs when Vertical Go Around mode is activated.

**Comment:** This event is defined this way to avoid circular dependencies. It would be preferable to define it as When(LGA = Selected).

<b>Definitions of Values to be Encapsulated</b>
---

---

STATE VARIABLE
----------------

---

**VGA**

---

**Parent:** Modes<sub>v-34</sub>.On

**Type:** {Cleared, Selected}<sub>ty-23</sub>

**Initial Value:** UNDEFINED

**Classified as:** State

**:=** Offside\_VGA<sub>v-144</sub> **if** **not** Is\_This\_Side\_Active<sub>v-30</sub>

UNDEFINED → Cleared **if**

$\begin{matrix} A \\ N \\ D \end{matrix}$	Select_VGA <sub>m-91</sub> ()	F
	Is_This_Side_Active <sub>v-30</sub>	T

UNDEFINED → Selected **if**

$\begin{matrix} A \\ N \\ D \end{matrix}$	Select_VGA <sub>m-91</sub> ()	T
	Is_This_Side_Active <sub>v-30</sub>	T

Cleared → Selected **if**

$\begin{matrix} A \\ N \\ D \end{matrix}$	Select_VGA <sub>m-91</sub> ()	T
	Is_This_Side_Active <sub>v-30</sub>	T

Selected → Cleared **if**

$\begin{matrix} A \\ N \\ D \end{matrix}$	Deselect_VGA <sub>m-91</sub> ()	T
	Is_This_Side_Active <sub>v-30</sub>	T

**Purpose:** This variable maintains the current state of Vertical Go Around mode. Note that if this side is not active, it obtains its value from the off side. The normal transitions for Vertical Go Around mode occur only when this side is active.

## 2.8 Flight Control Laws

The Flight Control Laws generate the pitch and roll guidance commands provided to the Flight Director and the Autopilot. Different Flight Control Laws are armed or activated based on the current modes of the FGS. They also provide inputs to the mode logic that determine when transitions are made from the armed to the active state. While the Flight Control Laws are not specified in this model, this component defines the interface between them and the mode logic and defines the events provided by the Flight Control Laws that the mode logic depends upon. As with events provided by the Flight Control Panel, some of these are prioritized so that if multiple events occur at the same time, only the most significant events are seen by the other parts of the specification.

<b>Definitions of Values to be Exported</b>
---

---

MACRO
-------

---

### When\_NAV\_Track\_Cond\_Met

---

**Condition:** Is\_NAV\_Track\_Cond\_Met<sub>iv-100</sub>

**Purpose:** This event indicates when the NAV Track Condition is met.

**Comment:** This is redefined as a macro to simplify verification.

---

MACRO
-------

---

### When\_NAV\_Track\_Cond\_Met\_Seen

---

**Condition:**

A	When_NAV_Track_Cond_Met <sub>m-95</sub> ()	T
D	No_Higher_Event_Than_NAV_Track_Cond_Met <sub>m-96</sub> ()	T

**Purpose:** This event indicates when the NAV Track condition is met and no higher priority event has occurred.

---

 MACRO
 

---

### No\_Higher\_Event\_Than\_NAV\_Track\_Cond\_Met

---

**Condition:**

A	When_FD_Switch_Pressed <sub>m-103</sub> ()	F
D	No_Higher_Event_Than_FD_Switch_Pressed <sub>m-104</sub> ()	T

**Purpose:** This event occurs when no event with a priority higher than pressing the NAV switch has occurred.

---

 MACRO
 

---

### When\_ALTSEL\_Track\_Cond\_Met

---

**Condition:** Is\_ALTSEL\_Track\_Cond\_Met<sub>iv-101</sub>

**Purpose:** This event indicates when the ALTSEL Track Condition is met.

**Comment:** This is redefined as a macro to simplify verification.

---

 MACRO
 

---

### When\_ALTSEL\_Track\_Cond\_Met\_Seen

---

**Condition:**

A	When_ALTSEL_Track_Cond_Met <sub>m-96</sub> ()	T
D	No_Higher_Event_Than_ALTSEL_Track_Cond_Met <sub>m-97</sub> ()	T

**Purpose:** This event indicates when the ALTSEL Track condition is met and no higher priority event has occurred.

---

MACRO

---

### No\_Higher\_Event\_Than\_ALTSEL\_Track\_Cond\_Met

---

**Condition:**

A	When_FD_Switch_Pressed <sub>m-103</sub> ()	F
D	No_Higher_Event_Than_FD_Switch_Pressed <sub>m-104</sub> ()	T

**Purpose:** This event occurs when no event with a priority higher than pressing the FD switch has occurred.

---

MACRO

---

### When\_ALTSEL\_Capture\_Cond\_Met

---

**Condition:** Is\_ALTSEL\_Capture\_Cond\_Met<sub>iv-101</sub>

**Purpose:** This event indicates when the ALTSEL Capture Condition is met.

**Comment:** This is redefined as a macro to simplify verification.

---

MACRO

---

### When\_ALTSEL\_Capture\_Cond\_Met\_Seen

---

**Condition:**

A	When_ALTSEL_Capture_Cond_Met <sub>m-97</sub> ()	T
D	No_Higher_Event_Than_ALTSEL_Capture_Cond_Met <sub>m-98</sub> ()	T

**Purpose:** This event indicates when the ALTSEL Capture condition is met and no higher priority event has occurred.

---

 MACRO
 

---

### No\_Higher\_Event\_Than\_ALTSEL\_Capture\_Cond\_Met

---

**Condition:**

A	When_FD_Switch_Pressed <sub>m-103</sub> ()	F
D	No_Higher_Event_Than_FD_Switch_Pressed <sub>m-104</sub> ()	T

**Purpose:** This event occurs when no event with a priority higher than pressing the FD switch has occurred.

---

 MACRO
 

---

### When\_LAPPR\_Track\_Cond\_Met

---

**Condition:** Is\_LAPPR\_Track\_Cond\_Met<sub>iv-101</sub>

**Purpose:** This event indicates when the LAPPR Track Condition is met.

**Comment:** This is redefined as a macro to simplify verification.

---

 MACRO
 

---

### When\_LAPPR\_Track\_Cond\_Met\_Seen

---

**Condition:**

A	When_LAPPR_Track_Cond_Met <sub>m-98</sub> ()	T
D	No_Higher_Event_Than_LAPPR_Track_Cond_Met <sub>m-99</sub> ()	T

**Purpose:** This event indicates when the LAPPR Track condition is met and no higher priority event has occurred.

---

 MACRO
 

---

### No\_Higher\_Event\_Than\_LAPPR\_Track\_Cond\_Met

---

**Condition:**

A	When_FD_Switch_Pressed <sub>m-103</sub> ()	F
D	No_Higher_Event_Than_FD_Switch_Pressed <sub>m-104</sub> ()	T

**Purpose:** This event occurs when no event with a priority higher than pressing the FD switch has occurred.

---

 MACRO
 

---

### When\_VAPPR\_Track\_Cond\_Met

---

**Condition:** Is\_VAPPR\_Track\_Cond\_Met<sub>iv-102</sub>

**Purpose:** This event indicates when the VAPPR Track Condition is met.

**Comment:** This is redefined as a macro to simplify verification.

---

 MACRO
 

---

### When\_VAPPR\_Track\_Cond\_Met\_Seen

---

**Condition:**

A	When_VAPPR_Track_Cond_Met <sub>m-99</sub> ()	T
D	No_Higher_Event_Than_VAPPR_Track_Cond_Met <sub>m-100</sub> ()	T

**Purpose:** This event indicates when the VAPPR Track condition is met and no higher priority event has occurred.

---

 MACRO
 

---

 No\_Higher\_Event\_Than\_VAPPR\_Track\_Cond\_Met
 

---

**Condition:**

A	When_FD_Switch_Pressed <sub>m-103</sub> ()	F
N	No_Higher_Event_Than_FD_Switch_Pressed <sub>m-104</sub> ()	T
D		

**Purpose:** This event occurs when no event with a priority higher than pressing the FD switch has occurred.

 Definitions of Values to be Encapsulated
 

---

 INPUT VARIABLE
 

---

 Is\_NAV\_Track\_Cond\_Met
 

---

**Type:** BOOLEAN

**Initial Value:** FALSE

**Classified as:** MONITORED

**Purpose:** This input variable becomes true when the lateral navigation (NAV) flight control law transitions from the armed to the track state.

---

INPUT VARIABLE
----------------

---

### Is\_ALTSEL\_Capture\_Cond\_Met

---

**Type:** BOOLEAN

**Initial Value:** FALSE

**Classified as:** MONITORED

**Purpose:** This input variable becomes true when the vertical altitude select flight control law transitions from the armed to the capture state.

---

INPUT VARIABLE
----------------

---

### Is\_ALTSEL\_Track\_Cond\_Met

---

**Type:** BOOLEAN

**Initial Value:** FALSE

**Classified as:** MONITORED

**Purpose:** This input variable becomes true when the vertical altitude select flight control law transitions from the capture to the track state.

---

INPUT VARIABLE
----------------

---

### Is\_LAPPR\_Track\_Cond\_Met

---

**Type:** BOOLEAN

**Initial Value:** FALSE

**Classified as:** MONITORED

**Purpose:** This input variable becomes true when the lateral approach (LAPPR) flight control law transitions from the armed to the track state.

---

INPUT VARIABLE
----------------

---

### Is\_VAPPR\_Track\_Cond\_Met

---

**Type:** BOOLEAN

**Initial Value:** FALSE

**Classified as:** MONITORED

**Purpose:** This input variable becomes true when the vertical approach (VAPPR) flight control law transitions from the armed to the track state.

## 2.9 Flight Control Panel (FCP)

The Flight Control Panel (FCP) is the primary device with which the flight crew interacts with the FGS. This component defines the FGS's view of the FCP. This consists of the definition of the monitored variables representing the switches and dials of the FCP and the controlled variables representing the lamps displayed on the FCP. A few additional inputs, such as the SYNC and AP Disengage switches on the control yokes and the GA switches on the throttles, are routed through the FCP and are defined here. This component also defines FCP specific events, such as the pressing of a switch, used in the rest of the specification. These events are prioritized so that if multiple events occur at the same time, only the most significant events are seen by the other parts of the specification.

<b>Definitions of Values to be Exported</b>
---

---

MACRO
-------

---

### When\_FD\_Switch\_Pressed

---

**Condition:** When(FD\_Switch<sub>iv-118</sub> =ON)

**Purpose:** This event indicates when the FD switch associated with this FGS is pressed.

**Comment:** This is redefined as a macro to simplify verification.

---

MACRO
-------

---

### When\_FD\_Switch\_Pressed\_Seen

---

**Condition:**

A	When_FD_Switch_Pressed <sub>m-103</sub> ()	T
D	No_Higher_Event_Than_FD_Switch_Pressed <sub>m-104</sub> ()	T

**Purpose:** This event occurs when the FD Switch is pressed and no higher priority event has occurred.

---

 MACRO
 

---

### No\_Higher\_Event\_Than\_FD\_Switch\_Pressed

---

**Condition:**

A	When_Transfer_Switch_Pressed <sub>m-104</sub> ()	F
D	No_Higher_Event_Than_Transfer_Switch_Pressed <sub>m-105</sub> ()	T

**Purpose:** This event occurs when no event with a priority higher than pressing the FD switch has occurred.

---

 MACRO
 

---

### When\_Transfer\_Switch\_Pressed

---

**Condition:** When(Transfer\_Switch<sub>iv-125</sub> =ON)

**Purpose:** This event indicates when the TRANSFER switch ais pressed.

**Comment:** This is redefined as a macro to simplify verification.

---

 MACRO
 

---

### When\_Transfer\_Switch\_Pressed\_Seen

---

**Condition:**

A	When_Transfer_Switch_Pressed <sub>m-104</sub> ()	T
D	No_Higher_Event_Than_Transfer_Switch_Pressed <sub>m-105</sub> ()	T

**Purpose:** This event indicates when the Transfer switch is pressed and no higher priority event has occurred.

---

 MACRO
 

---

### No\_Higher\_Event\_Than\_Transfer\_Switch\_Pressed

---

**Condition:**

A N D	When_AP_Engage_Switch_Pressed <sub>m-105</sub> ()	F
	No_Higher_Event_Than_AP_Engage_Switch_Pressed <sub>m-106</sub> ()	T
	When_VS_Pitch_Wheel_Rotated <sub>m-109</sub> ()	F
	No_Higher_Event_Than_VS_Pitch_Wheel_Rotated <sub>m-110</sub> ()	T
	When_NAV_Switch_Pressed <sub>m-107</sub> ()	F
	No_Higher_Event_Than_NAV_Switch_Pressed <sub>m-108</sub> ()	T

