**NASA Technical Memorandum** 102685

# Knowledge-Based System for Flight Information Management

Wendell R. Ricks

June 1990



National Aeronautics and Space Administration

**Langley Research Center**
Hampton, Virginia 23665

# Abstract

This thesis resulted from a research effort that explored the use of knowledge-based system (KBS) architectures to manage information on the primary flight display (PFD) of commercial aircraft. The information management strategy that was implemented tailored the information on the PFD to the tasks the pilot performed. This task-tailored approach to PFD information management required complex logic that led to difficult-to-manage software in earlier implementations using traditional programming techniques. Based on this problem, a KBS approach was chosen over the traditional programming approach. This decision was based, in part, on an earlier study that found KBS architectures easier to manage given complex decision logic. This thesis describes the KBS design and implementation of the task-tailored PFD information management application.

While working with the PFD information management system, an improvement to the system's functionality was made. In the logic used for task-tailored information management, knowledge of the phase of flight was necessary for correct operation. Needing this input required that the pilot track and enter this information (via cockpit switches) throughout the flight. This additional task for the pilot was not desirable, so an effort to automate the detection of flight phase was pursued. This thesis describes the knowledge acquisition and subsequent system design of the flight-phase detection KBS.

Since the task-tailored PFD information effort became the first study involving integrated KBS's running in LISP and executing in real time on a civil transport aircraft, a preliminary study to evaluate the feasibility of the KBS concept was performed. The objectives of this feasibility study were to test the resulting KBS's, collectively called the Task-Tailored Flight Information Manager (TTFIM), in flight, to verify their implementation, integration, and to validate the software engineering advantages in an operational environment. This thesis will discuss these flight tests and the subsequent results.

The results of the flight tests verified the feasibility of using KBS's for PFD management with actual data. Correct implementation and integration of the KBS with existing aircraft systems were evident by the correct mapping of KBS-dictated PFD formats with those generated by the traditional implementation. Flight tests were also successful in validating the logic used for flight-phase detection. The flight-phase detection logic was successful for all elements within the flight-test envelope except one. However, the cause of this one problem was easily isolated during the flight test given the KBS environment.

The process of implementing the KBS's for flight tests validated the software engineering advantages in an operational environment. Frequent modifications to TTFIM were necessary to achieve desirable performance. The KBS's built-in utilities enabled quick and easy modifications. This observation and positive programmer feedback validated the ability of a KBS approach to ease software maintenance. Another finding in favor of the software engineering advantages of the KBS approach was the programmer's ability to more easily develop initial systems (i.e., from scratch) using the KBS shell than with traditional programming techniques. And, the ease with which the one logic error in the flight-phase detection KBS was isolated during the flight tests was further evidence of the software engineering advantages of KBS architectures.

# Contents



# Acknowledgements


I would like to acknowledge the able technical support of James Ramsay, who managed the GoldWorks(TM) KBS software for the real-time aircraft experiment, and Kelly DeBure, who modified and maintained the aircraft software affected by the introduction of the KBS's.


# List of Acronyms

| | |
|---|---|
| CMP | Control Mode Panel |
| CR | Cruise |
| CRT | Cathode-Ray Tube |
| DATAC | Digital Autonomous Terminal Access Communication |
| EC | Enroute Climb |
| ED | Enroute Descend |
| EICAS | Engine Indication and Crew Alerting System |
| EPR | Engine Pressure Ratio |
| I/O | Input/Output |
| KBS | Knowledge-Based System |
| LaRC | Langley Research Center |
| LD | Land |

| | |
|---|---|
| NASA | National Aeronautics and Space Administration |
| PC | Personal Computer (TM) |
| PFD | Primary Flight Display |
| STAR | Standard Terminal Arrival Route |
| TC | Terminal Climb |
| TSRV | Transport Systems Research Vehicle |
| TTFIM | Task-Tailored Flight Information Manager |
| TX | Taxi |
| T/O | Takeoff |
| TD | Terminal Descend |
| VLDS | Visual Landing Display System |

# Introduction

The difficulties flight crews have experienced managing the large amount of information available in today's transport aircraft cockpit, and the trend to increase the amount of information in future cockpits, have made information management a primary avionics concern. Many airline incidents and accidents have been attributed to difficulties managing cockpit information. For example, a United Airlines DC-8 crash in 1978 during approach to Portland, Oregon, was attributed to an information management problem. The aircraft ran out of fuel while the crew was troubleshooting a landing gear problem. If the flight crew's attention had been focused on the fuel information at the right time, the accident may have been avoided.