**Purpose:** This event occurs when no event with a priority higher than pressing the Transfer switch has occurred.

---

 MACRO
 

---

### When\_AP\_Engage\_Switch\_Pressed

---

**Condition:** When(AP\_Engage\_Switch<sub>iv-126</sub> =ON)

**Purpose:** This event indicates when the AP Engaged switch is pressed.

**Comment:** This is redefined as a macro to simplify verification.

---

 MACRO
 

---

### When\_AP\_Engage\_Switch\_Pressed\_Seen

---

**Condition:**

A	When_AP_Engage_Switch_Pressed <sub>m-105</sub> ()	T
D	No_Higher_Event_Than_AP_Engage_Switch_Pressed <sub>m-106</sub> ()	T

**Purpose:** This event indicates when the AP Engage switch is pressed and no higher priority event has occurred.

---

 MACRO
 

---

### No\_Higher\_Event\_Than\_AP\_Engage\_Switch\_Pressed

---

**Condition:**

A	When_AP_Disconnect_Switch_Pressed <sub>m-106</sub> ()	F
D	No_Higher_Event_Than_AP_Disconnect_Switch_Pressed <sub>m-107</sub> ()	T

**Purpose:** This event occurs when no event with a priority higher than pressing the AP Engage switch has occurred.

---

 MACRO
 

---

### When\_AP\_Disconnect\_Switch\_Pressed

---

**Condition:** When(AP\_Disconnect\_Switch<sub>iv-126</sub> =ON)

**Purpose:** This event indicates when the AP Disconnect switch is pressed.

**Comment:** This is redefined as a macro to simplify verification.

---

MACRO

---

### When\_AP\_Disconnect\_Switch\_Pressed\_Seen

---

**Condition:**

A	When_AP_Disconnect_Switch_Pressed <sub>m-106</sub> ()	T
D	No_Higher_Event_Than_AP_Disconnect_Switch_Pressed <sub>m-107</sub> ()	T

**Purpose:** This event indicates when the AP Disconnect switch is pressed and no higher priority event has occurred.

---

MACRO

---

### No\_Higher\_Event\_Than\_AP\_Disconnect\_Switch\_Pressed

---

**Condition:** TRUE

**Purpose:** This event occurs when no event with a priority higher than pressing the AP Disconnect switch has occurred.

Lateral Switches

---

MACRO

---

### When\_NAV\_Switch\_Pressed

---

**Condition:** When(NAV\_Switch<sub>iv-119</sub> =ON)

**Purpose:** This event indicates when the NAV switch is pressed.

**Comment:** This is redefined as a macro to simplify verification.

---

 MACRO
 

---

### When\_NAV\_Switch\_Pressed\_Seen

---

**Condition:**

A	When_NAV_Switch_Pressed <sub>m-107</sub> ()	T
D	No_Higher_Event_Than_NAV_Switch_Pressed <sub>m-108</sub> ()	T

**Purpose:** This event indicates when the NAV Switch is pressed and no higher priority event has occurred.

---

 MACRO
 

---

### No\_Higher\_Event\_Than\_NAV\_Switch\_Pressed

---

**Condition:**

A	When_HDG_Switch_Pressed <sub>m-108</sub> ()	F
D	No_Higher_Event_Than_HDG_Switch_Pressed <sub>m-109</sub> ()	T

**Purpose:** This event occurs when no event with a priority higher than pressing the NAV switch has occurred.

---

 MACRO
 

---

### When\_HDG\_Switch\_Pressed

---

**Condition:** When(HDG\_Switch<sub>iV-118</sub> =ON)

**Purpose:** This event indicates when the HDG switch is pressed.

**Comment:** This is redefined as a macro to simplify verification.

---

 MACRO
 

---

### When\_HDG\_Switch\_Pressed\_Seen

---

**Condition:**

A	When_HDG_Switch_Pressed <sub>m-108</sub> ()	T
D	No_Higher_Event_Than_HDG_Switch_Pressed <sub>m-109</sub> ()	T

**Purpose:** This event indicates when the NAV Switch is pressed and no higher priority event has occurred.

---

 MACRO
 

---

### No\_Higher\_Event\_Than\_HDG\_Switch\_Pressed

---

**Condition:**

A	When_APPR_Switch_Pressed <sub>m-114</sub> ()	F
D	No_Higher_Event_Than_APPR_Switch_Pressed <sub>m-115</sub> ()	T

**Purpose:** This event occurs when no event with a priority higher than pressing the HDG switch has occurred.

 Vertical Switches
 

---

 MACRO
 

---

### When\_VS\_Pitch\_Wheel\_Rotated

---

**Condition:** When(VS\_Pitch\_Wheel\_In\_Motion<sub>iv-125</sub>)

**Purpose:** This event indicates when the VS Pitch Wheel is rotated.

---

 MACRO
 

---

### When\_VS\_Pitch\_Wheel\_Rotated\_Seen

---

**Condition:**

A	When_VS_Pitch_Wheel_Rotated <sub>m-109</sub> ()	T
D	No_Higher_Event_Than_VS_Pitch_Wheel_Rotated <sub>m-110</sub> ()	T

**Purpose:** This event indicates when the VS Pitch Wheel is rotated and no higher priority event has occurred.

---

 MACRO
 

---

### No\_Higher\_Event\_Than\_VS\_Pitch\_Wheel\_Rotated

---

**Condition:**

A	When_VS_Switch_Pressed <sub>m-110</sub> ()	F
D	No_Higher_Event_Than_VS_Switch_Pressed <sub>m-111</sub> ()	T

**Purpose:** This event occurs when no event with a priority higher than rotating the VS Pitch Wheel has occurred.

---

 MACRO
 

---

### When\_VS\_Switch\_Pressed

---

**Condition:** When(VS\_Switch<sub>iv-120</sub> =ON)

**Purpose:** This event indicates when the VS switch is pressed.

**Comment:** This is redefined as a macro to simplify verification.

---

MACRO

---

### When\_VS\_Switch\_Pressed\_Seen

---

**Condition:**

A	When_VS_Switch_Pressed <sub>m-110</sub> ()	T
D	No_Higher_Event_Than_VS_Switch_Pressed <sub>m-111</sub> ()	T

**Purpose:** This event indicates when the VS switch pressed and no higher priority event has occurred.

---

MACRO

---

### No\_Higher\_Event\_Than\_VS\_Switch\_Pressed

---

**Condition:**

A	When_FLC_Switch_Pressed <sub>m-111</sub> ()	F
D	No_Higher_Event_Than_FLC_Switch_Pressed <sub>m-112</sub> ()	T

**Purpose:** This event occurs when no event with a priority higher than pressing the VS switch has occurred.

---

MACRO

---

### When\_FLC\_Switch\_Pressed

---

**Condition:** When(FLC\_Switch<sub>iv-122</sub> =ON)

**Purpose:** This event indicates when the FLC switch is pressed.

**Comment:** This is redefined as a macro to simplify verification.

---

 MACRO
 

---

### When\_FLC\_Switch\_Pressed\_Seen

---

**Condition:**

A	When_FLC_Switch_Pressed <sub>m-111</sub> ()	T
D	No_Higher_Event_Than_FLC_Switch_Pressed <sub>m-112</sub> ()	T

**Purpose:** This event indicates when the FLC switch pressed and no higher priority event has occurred.

---

 MACRO
 

---

### No\_Higher\_Event\_Than\_FLC\_Switch\_Pressed

---

**Condition:**

A	When_ALT_Switch_Pressed <sub>m-112</sub> ()	F
D	No_Higher_Event_Than_ALT_Switch_Pressed <sub>m-113</sub> ()	T

**Purpose:** This event occurs when no event with a priority higher than pressing the FLC switch has occurred.

---

 MACRO
 

---

### When\_ALT\_Switch\_Pressed

---

**Condition:** When(ALT\_Switch<sub>i<sub>v</sub>-121</sub> =ON)

**Purpose:** This event indicates when the ALT switch is pressed.

**Comment:** This is redefined as a macro to simplify verification.

---

 MACRO
 

---

### When\_ALT\_Switch\_Pressed\_Seen

---

**Condition:**

A	When_ALT_Switch_Pressed <sub>m-112</sub> ()	T
D	No_Higher_Event_Than_ALT_Switch_Pressed <sub>m-113</sub> ()	T

**Purpose:** This event indicates when the ALT switch pressed and no higher priority event has occurred.

---

 MACRO
 

---

### No\_Higher\_Event\_Than\_ALT\_Switch\_Pressed

---

**Condition:**

A	When_ALTSEL_Target_Altitude_Changed <sub>m-113</sub> ()	F
D	No_Higher_Event_Than_ALTSEL_Target_Altitude_Changed <sub>m-114</sub> ()	T

**Purpose:** This event occurs when no event with a priority higher than pressing the ALT switch has occurred.

---

 MACRO
 

---

### When\_ALTSEL\_Target\_Altitude\_Changed

---

**Condition:**

A	ALTSEL_Target_Altitude_Changed <sub>iv-129</sub> =UNDEFINED	F
D	ALTSEL_Target_Altitude_Changed <sub>iv-129</sub>	T

**Purpose:** This event indicates when the ALTSEL target altitude is changed.

---

 MACRO
 

---

### When\_ALTSEL\_Target\_Altitude\_Changed\_Seen

---

**Condition:**

A	When_ALTSEL_Target_Altitude_Changed <sub>m-113</sub> ()	T
D	No_Higher_Event_Than_ALTSEL_Target_Altitude_Changed <sub>m-114</sub> ()	T

**Purpose:** This event indicates when the ALTSEL target altitude is changed and no higher priority event has occurred.

---

 MACRO
 

---

### No\_Higher\_Event\_Than\_ALTSEL\_Target\_Altitude\_Changed

---

**Condition:**

A	When_APPR_Switch_Pressed <sub>m-114</sub> ()	F
D	No_Higher_Event_Than_APPR_Switch_Pressed <sub>m-115</sub> ()	T

**Purpose:** This event occurs when no event with a priority higher than rotating the changing of the ALTSEL target altitude has occurred.

 Lateral and Vertical Events
 

---

 MACRO
 

---

### When\_APPR\_Switch\_Pressed

---

**Condition:** When(APP<sub>R</sub>Switch<sub>i<sub>v</sub>-123</sub> =ON)

**Purpose:** This event indicates when the APPR switch is pressed.

**Comment:** This is redefined as a macro to simplify verification.

---

MACRO

---

### When\_APPR\_Switch\_Pressed\_Seen

---

**Condition:**

$\begin{matrix} A \\ N \\ D \end{matrix}$	When_APPR_Switch_Pressed <sub>m-114</sub> ()	T
	No_Higher_Event_Than_APPR_Switch_Pressed <sub>m-115</sub> ()	T

**Purpose:** This event indicates when the APPR switch pressed and no higher priority event has occurred.

---

MACRO

---

### No\_Higher\_Event\_Than\_APPR\_Switch\_Pressed

---

**Condition:**

$\begin{matrix} A \\ N \\ D \end{matrix}$	When_GA_Switch_Pressed <sub>m-115</sub> ()	F
	No_Higher_Event_Than_GA_Switch_Pressed <sub>m-116</sub> ()	T

**Purpose:** This event occurs when no event with a priority higher than pressing the APPR switch has occurred.

---

MACRO

---

### When\_GA\_Switch\_Pressed

---

**Condition:** When(GA\_Switch<sub>iv-124</sub> =ON)

**Purpose:** This event indicates when the GA switch is pressed.

**Comment:** This is redefined as a macro to simplify verification.

---

 MACRO
 

---

### When\_GA\_Switch\_Pressed\_Seen

---

**Condition:**

$\begin{matrix} A \\ N \end{matrix}$	When_GA_Switch_Pressed <sub>m-115</sub> ()	$\begin{matrix} \boxed{T} \end{matrix}$
$\begin{matrix} D \end{matrix}$	No_Higher_Event_Than_GA_Switch_Pressed <sub>m-116</sub> ()	$\begin{matrix} \boxed{T} \end{matrix}$

**Purpose:** This event indicates when the GA switch pressed and no higher priority event has occurred.

---

 MACRO
 

---

### No\_Higher\_Event\_Than\_GA\_Switch\_Pressed

---

**Condition:**

$\begin{matrix} A \\ N \end{matrix}$	When_SYNC_Switch_Pressed <sub>m-116</sub> ()	$\begin{matrix} \boxed{F} \end{matrix}$
$\begin{matrix} D \end{matrix}$	No_Higher_Event_Than_SYNC_Switch_Pressed <sub>m-117</sub> ()	$\begin{matrix} \boxed{T} \end{matrix}$

**Purpose:** This event occurs when no event with a priority higher than pressing the GA switch has occurred.

---

 MACRO
 

---

### When\_SYNC\_Switch\_Pressed

---

**Condition:** When(SYNC\_Switch<sub>iv-125</sub> =ON)

**Purpose:** This event indicates when the SYNC switch is pressed.

**Comment:** This is redefined as a macro to simplify verification.

---

 MACRO
 

---

### When\_SYNC\_Switch\_Pressed\_Seen

---

**Condition:**

A	When_SYNC_Switch_Pressed <sub>m-116</sub> ()	T
D	No_Higher_Event_Than_SYNC_Switch_Pressed <sub>m-117</sub> ()	T

**Purpose:** This event indicates when the SYNC switch pressed and no higher priority event has occurred.

---

 MACRO
 

---

### No\_Higher\_Event\_Than\_SYNC\_Switch\_Pressed

---

**Condition:** TRUE

**Purpose:** This event occurs when no event with a priority higher than pressing the GA switch has occurred.

<b>Definitions of Values to be Encapsulated</b>
---

---

 TYPE
 

---

**Switch**

---

**Possible Values:** OFF, ON

---

TYPE
------

---

**Lamp**

---

**Possible Values:** OFF, ON

---

INPUT VARIABLE
----------------

---

**FD\_Switch**

---

**Type:** {OFF, ON}<sub>ty-117</sub>

**Initial Value:** UNDEFINED

**Classified as:** MONITORED

**Purpose:** Holds the last sensed position of the FD switch associated with this FGS.

---

INPUT VARIABLE
----------------

---

**HDG\_Switch**

---

**Type:** {OFF, ON}<sub>ty-117</sub>

**Initial Value:** UNDEFINED

**Classified as:** MONITORED

**Purpose:** Holds the last sensed position of the HDG switch.

---

STATE VARIABLE
----------------

---

### HDG\_Lamp

---

**Parent:** NONE

**Type:** {OFF, ON}<sub>ty-118</sub>

**Initial Value:** OFF

**Classified as:** CONTROLLED

**:=** ON if Is\_HDG\_Selected<sub>v-41</sub>

**:=** OFF if **not** Is\_HDG\_Selected<sub>v-41</sub>

**Purpose:** Indicates if the HDG switch lamp on the FCP should be on or off.

---

INPUT VARIABLE
----------------

---

### NAV\_Switch

---

**Type:** {OFF, ON}<sub>ty-117</sub>

**Initial Value:** UNDEFINED

**Classified as:** MONITORED

**Purpose:** Holds the last sensed position of the NAV switch.

---

STATE VARIABLE
----------------

---

### NAV\_Lamp

---

**Parent:** NONE

**Type:** {OFF, ON}<sub>ty-118</sub>

**Initial Value:** OFF

**Classified as:** CONTROLLED

**:= ON if** Is\_NAV\_Selected<sub>v-46</sub>

**:= OFF if not** Is\_NAV\_Selected<sub>v-46</sub>

**Purpose:** Indicates if the NAV switch lamp should be on or off.

---

INPUT VARIABLE
----------------

---

### VS\_Switch

---

**Type:** {OFF, ON}<sub>ty-117</sub>

**Initial Value:** UNDEFINED

**Classified as:** MONITORED

**Purpose:** Holds the last sensed position of the VS switch.

---

STATE VARIABLE
----------------

---

### VS\_Lamp

---

**Parent:** NONE

**Type:** {OFF, ON}<sub>ty-118</sub>

**Initial Value:** OFF

**Classified as:** CONTROLLED

**:= ON if** Is\_VS\_Selected<sub>v-66</sub>

**:= OFF if not** Is\_VS\_Selected<sub>v-66</sub>

**Purpose:** Indicates if the VS switch lamp should be on or off.

---

INPUT VARIABLE
----------------

---

### ALT\_Switch

---

**Type:** {OFF, ON}<sub>ty-117</sub>

**Initial Value:** UNDEFINED

**Classified as:** MONITORED

**Purpose:** Holds the last sensed position of the ALT switch.

---

STATE VARIABLE
----------------

---

**ALT\_Lamp**

---

**Parent:** NONE

**Type:** {OFF, ON}<sub>ty-118</sub>

**Initial Value:** OFF

**Classified as:** CONTROLLED

**:=** ON if `Is_ALT_Selectedv-74`

**:=** OFF if `not Is_ALT_Selectedv-74`

**Purpose:** Indicates if the ALT switch lamp should be on or off.

---

INPUT VARIABLE
----------------

---

**FLC\_Switch**

---

**Type:** {OFF, ON}<sub>ty-117</sub>

**Initial Value:** UNDEFINED

**Classified as:** MONITORED

**Purpose:** Holds the last sensed position of the FLC switch.

---

STATE VARIABLE
----------------

---

### FLC\_Lamp

---

**Parent:** NONE

**Type:** {OFF, ON}<sub>ty-118</sub>

**Initial Value:** OFF

**Classified as:** CONTROLLED

**:= ON if** Is\_FLC\_Selected<sub>v-70</sub>

**:= OFF if not** Is\_FLC\_Selected<sub>v-70</sub>

**Purpose:** Indicates if the FLC switch lamp should be on or off.

---

INPUT VARIABLE
----------------

---

### APPR\_Switch

---

**Type:** {OFF, ON}<sub>ty-117</sub>

**Initial Value:** UNDEFINED

**Classified as:** MONITORED

**Purpose:** Holds the last sensed position of the APPR switch.

## STATE VARIABLE

## APPR\_Lamp

**Parent:** NONE**Type:** {OFF, ON}<sub>ty-118</sub>**Initial Value:** OFF**Classified as:** CONTROLLED**:= ON if**

$\begin{matrix} A \\ N \\ D \end{matrix}$	Is_LAPPR_Selected <sub>v-52</sub>	<i>OR</i> $\begin{matrix} \overline{T} & \cdot \\ \cdot & T \end{matrix}$
	Is_VAPPR_Selected <sub>v-87</sub>	

**:= OFF if**

$\begin{matrix} A \\ N \\ D \end{matrix}$	Is_LAPPR_Selected <sub>v-52</sub>	$\begin{matrix} F \\ F \end{matrix}$
	Is_VAPPR_Selected <sub>v-87</sub>	

**Purpose:** Indicates if the APPR switch lamp should be on or off.

## INPUT VARIABLE

## GA\_Switch

**Type:** {OFF, ON}<sub>ty-117</sub>**Initial Value:** UNDEFINED**Classified as:** MONITORED**Purpose:** Holds the last sensed position of the GA switch.

---

INPUT VARIABLE
----------------

---

### SYNC\_Switch

---

**Type:** {OFF, ON}<sub>ty-117</sub>

**Initial Value:** UNDEFINED

**Classified as:** MONITORED

**Purpose:** Holds the last sensed position of the SYNC switch.

---

INPUT VARIABLE
----------------

---

### Transfer\_Switch

---

**Type:** {OFF, ON}<sub>ty-117</sub>

**Initial Value:** UNDEFINED

**Classified as:** MONITORED

**Purpose:** Holds the last sensed position of the TRANSFER switch.

---

INPUT VARIABLE
----------------

---

### VS\_Pitch\_Wheel\_In\_Motion

---

**Type:** BOOLEAN

**Initial Value:** UNDEFINED

**Classified as:** MONITORED

**Purpose:** Holds the last sensed position of the VS/Pitch wheel.

---

INPUT VARIABLE
----------------

---

### AP\_Engage\_Switch

---

**Type:** {OFF, ON}<sub>ty-117</sub>

**Initial Value:** UNDEFINED

**Classified as:** MONITORED

**Purpose:** Holds the last sensed position of the AP Engage switch.

---

STATE VARIABLE
----------------

---

### AP\_Lamp

---

**Parent:** NONE

**Type:** {OFF, ON}<sub>ty-118</sub>

**Initial Value:** OFF

**Classified as:** CONTROLLED

**:= ON if** Is\_AP\_Engaged<sub>v-131</sub>

**:= OFF if not** Is\_AP\_Engaged<sub>v-131</sub>

**Purpose:** Indicates if the AP switch lamp should be on or off.

---

INPUT VARIABLE
----------------

---

### AP\_Disconnect\_Switch

---

**Type:** {OFF, ON}<sub>ty-117</sub>

**Initial Value:** UNDEFINED

**Classified as:** MONITORED

**Purpose:** Holds the last sensed position of the AP Disconnect switch.

## 2.10 Navigation Sources

The FGS receives navigation information from several sources, including VHF Omni-Directional Range (VOR) for lateral navigation, Localizer (LOC) for precision lateral approach, and Glideslope (GS) for precision vertical approach. This component defines the FGS's view of its navigation sources. This includes definition of the types of sources available, the currently selected source, and the information available from that source.

<b>Definitions of Values to be Exported</b>
---

---

STATE VARIABLE
----------------

---

### When\_Selected\_Nav\_Source\_Changed

---

**Parent:** NONE

**Type:** BOOLEAN

**Initial Value:** FALSE

**Classified as:** MONITORED

**Purpose:** This event occurs when a new navigation source is selected.

---

STATE VARIABLE
----------------

---

### When\_Selected\_Nav\_Frequency\_Changed

---

**Parent:** NONE

**Type:** BOOLEAN

**Initial Value:** FALSE

**Classified as:** MONITORED

**Purpose:** This event occurs when the frequency of the selected navigation source is changed.

## 2.11 Air Data System

The Air Data System provides information about the aircraft state sensed from the surrounding air mass such as whether an overspeed condition exists. This component defines the FGS's view of the Air Data System.

<b>Definitions of Values to be Exported</b>
---

---

MACRO
-------

---

### Overspeed\_Condition

---

**Condition:**

A	Overspeed <sub>iv-128</sub> ≠ UNDEFINED	T
N		
D	Overspeed <sub>iv-128</sub> = TRUE	T

<b>Definitions of Values to be Encapsulated</b>
---

---

INPUT VARIABLE
----------------

---

### Overspeed

---

**Type:** BOOLEAN

**Initial Value:** UNDEFINED

**Classified as:** MONITORED

**Purpose:** This state variable indicates if an overspeed condition exists.

## 2.12 References

References refer to target values used by several of the modes, such as the Preselected Altitude. These may be set by the flight crew through the Flight Control Panel, or set to the current aircraft value on entry to a mode or a synchronization request. The references may be physically maintained by a variety of systems. This component defines the FGS's view of the references. Details of how these values are obtained from other systems are encapsulated within this component.

<b>Definitions of Values to be Exported</b>
---

---

INPUT VARIABLE
----------------

---

**ALTSEL\_Target\_Altitude\_Changed**

---

**Type:** BOOLEAN

**Initial Value:** TRUE

**Classified as:** MONITORED

**Purpose:** This input variable holds the most recent information on whether the ALTSEL target altitude (either the PSA or FPTA) has been changed.

## 2.13 Autopilot (AP)

The Autopilot (AP) commands movement of the control surfaces based on the pitch and roll commands generated by the FGS. This component defines the FGS's view of the AP.

**Definitions of Values to be Imported**

MACRO

**When\_Engage\_AP**

**Condition:** When\_AP\_Engage\_Switch\_Pressed\_Seen<sub>m-106</sub>()

**Purpose:** This event defines when the AP is to be engaged.

MACRO

**When\_Disengage\_AP**

**Condition:**

A	When_AP_Engage_Switch_Pressed_Seen <sub>m-106</sub> ()	T	·	·
N	When_AP_Disconnect_Switch_Pressed_Seen <sub>m-107</sub> ()	·	T	·
D	When_GA_Switch_Pressed_Seen <sub>m-116</sub> ()	·	·	T

*OR*

**Purpose:** This event defines when the AP is to be disengaged.

---

**Definitions of Values to be Exported**

---

---

**STATE VARIABLE**

---

**Is\_AP\_Engaged**

---

**Parent:** NONE**Type:** BOOLEAN**Initial Value:** FALSE**Classified as:** CONTROLLED**:=** (AP<sub>v-132</sub> =Engaged) **if** TRUE**Purpose:** Indicates if the AP is engaged.

---

**Definitions of Values to be Encapsulated**

---

---

**TYPE**

---

**AP\_State**

---

**Possible Values:** Disengaged, Engaged

---

STATE VARIABLE
----------------

---

**AP**

---

**Parent:** NONE**Type:** {Disengaged, Engaged}<sub>ty-131</sub>**Initial Value:** Disengaged**Classified as:** State

Disengaged
------------

 $\longrightarrow$ 

Engaged
---------

**if** When\_Engage\_AP<sub>m-130</sub>()

Engaged
---------

 $\longrightarrow$ 

Disengaged
------------

**if** When\_Disengage\_AP<sub>m-130</sub>()**Purpose:** This variable maintains the current state of the autopilot.

## 2.14 Offside FGS

The FGS has two physical sides, or channels, on the left and right sides of the aircraft. These provide redundant implementations that communicate with each other over a cross-channel bus. This component defines this side of the FGS's view of the other side.