While good information management has always been a concern, a new feature of the current generation of transport aircraft offers the potential for more information management problems than before. This feature is the cathode-ray tube (CRT) display. The almost unlimited flexibility of the CRT display lifted many of the restrictions imposed by the electro-mechanical instruments formerly used. Now, much more information can be presented on a CRT at any given time, and display formats can be altered to make room for even more pieces of information. This flexibility can lead to information management problems which yield factors that cause display clutter.

Some research efforts have already been made to reduce the information management problems of selected CRT display formats. For example, the Engine Indication and Crew Alerting System (EICAS) of the Boeing 757 and 767 aircraft reduced the information management problems on its target CRT by using a centralized caution and warning system to manage the engine information presented. With EICAS, only the parameters required to set and monitor engine thrust were displayed full time, while the remaining engine parameters were monitored and only presented when out of tolerance [Ford, 1982] [Ropelewski, 1982].

Another effort targeted at managing a specific CRT's information was the NASA, Langley Research Center (LaRC) research effort focussed on the primary flight display (PFD - see appendix A). Under this research effort, all information on the PFD necessary for the basic control of the aircraft (i.e., pitch, roll, airspeed, and altitude) was presented all of the times. Presentation of optional information (e.g., reference altitude, glideslope deviation, and vertical path) was tailored according to the task(s) the pilot performed, so that optional information was presented only when needed by the pilot. For example, when one of the pilot's tasks was to follow a glideslope signal, the PFD was configured with *Glideslope Deviation* guidance, instead of *Vertical Path* or *Reference Altitude* guidance.

This task-tailored approach to PFD information management required complex logic to automate. Originally, this logic was implemented using procedural programming techniques. However, this original implementation led to several software engineering disadvantages. The original procedural code was hard to trace, modify, and verify, and with each change to the logic, these problems were enhanced. Because of these problems, a knowledge-based system (KBS) approach was explored as an alternative to the traditional programming approach. The decision to use a KBS approach was based in part, on an earlier study that found KBS architectures easier to manage given complex decision logic [Ricks & Abbott,

1987]. This thesis describes the KBS design and implementation of the task-tailored PFD information management application.

Another aspect of this thesis dealt with an improvement to the information management system's functionality. For correct operation of the information management logic, knowledge of the phase of flight was needed. This required that the pilot (or test engineer) track and maintain this input through bezel switches (i.e., toggle switches) in the cockpit. This additional task for the pilot was not desirable, so an effort to automate the detection of flight phase was pursued. This thesis will discuss the approach taken to automate the flight-phase detection, and describe the resulting KBS.

Since this PFD information management effort resulted in the first study involving KBS's running in LISP in real time on a civil transport aircraft (i.e., the Transport Systems Research Vehicle - see appendix B) a preliminary study to evaluate the KBS concept was necessary. The objectives of this study were to design, implement, and test (in flight) the KBS approach to PFD information management, to determine the feasibility of addressing this problem with a KBS approach, to validate the flight-phase detection logic, and to confirm the software engineering advantages of the KBS approach while in an operational environment.

This paper is divided into four sections. The first section describes the PFD information management problem and resulting system design. The second section discusses the knowledge acquisition and construction of the flight-phase detection KBS. The third section describes the evaluation of the KBS concept (including the evaluation of the flight-phase detection KBS). And, the final section summarizes the results of this evaluation and lists recommendations for further research.

# Chapter 1 Task-Tailored PFD Information Management

## 1.1 Domain

In previous PFD research efforts, goals were not directed at managing the flow of information. In past efforts, each piece of information was simply given a location on the screen and presented whenever available. However, the increased amount of information targeted for the PFD made it difficult to present information based solely on availability. Continuing to dedicate space on the PFD for each piece of information would have contributed to factors that cause display clutter (e.g., display density). Display clutter increases the user's search time and inhibits the ability of the user to understand pertinent display information.

Therefore, an effort targeted at managing information on the PFD was initiated at NASA LaRC. Under this effort, the PFD information management philosophy presented all information necessary for the fundamental control of the aircraft (i.e., pitch, roll, airspeed, and altitude) at all times. It then tailored the optional information on the PFD according to the task(s) the pilot performed, so that optional information was presented only when needed by the pilot. For example, when one of the pilot's tasks was to follow a glideslope signal, the PFD was configured with *Glideslope Deviation* guidance, instead of *Vertical Path* or *Reference Altitude* guidance.