<b>Definitions of Values to be Exported</b>
---

STATE VARIABLE
----------------

---

**Offside\_FD**

---

**Parent:** NONE

**Type:** {Off, On}<sub>ty-23</sub>

**Initial Value:** Off

**Classified as:** MONITORED

**:= Off if not** Offside\_FD\_On<sub>iv-145</sub>

**:= On if** Offside\_FD\_On<sub>iv-145</sub>

**Purpose:** This variable holds the most recent information on the state of the FD on the offside FD.

---

STATE VARIABLE
----------------

---

### Is\_Offside\_LAPPR\_Active

---

**Parent:** NONE

**Type:** BOOLEAN

**Initial Value:** FALSE

**Classified as:** MONITORED

**:=** Offside\_LAPPR\_Selected<sub>v-138</sub> =Active **if** TRUE

**Purpose:** Indicates if offside LAPPR mode is active.

---

STATE VARIABLE
----------------

---

### Is\_Offside\_VAPPR\_Active

---

**Parent:** NONE

**Type:** BOOLEAN

**Initial Value:** FALSE

**Classified as:** MONITORED

**:=** Offside\_VAPPR\_Selected<sub>v-143</sub> =Active **if** TRUE

**Purpose:** Indicates if offside VAPPR mode is active.

---

STATE VARIABLE
----------------

---

### Is\_Offside\_VGA\_Active

---

**Parent:** NONE

**Type:** BOOLEAN

**Initial Value:** FALSE

**Classified as:** MONITORED

**:=** Offside\_VGA<sub>v-144</sub> =Selected if TRUE

**Purpose:** Indicates if offside VGA mode is active.

---

STATE VARIABLE
----------------

---

### Offside\_Modes

---

**Parent:** NONE

**Type:** {Off, On}<sub>ty-23</sub>

**Initial Value:** Off

**Classified as:** MONITORED

**:=** Off if **not** Offside\_Modes\_On<sub>iv-145</sub>

**:=** On if Offside\_Modes\_On<sub>iv-145</sub>

**Purpose:** This variable holds the most recent information on the state of the mode annunciations on the offside FGS.

---

STATE VARIABLE
----------------

---

### Offside\_ROLL

---

**Parent:** Offside\_Modes<sub>v-135</sub>.On

**Type:** {Cleared, Selected}<sub>ty-23</sub>

**Initial Value:** UNDEFINED

**Classified as:** MONITORED

**:= Cleared if not** Offside\_Roll\_Selected<sub>iv-145</sub>

**:= Selected if** Offside\_Roll\_Selected<sub>iv-145</sub>

**Purpose:** This variable holds the most recent information on the base state of the lateral ROLL mode of the offside FGS.

---

STATE VARIABLE
----------------

---

### Offside\_HDG

---

**Parent:** Offside\_Modes<sub>v-135</sub>.On

**Type:** {Cleared, Selected}<sub>ty-23</sub>

**Initial Value:** UNDEFINED

**Classified as:** MONITORED

**:= Cleared if not** Offside\_Hdg\_Selected<sub>iv-146</sub>

**:= Selected if** Offside\_Hdg\_Selected<sub>iv-146</sub>

**Purpose:** This variable holds the most recent information on the base state of the lateral HDG mode of the offside FGS.

## STATE VARIABLE

## Offside\_NAV

**Parent:** Offside\_Modes<sub>v-135</sub>.On

**Type:** {Cleared, Selected}<sub>ty-23</sub>

**Initial Value:** UNDEFINED

**Classified as:** MONITORED

**:= Cleared if not** Offside\_Nav\_Selected<sub>iv-146</sub>

**:= Selected if** Offside\_Nav\_Selected<sub>iv-146</sub>

**Purpose:** This variable holds the most recent information on the base state of the lateral NAV mode of the offside FGS.

## STATE VARIABLE

## Offside\_NAV\_Selected

**Parent:** Offside\_Modes<sub>v-135</sub>.On ▷ Offside\_NAV<sub>v-137</sub>.Selected

**Type:** {Armed, Active}<sub>ty-23</sub>

**Initial Value:** UNDEFINED

**Classified as:** MONITORED

**:= Armed if not** Offside\_Nav\_Active<sub>iv-146</sub>

**:= Active if** Offside\_Nav\_Active<sub>iv-146</sub>

**Purpose:** This variable holds the most recent information on the selected state of the lateral NAV mode of the offside FGS.

STATE VARIABLE

**Offside\_LAPPR****Parent:** Offside\_Modes<sub>v-135</sub>.On**Type:** {Cleared, Selected}<sub>ty-23</sub>**Initial Value:** UNDEFINED**Classified as:** MONITORED**:=** Cleared **if** **not** Offside\_Lappr\_Selected<sub>iv-147</sub>**:=** Selected **if** Offside\_Lappr\_Selected<sub>iv-147</sub>

**Purpose:** This variable holds the most recent information on the base state of the lateral APPR mode of the offside FGS.

STATE VARIABLE

**Offside\_LAPPR\_Selected****Parent:** Offside\_Modes<sub>v-135</sub>.On ▷ Offside\_LAPPR<sub>v-138</sub>.Selected**Type:** {Armed, Active}<sub>ty-23</sub>**Initial Value:** UNDEFINED**Classified as:** MONITORED**:=** Armed **if** **not** Offside\_Lappr\_Active<sub>iv-147</sub>**:=** Active **if** Offside\_Lappr\_Active<sub>iv-147</sub>

**Purpose:** This variable holds the most recent information on the selected state of the lateral APPR mode of the offside FGS.

---

STATE VARIABLE
----------------

---

### Offside\_LGA

---

**Parent:** Offside\_Modes<sub>v-135</sub>.On

**Type:** {Cleared, Selected}<sub>ty-23</sub>

**Initial Value:** UNDEFINED

**Classified as:** MONITORED

**:= Cleared if not** Offside\_Lga\_Selected<sub>iv-147</sub>

**:= Selected if** Offside\_Lga\_Selected<sub>iv-147</sub>

**Purpose:** This variable holds the most recent information on the base state of the lateral GA mode of the offside FGS.

---

STATE VARIABLE
----------------

---

### Offside\_PITCH

---

**Parent:** Offside\_Modes<sub>v-135</sub>.On

**Type:** {Cleared, Selected}<sub>ty-23</sub>

**Initial Value:** UNDEFINED

**Classified as:** MONITORED

**:= Cleared if not** Offside\_Pitch\_Selected<sub>iv-148</sub>

**:= Selected if** Offside\_Pitch\_Selected<sub>iv-148</sub>

**Purpose:** This variable holds the most recent information on the base state of the vertical PITCH mode of the offside FGS.

---

STATE VARIABLE
----------------

---

**Offside\_VS**

---

**Parent:** Offside\_Modes<sub>v-135</sub>.On

**Type:** {Cleared, Selected}<sub>ty-23</sub>

**Initial Value:** UNDEFINED

**Classified as:** MONITORED

**:= Cleared if not** Offside\_Vs\_Selected<sub>iv-148</sub>

**:= Selected if** Offside\_Vs\_Selected<sub>iv-148</sub>

**Purpose:** This variable holds the most recent information on the base state of the vertical VS mode of the offside FGS.

---

STATE VARIABLE
----------------

---

**Offside\_FLC**

---

**Parent:** Offside\_Modes<sub>v-135</sub>.On

**Type:** {Cleared, Selected}<sub>ty-23</sub>

**Initial Value:** UNDEFINED

**Classified as:** MONITORED

**:= Cleared if not** Offside\_Flc\_Selected<sub>iv-148</sub>

**:= Selected if** Offside\_Flc\_Selected<sub>iv-148</sub>

**Purpose:** This variable holds the most recent information on the base state of the vertical FLC mode of the offside FGS.

---

STATE VARIABLE
----------------

---

### Offside\_ALT

---

**Parent:** Offside\_Modes<sub>v-135</sub>.On

**Type:** {Cleared, Selected}<sub>ty-23</sub>

**Initial Value:** UNDEFINED

**Classified as:** MONITORED

**:= Cleared if not** Offside\_Alt\_Selected<sub>iv-149</sub>

**:= Selected if** Offside\_Alt\_Selected<sub>iv-149</sub>

**Purpose:** This variable holds the most recent information on the base state of the vertical ALT mode of the offside FGS.

---

STATE VARIABLE
----------------

---

### Offside\_ALTSEL

---

**Parent:** Offside\_Modes<sub>v-135</sub>.On

**Type:** {Cleared, Selected}<sub>ty-23</sub>

**Initial Value:** UNDEFINED

**Classified as:** MONITORED

**:= Cleared if not** Offside\_Altsel\_Selected<sub>iv-149</sub>

**:= Selected if** Offside\_Altsel\_Selected<sub>iv-149</sub>

**Purpose:** This variable holds the most recent information on the base state of the vertical ALTSEL mode of the offside FGS.

STATE VARIABLE

**Offside\_ALTSEL\_Selected****Parent:** Offside\_Modes<sub>v-135</sub>.On ▷ Offside\_ALTSEL<sub>v-141</sub>.Selected**Type:** {Armed, Active}<sub>ty-23</sub>**Initial Value:** UNDEFINED**Classified as:** MONITORED**:=** Armed **if** **not** Offside\_Altsel\_Active<sub>iv-149</sub>**:=** Active **if** Offside\_Altsel\_Active<sub>iv-149</sub>

**Purpose:** This variable holds the most recent information on the selected state of the vertical ALTSEL mode of the offside FGS.

STATE VARIABLE

**Offside\_ALTSEL\_Active****Parent:**Offside\_Modes<sub>v-135</sub>.On ▷ Offside\_ALTSEL<sub>v-141</sub>.Selected ▷ Offside\_ALTSEL\_Selected<sub>v-142</sub>.Active**Type:** {Capture, Track}<sub>ty-23</sub>**Initial Value:** UNDEFINED**Classified as:** MONITORED**:=** Capture **if** **not** Offside\_Altsel\_Track<sub>iv-150</sub>**:=** Track **if** Offside\_Altsel\_Track<sub>iv-150</sub>

**Purpose:** This variable holds the most recent information on the active state of the vertical ALTSEL mode of the offside FGS.

STATE VARIABLE

**Offside\_VAPPR****Parent:** Offside\_Modes<sub>v-135</sub>.On**Type:** {Cleared, Selected}<sub>ty-23</sub>**Initial Value:** UNDEFINED**Classified as:** MONITORED**:=** Cleared **if** **not** Offside\_Vappr\_Selected<sub>iv-150</sub>**:=** Selected **if** Offside\_Vappr\_Selected<sub>iv-150</sub>

**Purpose:** This variable holds the most recent information on the base state of the vertical APPR mode of the offside FGS.

STATE VARIABLE

**Offside\_VAPPR\_Selected****Parent:** Offside\_Modes<sub>v-135</sub>.On ▷ Offside\_VAPPR<sub>v-143</sub>.Selected**Type:** {Armed, Active}<sub>ty-23</sub>**Initial Value:** UNDEFINED**Classified as:** MONITORED**:=** Armed **if** **not** Offside\_Vappr\_Active<sub>iv-150</sub>**:=** Active **if** Offside\_Vappr\_Active<sub>iv-150</sub>

**Purpose:** This variable holds the most recent information on the selected state of the vertical APPR mode of the offside FGS.

---

STATE VARIABLE
----------------

---

**Offside\_VGA**

---

**Parent:** Offside\_Modes<sub>v-135</sub>.On

**Type:** {Cleared, Selected}<sub>ty-23</sub>

**Initial Value:** UNDEFINED

**Classified as:** MONITORED

**:=** Cleared **if** **not** Offside\_Vga\_Selected<sub>iv-151</sub>

**:=** Selected **if** Offside\_Vga\_Selected<sub>iv-151</sub>

**Purpose:** This variable holds the most recent information on the base state of the vertical GA mode of the offside FGS.

Definitions of Values to be Encapsulated
--

---

INPUT VARIABLE
----------------

---

**Offside\_FGS\_Active**

---

**Type:** BOOLEAN

**Initial Value:** UNDEFINED

**Classified as:** Input

**Purpose:** This input variable holds the most recent information on whether the offside FGS believes it is active.

---

INPUT VARIABLE

---

### Offside\_FD\_On

---

**Type:** BOOLEAN

**Initial Value:** UNDEFINED

**Classified as:** Input

**Purpose:** This input variable holds the most recent information on whether the offside FD is turned on.

---

INPUT VARIABLE

---

### Offside\_Modes\_On

---

**Type:** BOOLEAN

**Initial Value:** UNDEFINED

**Classified as:** Input

**Purpose:** This input variable holds the most recent information on whether the offside modes are on.

---

INPUT VARIABLE

---

### Offside\_Roll\_Selected

---

**Type:** BOOLEAN

**Initial Value:** UNDEFINED

**Classified as:** Input

**Purpose:** This input variable holds the most recent information on whether the offside ROLL mode is selected.

---

INPUT VARIABLE
----------------

---

### Offside\_Hdg\_Selected

---

**Type:** BOOLEAN

**Initial Value:** UNDEFINED

**Classified as:** Input

**Purpose:** This input variable holds the most recent information on whether the offside Heading mode is selected.

---

INPUT VARIABLE
----------------

---

### Offside\_Nav\_Selected

---

**Type:** BOOLEAN

**Initial Value:** UNDEFINED

**Classified as:** Input

**Purpose:** This input variable holds the most recent information on whether the offside Nav mode is selected.

---

INPUT VARIABLE
----------------

---

### Offside\_Nav\_Active

---

**Type:** BOOLEAN

**Initial Value:** UNDEFINED

**Classified as:** Input

**Purpose:** This input variable holds the most recent information on whether the offside Nav mode is active.

---

INPUT VARIABLE
----------------

---

### Offside\_Lappr\_Selected

---

**Type:** BOOLEAN

**Initial Value:** UNDEFINED

**Classified as:** Input

**Purpose:** This input variable holds the most recent information on whether the offside Lappr mode is selected.

---

INPUT VARIABLE
----------------

---

### Offside\_Lappr\_Active

---

**Type:** BOOLEAN

**Initial Value:** UNDEFINED

**Classified as:** Input

**Purpose:** This input variable holds the most recent information on whether the offside Lappr mode is active.

---

INPUT VARIABLE
----------------

---

### Offside\_Lga\_Selected

---

**Type:** BOOLEAN

**Initial Value:** UNDEFINED

**Classified as:** Input

**Purpose:** This input variable holds the most recent information on whether the offside Lga mode is selected.

---

INPUT VARIABLE
----------------

---

### Offside\_Pitch\_Selected

---

**Type:** BOOLEAN

**Initial Value:** UNDEFINED

**Classified as:** Input

**Purpose:** This input variable holds the most recent information on whether the offside vertical Pitch mode is selected.

---

INPUT VARIABLE
----------------

---

### Offside\_Vs\_Selected

---

**Type:** BOOLEAN

**Initial Value:** UNDEFINED

**Classified as:** Input

**Purpose:** This input variable holds the most recent information on whether the offside vertical Speed mode is selected.

---

INPUT VARIABLE
----------------

---

### Offside\_Flc\_Selected

---

**Type:** BOOLEAN

**Initial Value:** UNDEFINED

**Classified as:** Input

**Purpose:** This input variable holds the most recent information on whether the offside vertical Flight Level Change mode is selected.

---

INPUT VARIABLE
----------------

---

### Offside\_Alt\_Selected

---

**Type:** BOOLEAN

**Initial Value:** UNDEFINED

**Classified as:** Input

**Purpose:** This input variable holds the most recent information on whether the offside vertical Altitude Hold mode is selected.

---

INPUT VARIABLE
----------------

---

### Offside\_Altsel\_Selected

---

**Type:** BOOLEAN

**Initial Value:** UNDEFINED

**Classified as:** Input

**Purpose:** This input variable holds the most recent information on whether the offside vertical Altitude Select mode is selected.

---

INPUT VARIABLE
----------------

---

### Offside\_Altsel\_Active

---

**Type:** BOOLEAN

**Initial Value:** UNDEFINED

**Classified as:** Input

**Purpose:** This input variable holds the most recent information on whether the offside vertical Altitude Select mode is active.

---

INPUT VARIABLE
----------------

---

### Offside\_AltSel\_Track

---

**Type:** BOOLEAN

**Initial Value:** UNDEFINED

**Classified as:** Input

**Purpose:** This input variable holds the most recent information on whether the offside vertical Altitude Select mode is tracking.

---

INPUT VARIABLE
----------------

---

### Offside\_Vappr\_Selected

---

**Type:** BOOLEAN

**Initial Value:** UNDEFINED

**Classified as:** Input

**Purpose:** This input variable holds the most recent information on whether the offside Vappr mode is selected.

---

INPUT VARIABLE
----------------

---

### Offside\_Vappr\_Active

---

**Type:** BOOLEAN

**Initial Value:** UNDEFINED

**Classified as:** Input

**Purpose:** This input variable holds the most recent information on whether the offside Vappr mode is active.

---

INPUT VARIABLE
----------------

---

### Offside\_Vga\_Selected

---

**Type:** BOOLEAN

**Initial Value:** UNDEFINED

**Classified as:** Input

**Purpose:** This input variable holds the most recent information on whether the offside Vga mode is selected.

## 2.15 FGS Inputs

This section defines the physical interface for all inputs to the FGS. The input variables associated with these fields are defined in the part of the specification to which they are logically related.

---

MESSAGE
---------

---

### Other\_Input\_Msg

---

<b>Fields:</b>	AltSel	is	BOOLEAN
	, AltselAct	is	BOOLEAN
	, AltselSel	is	BOOLEAN
	, AltselTrk	is	BOOLEAN
	, FdOn	is	BOOLEAN
	, FGSActive	is	BOOLEAN
	, FlcSel	is	BOOLEAN
	, HdgSel	is	BOOLEAN
	, LapprAct	is	BOOLEAN
	, LapprSel	is	BOOLEAN
	, LgaSel	is	BOOLEAN
	, ModesOn	is	BOOLEAN
	, NavAct	is	BOOLEAN
	, NavSel	is	BOOLEAN
	, PthSel	is	BOOLEAN
	, RollSel	is	BOOLEAN
	, VapprAct	is	BOOLEAN
	, VapprSel	is	BOOLEAN
	, VgaSel	is	BOOLEAN
	, VsSel	is	BOOLEAN

---

MESSAGE
---------

---

### This\_Input\_Msg

---

<b>Fields:</b>	AltPreRefChanged	is	BOOLEAN
	, AltselCaptureCondMet	is	BOOLEAN
	, AltselTargetAltChanged	is	BOOLEAN
	, AltselTrackCondMet	is	BOOLEAN
	, AltSwi	is	{OFF, ON} <sub>ty-117</sub>
	, ApDiscSwi	is	{OFF, ON} <sub>ty-117</sub>
	, ApEngSwi	is	{OFF, ON} <sub>ty-117</sub>
	, ApprSwi	is	{OFF, ON} <sub>ty-117</sub>
	, FdSwi	is	{OFF, ON} <sub>ty-117</sub>
	, FlcSwi	is	{OFF, ON} <sub>ty-117</sub>
	, GaSwi	is	{OFF, ON} <sub>ty-117</sub>
	, HdgSwi	is	{OFF, ON} <sub>ty-117</sub>
	, LapprTrackCondMet	is	BOOLEAN
	, NavSwi	is	{OFF, ON} <sub>ty-117</sub>
	, NavTrackCondMet	is	BOOLEAN
	, Overspeed	is	BOOLEAN
	, SyncSwi	is	{OFF, ON} <sub>ty-117</sub>
	, TransSwi	is	{OFF, ON} <sub>ty-117</sub>
	, VapprTrackCondMet	is	BOOLEAN
	, VsPthWhlMot	is	BOOLEAN
	, VsSwi	is	{OFF, ON} <sub>ty-117</sub>

---

INPUT INTERFACE
-----------------

---

### Other\_Input

---

**Min. Separation:** UNDEFINED

**Max. Separation:** UNDEFINED

**Input Action:** READ(Other\_Input\_Msg<sub>m-152</sub> )

**Handler**

**Guard Condition:** TRUE

<b>Assignment(s)</b>	Offside_Alt_Selected <sub>iv-149</sub>	<b>is assigned</b>	AltSel
	Offside_AltSel_Active <sub>iv-149</sub>	<b>is assigned</b>	AltSelAct
	Offside_AltSel_Selected <sub>iv-149</sub>	<b>is assigned</b>	AltSelSel
	Offside_AltSel_Track <sub>iv-150</sub>	<b>is assigned</b>	AltSelTrk
	Offside_FD_On <sub>iv-145</sub>	<b>is assigned</b>	FdOn
	Offside_FGS_Active <sub>iv-144</sub>	<b>is assigned</b>	FGSActive
	Offside_Flc_Selected <sub>iv-148</sub>	<b>is assigned</b>	FlcSel
	Offside_Hdg_Selected <sub>iv-146</sub>	<b>is assigned</b>	HdgSel
	Offside_Lappr_Selected <sub>iv-147</sub>	<b>is assigned</b>	LapprSel
	Offside_Lappr_Active <sub>iv-147</sub>	<b>is assigned</b>	LapprAct
	Offside_Lga_Selected <sub>iv-147</sub>	<b>is assigned</b>	LgaSel
	Offside_Modes_On <sub>iv-145</sub>	<b>is assigned</b>	ModesOn
	Offside_Nav_Active <sub>iv-146</sub>	<b>is assigned</b>	NavAct
	Offside_Nav_Selected <sub>iv-146</sub>	<b>is assigned</b>	NavSel
	Offside_Pitch_Selected <sub>iv-148</sub>	<b>is assigned</b>	PthSel
	Offside_Roll_Selected <sub>iv-145</sub>	<b>is assigned</b>	RollSel
	Offside_Vappr_Selected <sub>iv-150</sub>	<b>is assigned</b>	VapprSel
	Offside_Vappr_Active <sub>iv-150</sub>	<b>is assigned</b>	VapprAct
	Offside_Vga_Selected <sub>iv-151</sub>	<b>is assigned</b>	VgaSel
	Offside_Vs_Selected <sub>iv-148</sub>	<b>is assigned</b>	VsSel

---

INPUT INTERFACE
-----------------

---

### This\_Input

---

**Min. Separation:** UNDEFINED

**Max. Separation:** UNDEFINED

**Input Action:** READ(This\_Input\_Msg<sub>m-152</sub> )

#### Handler

**Guard Condition:** TRUE

#### Assignment(s)

Is_ALTSEL_Capture_Cond_Met <sub>iv-101</sub>	<b>is assigned</b>	AltselCaptureCondMet
ALTSEL_Target_Altitude_Changed <sub>iv-129</sub>	<b>is assigned</b>	AltselTargetAltChanged
Is_ALTSEL_Track_Cond_Met <sub>iv-101</sub>	<b>is assigned</b>	AltselTrackCondMet
ALT_Switch <sub>iv-121</sub>	<b>is assigned</b>	AltSwi
AP_Disconnect_Switch <sub>iv-126</sub>	<b>is assigned</b>	ApDiscSwi
AP_Engage_Switch <sub>iv-126</sub>	<b>is assigned</b>	ApEngSwi
APPR_Switch <sub>iv-123</sub>	<b>is assigned</b>	ApprSwi
FD_Switch <sub>iv-118</sub>	<b>is assigned</b>	FdSwi
FLC_Switch <sub>iv-122</sub>	<b>is assigned</b>	FlcSwi
GA_Switch <sub>iv-124</sub>	<b>is assigned</b>	GaSwi
HDG_Switch <sub>iv-118</sub>	<b>is assigned</b>	HdgSwi
Is_LAPPR_Track_Cond_Met <sub>iv-101</sub>	<b>is assigned</b>	LapprTrackCondMet
NAV_Switch <sub>iv-119</sub>	<b>is assigned</b>	NavSwi
Is_NAV_Track_Cond_Met <sub>iv-100</sub>	<b>is assigned</b>	NavTrackCondMet
Overspeed <sub>iv-128</sub>	<b>is assigned</b>	Overspeed <sub>iv-128</sub>
SYNC_Switch <sub>iv-125</sub>	<b>is assigned</b>	SyncSwi
Transfer_Switch <sub>iv-125</sub>	<b>is assigned</b>	TransSwi
Is_VAPPR_Track_Cond_Met <sub>iv-102</sub>	<b>is assigned</b>	VapprTrackCondMet
VS_Pitch_Wheel_In_Motion <sub>iv-125</sub>	<b>is assigned</b>	VsPthWhlMot
VS_Switch <sub>iv-120</sub>	<b>is assigned</b>	VsSwi

## 2.16 FGS Outputs

This section defines the physical interface for all outputs from the FGS. The output variables associated with these fields are defined in the part of the specification to which they are logically related.

---

MESSAGE
---------

---

### This\_Output\_Msg

---

<b>Fields:</b>	AltLamp <b>is</b> {OFF, ON} <sub>ty-118</sub> , AltSel <b>is</b> BOOLEAN , AltselAct <b>is</b> BOOLEAN , AltselSel <b>is</b> BOOLEAN , AltselTrk <b>is</b> BOOLEAN , ApEng <b>is</b> BOOLEAN , ApLamp <b>is</b> {OFF, ON} <sub>ty-118</sub> , ApprLamp <b>is</b> {OFF, ON} <sub>ty-118</sub> , FdOn <b>is</b> BOOLEAN , FGSActive <b>is</b> BOOLEAN , FlcLamp <b>is</b> {OFF, ON} <sub>ty-118</sub> , FlcSel <b>is</b> BOOLEAN , HdgLamp <b>is</b> {OFF, ON} <sub>ty-118</sub> , HdgSel <b>is</b> BOOLEAN , LapprAct <b>is</b> BOOLEAN , LapprSel <b>is</b> BOOLEAN , LgaSel <b>is</b> BOOLEAN , ModesOn <b>is</b> BOOLEAN , NavAct <b>is</b> BOOLEAN , NavLamp <b>is</b> {OFF, ON} <sub>ty-118</sub> , NavSel <b>is</b> BOOLEAN , PilotFlying <b>is</b> {LEFT, RIGHT} <sub>ty-23</sub> , PthSel <b>is</b> BOOLEAN , RollSel <b>is</b> BOOLEAN , VapprAct <b>is</b> BOOLEAN , VapprSel <b>is</b> BOOLEAN , VgaSel <b>is</b> BOOLEAN , VsLamp <b>is</b> {OFF, ON} <sub>ty-118</sub> , VsSel <b>is</b> BOOLEAN
----------------	--







<b>Assignment(s)</b>	AltLamp	<b>is assigned</b>	ALT_Lamp <sub>v-122</sub>
	AltSel	<b>is assigned</b>	Is_ALT_Selected <sub>v-74</sub>
	AltselAct	<b>is assigned</b>	Is_ALTSEL_Active <sub>v-80</sub>
	AltselSel	<b>is assigned</b>	Is_ALTSEL_Selected <sub>v-79</sub>
	AltselTrk	<b>is assigned</b>	Is_ALTSEL_Track <sub>v-80</sub>
	ApEng	<b>is assigned</b>	Is_AP_Engaged <sub>v-131</sub>
	ApLamp	<b>is assigned</b>	AP_Lamp <sub>v-126</sub>
	ApprLamp	<b>is assigned</b>	APPR_Lamp <sub>v-124</sub>
	FdOn	<b>is assigned</b>	Onside_FD_On <sub>v-28</sub>
	FGSActive	<b>is assigned</b>	Is_This_Side_Active <sub>v-30</sub>
	FlcLamp	<b>is assigned</b>	FLC_Lamp <sub>v-123</sub>
	FlcSel	<b>is assigned</b>	Is_FLC_Selected <sub>v-70</sub>
	HdgLamp	<b>is assigned</b>	HDG_Lamp <sub>v-119</sub>
	HdgSel	<b>is assigned</b>	Is_HDG_Selected <sub>v-41</sub>
	LapprSel	<b>is assigned</b>	Is_LAPPR_Selected <sub>v-52</sub>
	LapprAct	<b>is assigned</b>	Is_LAPPR_Active <sub>v-52</sub>
	LgaSel	<b>is assigned</b>	Is_LGA_Selected <sub>v-57</sub>
	ModesOn	<b>is assigned</b>	Mode_Annunciations_On <sub>v-33</sub>
	NavAct	<b>is assigned</b>	Is_NAV_Active <sub>v-46</sub>
	NavLamp	<b>is assigned</b>	NAV_Lamp <sub>v-120</sub>
	NavSel	<b>is assigned</b>	Is_NAV_Selected <sub>v-46</sub>
	PilotFlying	<b>is assigned</b>	Pilot_Flying <sub>v-29</sub>
	PthSel	<b>is assigned</b>	Is_PITCH_Selected <sub>v-63</sub>
	RollSel	<b>is assigned</b>	Is_ROLL_Selected <sub>v-38</sub>
	VapprSel	<b>is assigned</b>	Is_VAPPR_Selected <sub>v-87</sub>
	VapprAct	<b>is assigned</b>	Is_VAPPR_Active <sub>v-87</sub>
	VgaSel	<b>is assigned</b>	Is_VGA_Selected <sub>v-92</sub>
	VsLamp	<b>is assigned</b>	VS_Lamp <sub>v-121</sub>
	VsSel	<b>is assigned</b>	Is_VS_Selected <sub>v-66</sub>

**Action:** SEND

# Bibliography

- [1] Anonymous. Mission critical systems: Defense attempting to address major software challenges. Technical Report GAO/IMTEC-93-13, U.S. General Accounting Office, 1992.
- [2] Anonymous. Joint advisory circular: Flight guidance system approval. Technical Report AC/ACJ 25.1329, Federal Aviation Administration, 2001.
- [3] Barry Boehm. Prentice-Hall, Englewood Cliffs, NJ, 1981.
- [4] Fred Brooks. No silver bullet: : Essence and accidents of software engineering. *IEEE Computer*, pages 10–19, April 1997.
- [5] David Harel and Amnon Naamad. The STATEMATE semantics of statecharts. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 5(4):293–333, October 1996.
- [6] Matts Heimdahl and Nancy G. Leveson. Completeness and consistency in hierarchical state-based requirements. *IEEE Transactions on Software Engineering*, 22(6):363–377, June 1996.
- [7] Constance L. Heitmeyer, James Kirby, , and Bruce G. Labaw. Automated consistency checking of requirements specification. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 5(3):231–261, July 1996.
- [8] Nancy Leveson, Matts Heimdahl, Holly Hildreth, and Jon Reese. Requirements specifications for process-control systems. *IEEE Transactions on Software Engineering*, 20(9):684–707, September 1994.
- [9] Nancy Leveson, Denise Pinnel, Sean Sandys, Shuichi Koga, and Jon Reese. Analyzing software specifications for mode confusion potential. 1997.
- [10] Nancy Leveson, Sean Sandys, Denise Pinnel, Susan Joslyn, Liliana Alfaro, Zelda Zabinsky, and Alan Shaw. Safety analysis of air traffic control upgrades. Technical report, Safeware Engineering, 1997.
- [11] Nancy G. Leveson. Addison-Wesley Publishing Company, Reading, Massachusetts, 1995.

- [12] Robyn Lutz. Analyzing software requirements errors in safety-critical, embedded systems. In *IEEE Symposium on Requirements Engineering*, San Diego, CA, 1993.
- [13] Steven P. Miller. Specifying the mode logic of a flight guidance system in core and scr. In *Second Workshop on Formal Methods in Software Practice (FMSP98)*, Clearwater Beach, Florida, 1998.
- [14] E. H. Phillip. Global teamwork called key to improving aviation safety. *Aviation Week & Space Technology*, page 70, July 2001.

# Index

- ALTSEL (State Variable)
  - Definition, 82
  - Reference(s), 79, 83, 84
- ALT (State Variable)
  - Definition, 76
  - Reference(s), 74, 75
- ALT\_Lamp (State Variable)
  - Definition, 122
  - Reference(s), 158, 160
- ALT\_Switch (Input Variable)
  - Definition, 121
  - Reference(s), 112, 155
- ALTSEL\_Active (State Variable)
  - Definition, 84
  - Reference(s), 80
- ALTSEL\_Selected (State Variable)
  - Definition, 83
  - Reference(s), 80, 81, 84
- ALTSEL\_Target\_Altitude\_Changed (Input Variable)
  - Definition, 129
  - Reference(s), 113, 155
- AP\_Disconnect\_Switch (Input Variable)
  - Definition, 126
  - Reference(s), 106, 155
- AP\_Engage\_Switch (Input Variable)
  - Definition, 126
  - Reference(s), 105, 155
- AP\_Lamp (State Variable)
  - Definition, 126
  - Reference(s), 158, 160
- AP\_State (Type)
  - Definition, 131
  - Reference(s), 132
- APPR\_Lamp (State Variable)
  - Definition, 124
  - Reference(s), 158, 160
- APPR\_Switch (Input Variable)
  - Definition, 123
  - Reference(s), 114, 155
- Activate\_LAPPR (Macro)
  - Definition, 50
  - Reference(s), 53, 55
- Activate\_VAPPR (Macro)
  - Definition, 85
  - Reference(s), 88, 90
- Activate\_NAV (Macro)
  - Definition, 44
  - Reference(s), 47, 49
- Active\_State (Type)
  - Definition, 23
  - Reference(s), 84, 142
- AP (State Variable)
  - Definition, 132
  - Reference(s), 25, 32, 56, 91, 131
- Base\_State (Type)
  - Definition, 23
  - Reference(s), 39, 43, 48, 54, 59, 64, 68, 72, 76, 82, 89, 94, 136–141, 143, 144
- Boolean
  - Reference, 28, 30, 33, 38, 41, 46, 52, 57, 63, 66, 70, 71, 74, 79, 80, 87, 92, 100–102, 125, 127–129, 131, 134, 135, 144–153, 156
- Capture\_ALTSEL (Macro)
  - Definition, 78
  - Reference(s), 81, 83
- Deactivate\_ALTSEL (Macro)
  - Definition, 79
  - Reference(s), 83
- Deactivate\_LAPPR (Macro)
  - Definition, 51
  - Reference(s), 54
- Deactivate\_VAPPR (Macro)
  - Definition, 86
  - Reference(s), 89

- Deactivate\_NAV (Macro)
  - Definition, 45
  - Reference(s), 48
- Dearm\_LAPPR (Macro)
  - Definition, 51
  - Reference(s), 54
- Dearm\_NAV (Macro)
  - Definition, 45
  - Reference(s), 48
- Deselect\_ALTSEL (Macro)
  - Definition, 78
  - Reference(s), 82
- Deselect\_LAPPR (Macro)
  - Definition, 51
  - Reference(s), 54
- Deselect\_PITCH (Macro)
  - Definition, 62
  - Reference(s), 64
- Deselect\_ROLL (Macro)
  - Definition, 37
  - Reference(s), 39
- Deselect\_VAPPR (Macro)
  - Definition, 86
  - Reference(s), 89
- Deselect\_ALT (Macro)
  - Definition, 73
  - Reference(s), 76
- Deselect\_FLC (Macro)
  - Definition, 70
  - Reference(s), 72
- Deselect\_HDG (Macro)
  - Definition, 40
  - Reference(s), 43
- Deselect\_LGA (Macro)
  - Definition, 56
  - Reference(s), 59
- Deselect\_NAV (Macro)
  - Definition, 45
  - Reference(s), 48
- Deselect\_VGA (Macro)
  - Definition, 91
  - Reference(s), 94
- Deselect\_VS (Macro)
  - Definition, 65
  - Reference(s), 68
- FD\_Switch (Input Variable)
  - Definition, 118
- Reference(s), 103, 155
- FLC (State Variable)
  - Definition, 72
  - Reference(s), 70, 71
- FLC\_Lamp (State Variable)
  - Definition, 123
  - Reference(s), 158, 160
- FLC\_Switch (Input Variable)
  - Definition, 122
  - Reference(s), 111, 155
- GA\_Switch (Input Variable)
  - Definition, 124
  - Reference(s), 115, 155
- HDG (State Variable)
  - Definition, 43
  - Reference(s), 41, 42
- HDG\_Lamp (State Variable)
  - Definition, 119
  - Reference(s), 158, 160
- HDG\_Switch (Input Variable)
  - Definition, 118
  - Reference(s), 108, 155
- Independent\_Mode (State Variable)
  - Definition, 31
  - Reference(s), 30
- Independent\_Mode\_Condition (Macro)
  - Definition, 31
  - Reference(s), 31
- Is\_ALT\_Active (State Variable)
  - Definition, 74
  - Reference(s), 61, 69, 77, 78
- Is\_ALT\_Selected (State Variable)
  - Definition, 74
  - Reference(s), 122, 157, 160
- Is\_ALTSEL\_Active (State Variable)
  - Definition, 80
  - Reference(s), 27, 61, 69, 157, 160
- Is\_ALTSEL\_Capture\_Cond\_Met (Input Variable)
  - Definition, 101
  - Reference(s), 97, 155
- Is\_ALTSEL\_Selected (State Variable)
  - Definition, 79
  - Reference(s), 157, 160
- Is\_ALTSEL\_Track (State Variable)
  - Definition, 80
  - Reference(s), 73, 157, 160

- Is\_ALTSEL\_Track\_Cond\_Met (Input Variable)  
 Definition, 101  
 Reference(s), 96, 155
- Is\_AP\_Engaged (State Variable)  
 Definition, 131  
 Reference(s), 32, 126, 159, 160
- Is\_FLC\_Active (State Variable)  
 Definition, 71  
 Reference(s), 61
- Is\_FLC\_Selected (State Variable)  
 Definition, 70  
 Reference(s), 123, 157, 160
- Is\_HDG\_Active (State Variable)  
 Definition, 41  
 Reference(s), 36
- Is\_HDG\_Selected (State Variable)  
 Definition, 41  
 Reference(s), 119, 157, 160
- Is\_LAPPR\_Active (State Variable)  
 Definition, 52  
 Reference(s), 31, 36, 85, 157, 160
- Is\_LAPPR\_Selected (State Variable)  
 Definition, 52  
 Reference(s), 86, 124, 157, 160
- Is\_LAPPR\_Track\_Cond\_Met (Input Variable)  
 Definition, 101  
 Reference(s), 98, 155
- Is\_LGA\_Active (State Variable)  
 Definition, 57  
 Reference(s), 36
- Is\_LGA\_Selected (State Variable)  
 Definition, 57  
 Reference(s), 157, 160
- Is\_NAV\_Active (State Variable)  
 Definition, 46  
 Reference(s), 36, 157, 160
- Is\_NAV\_Selected (State Variable)  
 Definition, 46  
 Reference(s), 120, 157, 160
- Is\_NAV\_Track\_Cond\_Met (Input Variable)  
 Definition, 100  
 Reference(s), 95, 155
- Is\_No\_Nonbasic\_Lateral\_Mode\_Active (Macro)  
 Definition, 36  
 Reference(s), 37
- Is\_No\_Nonbasic\_Vertical\_Mode\_Active (Macro)  
 Definition, 61  
 Reference(s), 62
- Is\_Offside\_LAPPR\_Active (State Variable)  
 Definition, 134  
 Reference(s), 31
- Is\_Offside\_VAPPR\_Active (State Variable)  
 Definition, 134  
 Reference(s), 31
- Is\_Offside\_VGA\_Active (State Variable)  
 Definition, 135  
 Reference(s), 31
- Is\_PITCH\_Active (State Variable)  
 Definition, 63
- Is\_PITCH\_Selected (State Variable)  
 Definition, 63  
 Reference(s), 157, 160
- Is\_ROLL\_Active (State Variable)  
 Definition, 38
- Is\_ROLL\_Selected (State Variable)  
 Definition, 38  
 Reference(s), 157, 160
- Is\_This\_Side\_Active (State Variable)  
 Definition, 30  
 Reference(s), 26, 27, 34, 39, 42, 43, 47–49,  
 53–55, 58, 59, 64, 67, 68, 71, 72, 75, 76,  
 81–84, 88–90, 93, 94, 159, 160
- Is\_VAPPR\_Active (State Variable)  
 Definition, 87  
 Reference(s), 27, 31, 61, 65, 69, 73, 77, 78,  
 157, 160
- Is\_VAPPR\_Selected (State Variable)  
 Definition, 87  
 Reference(s), 124, 157, 160
- Is\_VAPPR\_Track\_Cond\_Met (Input Variable)  
 Definition, 102  
 Reference(s), 99, 155
- Is\_VGA\_Active (State Variable)  
 Definition, 92  
 Reference(s), 31, 61, 77, 78
- Is\_VGA\_Selected (State Variable)  
 Definition, 92  
 Reference(s), 157, 160
- Is\_VS\_Active (State Variable)  
 Definition, 66  
 Reference(s), 27, 61
- Is\_VS\_Selected (State Variable)  
 Definition, 66  
 Reference(s), 121, 157, 160
- LAPPR (State Variable)

- Definition, 54
- Reference(s), 52, 53, 55
- LAPPR\_Selected (State Variable)
  - Definition, 55
  - Reference(s), 52–54
- LGA (State Variable)
  - Definition, 59
  - Reference(s), 57, 58
- Lamp (Type)
  - Definition, 118
  - Reference(s), 119–124, 126, 156
- Mode\_Annunciations\_On (State Variable)
  - Definition, 33
  - Reference(s), 25–27, 159, 160
- Modes (State Variable)
  - Definition, 34
  - Reference(s), 33, 37, 39, 40, 43, 45, 48, 49, 51, 54–56, 59, 62, 64, 65, 68, 70, 72, 73, 76–78, 82–84, 86, 89–91, 94
- NAV (State Variable)
  - Definition, 48
  - Reference(s), 46, 47, 49
- NAV\_Lamp (State Variable)
  - Definition, 120
  - Reference(s), 158, 160
- NAV\_Selected (State Variable)
  - Definition, 49
  - Reference(s), 46–48
- NAV\_Switch (Input Variable)
  - Definition, 119
  - Reference(s), 107, 155
- No\_Higher\_Event\_Than\_ALT\_Switch\_Pressed (Macro)
  - Definition, 113
  - Reference(s), 112, 113
- No\_Higher\_Event\_Than\_ALTSEL\_Capture\_Cond\_Met (Macro)
  - Definition, 98
  - Reference(s), 97
- No\_Higher\_Event\_Than\_ALTSEL\_Target\_Altitude\_Changed (Macro)
  - Definition, 114
  - Reference(s), 113, 114
- No\_Higher\_Event\_Than\_ALTSEL\_Track\_Cond\_Met (Macro)
  - Definition, 97
  - Reference(s), 96
- No\_Higher\_Event\_Than\_AP\_Disconnect\_Switch\_Pressed (Macro)
  - Definition, 107
  - Reference(s), 106, 107
- No\_Higher\_Event\_Than\_AP\_Engage\_Switch\_Pressed (Macro)
  - Definition, 106
  - Reference(s), 105, 106
- No\_Higher\_Event\_Than\_APPR\_Switch\_Pressed (Macro)
  - Definition, 115
  - Reference(s), 109, 114, 115
- No\_Higher\_Event\_Than\_FD\_Switch\_Pressed (Macro)
  - Definition, 104
  - Reference(s), 96–100, 103
- No\_Higher\_Event\_Than\_FLC\_Switch\_Pressed (Macro)
  - Definition, 112
  - Reference(s), 111, 112
- No\_Higher\_Event\_Than\_GA\_Switch\_Pressed (Macro)
  - Definition, 116
  - Reference(s), 115, 116
- No\_Higher\_Event\_Than\_HDG\_Switch\_Pressed (Macro)
  - Definition, 109
  - Reference(s), 108, 109
- No\_Higher\_Event\_Than\_LAPPR\_Track\_Cond\_Met (Macro)
  - Definition, 99
  - Reference(s), 98
- No\_Higher\_Event\_Than\_NAV\_Switch\_Pressed (Macro)
  - Definition, 108
  - Reference(s), 105, 108
- No\_Higher\_Event\_Than\_NAV\_Track\_Cond\_Met (Macro)
  - Definition, 96
  - Reference(s), 95

No\_Higher\_Event\_Than\_SYNC\_Switch\_Pressed  
     (Macro)  
     Definition, 117  
     Reference(s), 116, 117  
 No\_Higher\_Event\_Than\_Transfer\_Switch\_Pressed  
     (Macro)  
     Definition, 105  
     Reference(s), 104  
 No\_Higher\_Event\_Than\_VAPPR\_Track\_Cond-  
     Met  
     (Macro)  
     Definition, 100  
     Reference(s), 99  
 No\_Higher\_Event\_Than\_VS\_Pitch\_Wheel-  
     Rotated  
     (Macro)  
     Definition, 110  
     Reference(s), 105, 110  
 No\_Higher\_Event\_Than\_VS\_Switch\_Pressed  
     (Macro)  
     Definition, 111  
     Reference(s), 110, 111  
  
 Offside\_ALTSEL (State Variable)  
     Definition, 141  
     Reference(s), 82, 142  
 Offside\_LAPPR (State Variable)  
     Definition, 138  
     Reference(s), 54, 138  
 Offside\_PITCH (State Variable)  
     Definition, 139  
     Reference(s), 64  
 Offside\_ROLL (State Variable)  
     Definition, 136  
     Reference(s), 39  
 Offside\_VAPPR (State Variable)  
     Definition, 143  
     Reference(s), 89, 143  
 Offside\_ALT (State Variable)  
     Definition, 141  
     Reference(s), 76  
 Offside\_Alt\_Selected (Input Variable)  
     Definition, 149  
     Reference(s), 141, 154  
 Offside\_Altsel\_Active (Input Variable)  
     Definition, 149  
     Reference(s), 142, 154  
 Offside\_ALTSEL\_Active (State Variable)  
     Definition, 142  
     Reference(s), 84  
 Offside\_Altsel\_Selected (Input Variable)  
     Definition, 149  
     Reference(s), 141, 154  
 Offside\_ALTSEL\_Selected (State Variable)  
     Definition, 142  
     Reference(s), 83, 142  
 Offside\_Altsel\_Track (Input Variable)  
     Definition, 150  
     Reference(s), 142, 154  
 Offside\_FLC (State Variable)  
     Definition, 140  
     Reference(s), 72  
 Offside\_FD (State Variable)  
     Definition, 133  
     Reference(s), 32  
 Offside\_FD\_On (Input Variable)  
     Definition, 145  
     Reference(s), 133, 154  
 Offside\_FGS\_Active (Input Variable)  
     Definition, 144  
     Reference(s), 154  
 Offside\_Flc\_Selected (Input Variable)  
     Definition, 148  
     Reference(s), 140, 154  
 Offside\_HDG (State Variable)  
     Definition, 136  
     Reference(s), 43  
 Offside\_Hdg\_Selected (Input Variable)  
     Definition, 146  
     Reference(s), 136, 154  
 Offside\_LGA (State Variable)  
     Definition, 139  
     Reference(s), 59  
 Offside\_Lappr\_Active (Input Variable)  
     Definition, 147  
     Reference(s), 138, 154  
 Offside\_Lappr\_Selected (Input Variable)  
     Definition, 147  
     Reference(s), 138, 154  
 Offside\_LAPPR\_Selected (State Variable)  
     Definition, 138  
     Reference(s), 55, 134  
 Offside\_Lga\_Selected (Input Variable)  
     Definition, 147  
     Reference(s), 139, 154  
 Offside\_Modes (State Variable)

- Definition, 135
- Reference(s), 30, 34, 136–144
- Offside\_Modes\_On (Input Variable)
  - Definition, 145
  - Reference(s), 135, 154
- Offside\_NAV (State Variable)
  - Definition, 137
  - Reference(s), 48, 137
- Offside\_Nav\_Active (Input Variable)
  - Definition, 146
  - Reference(s), 137, 154
- Offside\_Nav\_Selected (Input Variable)
  - Definition, 146
  - Reference(s), 137, 154
- Offside\_NAV\_Selected (State Variable)
  - Definition, 137
  - Reference(s), 49
- Offside\_Pitch\_Selected (Input Variable)
  - Definition, 148
  - Reference(s), 139, 154
- Offside\_Roll\_Selected (Input Variable)
  - Definition, 145
  - Reference(s), 136, 154
- Offside\_VGA (State Variable)
  - Definition, 144
  - Reference(s), 94, 135
- Offside\_Vappr\_Active (Input Variable)
  - Definition, 150
  - Reference(s), 143, 154
- Offside\_Vappr\_Selected (Input Variable)
  - Definition, 150
  - Reference(s), 143, 154
- Offside\_VAPPR\_Selected (State Variable)
  - Definition, 143
  - Reference(s), 90, 134
- Offside\_Vga\_Selected (Input Variable)
  - Definition, 151
  - Reference(s), 144, 154
- Offside\_VS (State Variable)
  - Definition, 140
  - Reference(s), 68
- Offside\_Vs\_Selected (Input Variable)
  - Definition, 148
  - Reference(s), 140, 154
- On\_Off (Type)
  - Definition, 23
  - Reference(s), 28, 31, 34, 133, 135
- Onside\_FD (State Variable)
  - Definition, 28
  - Reference(s), 28, 32
- Onside\_FD\_On (State Variable)
  - Definition, 28
  - Reference(s), 159, 160
- Other\_Input (Input Interface)
  - Definition, 153
- Other\_Input\_Msg (Message)
  - Definition, 152
  - Reference(s), 153
- Overspeed (Input Variable)
  - Definition, 128
  - Reference(s), 27, 128, 155
- Overspeed\_Condition (Macro)
  - Definition, 128
  - Reference(s), 25, 26, 56, 65, 69, 70, 85, 91
- PITCH (State Variable)
  - Definition, 64
  - Reference(s), 63
- Pilot\_Flying (State Variable)
  - Definition, 29
  - Reference(s), 25, 29, 30, 159, 160
- ROLL (State Variable)
  - Definition, 39
  - Reference(s), 38
- SYNC\_Switch (Input Variable)
  - Definition, 125
  - Reference(s), 116, 155
- Select\_ALTSEL (Macro)
  - Definition, 77
  - Reference(s), 82
- Select\_LAPPR (Macro)
  - Definition, 50
  - Reference(s), 53, 54
- Select\_PITCH (Macro)
  - Definition, 62
  - Reference(s), 64
- Select\_ROLL (Macro)
  - Definition, 37
  - Reference(s), 39
- Select\_VAPPR (Macro)
  - Definition, 85
  - Reference(s), 89
- Select\_ALT (Macro)
  - Definition, 73
  - Reference(s), 75, 76

- Select\_FLC (Macro)
  - Definition, 69
  - Reference(s), 71, 72
- Select\_HDG (Macro)
  - Definition, 40
  - Reference(s), 42, 43
- Select\_LGA (Macro)
  - Definition, 56
  - Reference(s), 58, 59
- Select\_NAV (Macro)
  - Definition, 44
  - Reference(s), 47, 48
- Select\_VGA (Macro)
  - Definition, 91
  - Reference(s), 93, 94
- Select\_VS (Macro)
  - Definition, 65
  - Reference(s), 67, 68
- Selected\_State (Type)
  - Definition, 23
  - Reference(s), 49, 55, 83, 90, 137, 138, 142, 143
- Side (Type)
  - Definition, 23
  - Reference(s), 29, 156
- Switch (Type)
  - Definition, 117
  - Reference(s), 118–126, 153
- THIS\_SIDE (Constant)
  - Definition, 24
  - Reference(s), 25, 30
- This\_Input (Input Interface)
  - Definition, 155
- This\_Input\_Msg (Message)
  - Definition, 152
  - Reference(s), 155
- This\_Output (Output Interface)
  - Definition, 159
- This\_Output\_Msg (Message)
  - Definition, 156
  - Reference(s), 159
- Track\_ALTSEL (Macro)
  - Definition, 78
  - Reference(s), 84
- Transfer\_Switch (Input Variable)
  - Definition, 125
  - Reference(s), 104, 155
- VAPPR (State Variable)
  - Definition, 89
  - Reference(s), 87, 90
- VAPPR\_Selected (State Variable)
  - Definition, 90
  - Reference(s), 87–89
- VGA (State Variable)
  - Definition, 94
  - Reference(s), 92, 93
- VS\_Lamp (State Variable)
  - Definition, 121
  - Reference(s), 158, 160
- VS\_Pitch\_Wheel\_In\_Motion (Input Variable)
  - Definition, 125
  - Reference(s), 109, 155
- VS\_Switch (Input Variable)
  - Definition, 120
  - Reference(s), 110, 155
- VS (State Variable)
  - Definition, 68
  - Reference(s), 66, 67
- When\_ALT\_Activated (Macro)
  - Definition, 75
  - Reference(s), 60, 69
- When\_ALT\_Switch\_Pressed (Macro)
  - Definition, 112
  - Reference(s), 112, 113
- When\_ALT\_Switch\_Pressed\_Seen (Macro)
  - Definition, 113
  - Reference(s), 27, 73
- When\_ALTSEL\_Activated (Macro)
  - Definition, 81
  - Reference(s), 60, 69
- When\_ALTSEL\_Capture\_Cond\_Met (Macro)
  - Definition, 97
  - Reference(s), 97
- When\_ALTSEL\_Capture\_Cond\_Met\_Seen (Macro)
  - Definition, 97
  - Reference(s), 78
- When\_ALTSEL\_Target\_Altitude\_Changed (Macro)
  - Definition, 113
  - Reference(s), 113, 114
- When\_ALTSEL\_Target\_Altitude\_Changed\_Seen (Macro)
  - Definition, 114

- Reference(s), 27, 73, 79
- When\_ALTSEL\_Track\_Cond\_Met (Macro)
  - Definition, 96
  - Reference(s), 96
- When\_ALTSEL\_Track\_Cond\_Met\_Seen (Macro)
  - Definition, 96
  - Reference(s), 78
- When\_AP\_Disconnect\_Switch\_Pressed (Macro)
  - Definition, 106
  - Reference(s), 106, 107
- When\_AP\_Disconnect\_Switch\_Pressed\_Seen (Macro)
  - Definition, 107
  - Reference(s), 130
- When\_AP\_Engage\_Switch\_Pressed (Macro)
  - Definition, 105
  - Reference(s), 105, 106
- When\_AP\_Engage\_Switch\_Pressed\_Seen (Macro)
  - Definition, 106
  - Reference(s), 130
- When\_APPR\_Switch\_Pressed (Macro)
  - Definition, 114
  - Reference(s), 109, 114, 115
- When\_APPR\_Switch\_Pressed\_Seen (Macro)
  - Definition, 115
  - Reference(s), 26, 27, 50, 51, 85, 86
- When\_Disengage\_AP (Macro)
  - Definition, 130
  - Reference(s), 132
- When\_Engage\_AP (Macro)
  - Definition, 130
  - Reference(s), 132
- When\_FD\_Switch\_Pressed (Macro)
  - Definition, 103
  - Reference(s), 96–100, 103
- When\_FD\_Switch\_Pressed\_Seen (Macro)
  - Definition, 103
  - Reference(s), 25, 26
- When\_FLC\_Activated (Macro)
  - Definition, 71
  - Reference(s), 60
- When\_FLC\_Switch\_Pressed (Macro)
  - Definition, 111
  - Reference(s), 111, 112
- When\_FLC\_Switch\_Pressed\_Seen (Macro)
  - Definition, 112
  - Reference(s), 27, 69, 70
- When\_GA\_Switch\_Pressed (Macro)
  - Definition, 115
  - Reference(s), 115, 116
- When\_GA\_Switch\_Pressed\_Seen (Macro)
  - Definition, 116
  - Reference(s), 26, 27, 56, 91, 130
- When\_HDG\_Activated (Macro)
  - Definition, 42
  - Reference(s), 35
- When\_HDG\_Switch\_Pressed (Macro)
  - Definition, 108
  - Reference(s), 108, 109
- When\_HDG\_Switch\_Pressed\_Seen (Macro)
  - Definition, 109
  - Reference(s), 26, 40
- When\_Lamp\_Changed (Macro)
  - Definition, 158
  - Reference(s), 159
- When\_LAPPR\_Activated (Macro)
  - Definition, 53
  - Reference(s), 35
- When\_LAPPR\_Armed (Macro)
  - Definition, 53
  - Reference(s), 45
- When\_LAPPR\_Track\_Cond\_Met (Macro)
  - Definition, 98
  - Reference(s), 98
- When\_LAPPR\_Track\_Cond\_Met\_Seen (Macro)
  - Definition, 98
  - Reference(s), 50
- When\_Lateral\_Mode\_Changed (Macro)
  - Definition, 157
  - Reference(s), 159
- When\_Lateral\_Mode\_Manually\_Selected (Macro)
  - Definition, 26
  - Reference(s), 25
- When\_LGA\_Activated (Macro)
  - Definition, 58
  - Reference(s), 35
- When\_NAV\_Activated (Macro)
  - Definition, 47
  - Reference(s), 35
- When\_NAV\_Armed (Macro)
  - Definition, 47
  - Reference(s), 51
- When\_NAV\_Switch\_Pressed (Macro)
  - Definition, 107
  - Reference(s), 105, 108
- When\_NAV\_Switch\_Pressed\_Seen (Macro)
  - Definition, 115
  - Reference(s), 115, 116

- Definition, 108
- Reference(s), 26, 44, 45
- When\_NAV\_Track\_Cond\_Met (Macro)
  - Definition, 95
  - Reference(s), 95
- When\_NAV\_Track\_Cond\_Met\_Seen (Macro)
  - Definition, 95
  - Reference(s), 44
- When\_Nonbasic\_Lateral\_Mode\_Activated (Macro)
  - Definition, 35
  - Reference(s), 37, 40, 45, 51, 56, 91
- When\_Nonbasic\_Vertical\_Mode\_Activated (Macro)
  - Definition, 60
  - Reference(s), 56, 62, 65, 70, 73, 79, 86, 91
- When\_Pilot\_Flying\_Transfer (Macro)
  - Definition, 29
  - Reference(s), 25, 40, 45, 51, 56, 65, 70, 73, 79, 86, 91
- When\_Selected\_Nav\_Frequency\_Changed (State Variable)
  - Definition, 127
  - Reference(s), 45, 51, 86
- When\_Selected\_Nav\_Source\_Changed (State Variable)
  - Definition, 127
  - Reference(s), 45, 51, 86
- When\_SYNC\_Switch\_Pressed (Macro)
  - Definition, 116
  - Reference(s), 116, 117
- When\_SYNC\_Switch\_Pressed\_Seen (Macro)
  - Definition, 117
  - Reference(s), 56, 91
- When\_Transfer\_Switch\_Pressed (Macro)
  - Definition, 104
  - Reference(s), 104
- When\_Transfer\_Switch\_Pressed\_Seen (Macro)
  - Definition, 104
  - Reference(s), 29
- When\_Turn\_FD\_Off (Macro)
  - Definition, 26
  - Reference(s), 28
- When\_Turn\_FD\_On (Macro)
  - Definition, 25
  - Reference(s), 28
- When\_Turn\_Modes\_Off (Macro)
  - Definition, 32
- Reference(s), 34
- When\_Turn\_Modes\_On (Macro)
  - Definition, 32
  - Reference(s), 34
- When\_VAPPR\_Activated (Macro)
  - Definition, 88
  - Reference(s), 60
- When\_VAPPR\_Track\_Cond\_Met (Macro)
  - Definition, 99
  - Reference(s), 99
- When\_VAPPR\_Track\_Cond\_Met\_Seen (Macro)
  - Definition, 99
  - Reference(s), 85
- When\_Vertical\_Mode\_Changed (Macro)
  - Definition, 157
  - Reference(s), 159
- When\_Vertical\_Mode\_Manually\_Selected (Macro)
  - Definition, 27
  - Reference(s), 25
- When\_VGA\_Activated (Macro)
  - Definition, 93
  - Reference(s), 60
- When\_VS\_Activated (Macro)
  - Definition, 67
  - Reference(s), 60
- When\_VS\_Pitch\_Wheel\_Rotated (Macro)
  - Definition, 109
  - Reference(s), 105, 110
- When\_VS\_Pitch\_Wheel\_Rotated\_Seen (Macro)
  - Definition, 110
  - Reference(s), 27, 70, 73, 79, 91
- When\_VS\_Switch\_Pressed (Macro)
  - Definition, 110
  - Reference(s), 110, 111
- When\_VS\_Switch\_Pressed\_Seen (Macro)
  - Definition, 111
  - Reference(s), 27, 65

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188		
<p>The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.</p> <p><b>PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.</b></p>					
1. REPORT DATE (DD-MM-YYYY) 01-06-2003		2. REPORT TYPE Contractor Report		3. DATES COVERED (From - To) 12/2000-5/2003	
4. TITLE AND SUBTITLE Flight Guidance System Requirements Specification			5a. CONTRACT NUMBER		
			5b. GRANT NUMBER		
			5c. PROGRAM ELEMENT NUMBER		
6. AUTHOR(S) Steven P. Miller Alan C. Tribble Timothy M. Carlson Eric J. Danielson			5d. PROJECT NUMBER NCC1-01001		
			5e. TASK NUMBER		
			5f. WORK UNIT NUMBER 728-30-10-03		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) NASA Langley Research Center Hampton, VA 23681-2199			8. PERFORMING ORGANIZATION REPORT NUMBER		
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) National Aeronautics and Space Administration Washington, DC 20546-0001			10. SPONSOR/MONITOR'S ACRONYM(S) NASA		
			11. SPONSOR/MONITOR'S REPORT NUMBER(S) NASA/CR-2003-212426		
12. DISTRIBUTION/AVAILABILITY STATEMENT Unclassified - Unlimited Subject Category 61 Availability: NASA CASI (301) 621-0390      Distribution: Standard					
13. SUPPLEMENTARY NOTES An electronic version can be found at <a href="http://techreports.larc.nasa.gov/ltrs/">http://techreports.larc.nasa.gov/ltrs/</a> or <a href="http://ntrs.nasa.gov">http://ntrs.nasa.gov</a> Langley Technical Monitor: Ricky W. Butler					
14. ABSTRACT This report describes a requirements specification written in the RSML-e language for the mode logic of a Flight Guidance System of a typical regional jet aircraft. This model was created as one of the first steps in a five-year project sponsored by the NASA Langley Research Center, Rockwell Collins Inc., and the Critical Systems Research Group of the University of Minnesota to develop new methods and tools to improve the safety of avionics designs. This model will be used to demonstrate the application of a variety of methods and techniques, including safety analysis of system and subsystem requirements, verification of key properties using theorem provers and model checkers, identification of potential sources mode confusion in system designs, partitioning of applications based on the criticality of system hazards, and autogeneration of avionics quality code. While this model is representative of the mode logic of a typical regional jet aircraft, it does not describe an actual or planned product. Several aspects of a full Flight Guidance System, such as recovery from failed sensors, have been omitted, and no claims are made regarding the accuracy or completeness of this specification.					
15. SUBJECT TERMS formal methods, requirements, mode confusion, flight guidance system, avionics, formal analysis, flight software					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			STI Help Desk (email: <a href="mailto:help@sti.nasa.gov">help@sti.nasa.gov</a> )
U	U	U	UU	176	19b. TELEPHONE NUMBER (Include area code) (301) 621-0390