Providing only relevant information on the PFD was a logical approach to explore, since the more information the greater the competition among screen components for a person's attention. Visual search times would have been longer, and meaningful patterns more difficult to perceive if the screen flooded the user with too much information [Galitz, 1989]. The overall cleanliness of the display heightens the operator's ability to successfully perform his search and identify information. Because when an operator scans a display for a specific parameter (target), all other information on the screen is noise [Gilmore, 1985].

Since it was decided that only relevant information be presented, relevant PFD information was defined as information necessary and helpful for the fundamental guidance and control of the aircraft at any point in time. In determining the relevance of PFD information, the information was categorized as either basic or optional. Basic PFD information (e.g., altitude, airspeed, pitch, etc..) was defined as the information necessary for the fundamental guidance and control of the aircraft, and by definition, basic PFD information was always relevant. Optional PFD information (e.g., reference altitude, vertical path, etc...) was identified as the information that was helpful in performing certain guidance and control tasks, such as maintaining a specified altitude, or following a vertical path. However, optional information was not required for the fundamental guidance and control of the aircraft, nor was it always relevant to the pilot's current tasks. Since basic information was always relevant and therefore always displayed, optional PFD information was the target of the task-tailored PFD information management effort.

Following is a list of the optional information symbols managed by TTFIM. The optional information controlled by this system was Localizer Deviation, Horizontal Deviation, Track-Angle Error (1, 2, and 3), Vertical Path, Reference Altitude, Glideslope Deviation, Radar Altitude, Runway Image, Waypoint Star, Flare Guide, and Commanded Airspeed (1 and 2). Appendix C gives a detailed description of each piece of information.

The logic necessary to carry out this task-tailored approach to information management required input data from many sources. The information necessary to decide what optional PFD information to present consisted of the airplane's automatic control mode configurations, the cockpit switch settings, sensor and system information (e.g., signal availability and numerical sensor values), and the current flight phase. Descriptions of the input data can be found in appendix D.

As mentioned above, the tasks the pilot performed were used to determine what information was relevant and subsequently, what optional information to present. This mapping of tasks to relevant optional information was done implicitly. In other words, the tasks were not sought explicitly at run-time and then all information necessary for the tasks identified. Instead, the conditions that were true when a task was being executed were used for the conditions of the subsequent optional information rule. For example, if the airplane was in the landing phase of flight, with a valid glideslope signal, and the automatic land mode had been engaged on the control-mode panel (CMP), then the task of following a glideslope was being executed. Therefore, the guidance symbol on the altitude scale should represent the glideslope signal (i.e., Glideslope Deviation Symbol).

The rules used for the task-tailored management of optional information displays are described in detail in appendix E.

## 1.2 Problem

This task-tailored approach to PFD information management required complex logic to automate. Early attempts at implementing the complex logic with traditional programming techniques led to difficult to manage programs which proved costly in time and clarity. With the traditional implementation, logic pertinent to managing the display elements was hard to distinguish from other code. As a result, it also became increasingly harder to understand and modify the logic used to control the optional information presentation as the implementation progressed.

The software engineering problems were critical with the PFD management application since the vehicle for this system was a research environment and the frequency of logic changes was high. Since changes were frequently occurring to the logic, the costs of time and clarity mentioned above were amplified. Therefore, a new implementation approach was sought.

## 1.3 Approach

Because of the software engineering problems with the traditional implementation, a KBS approach was explored. The decision to use a KBS approach was based, in part, on an earlier study that found KBS architectures easier to manage given applications with complex logic [Ricks & Abbott, 1987]. In this earlier study the traditional implementation was more efficient in execution time, however, the KBS provided the potential for improving the productivity of the programmer and designer. In the study, modifications to the KBS implementation were found to be easier, more efficient, and less error-prone than with the traditional implementation. The homogeneous representation of the rule-base was found to be instrumental in code simplification and test-tool development needed during the verification process. The overall simplicity and modularity of the KBS were found to be more amenable to utilities that aided in the explanation of the system's execution.

Another factor contributing to the exploration of a KBS approach for this application was the successful use of KBS architectures for rapid prototyping. It has been found that rapid prototyping environments generated systems that simulated the important interfaces, and performed the main functions of the intended system [Rushby, 1988]. These features of rapid prototyping allowed early experience with, and direct testing of, the main aspects of the system's proposed functionality, thereby allowing much earlier and more realistic appraisals of the system's requirements specifications. Therefore, rapid prototyping environments fit neatly into the standard life-cycle model of software engineering. Rapid prototyping helped avoid the problem of making errors early and not detecting them until late in the life-cycle (a particularly costly and serious problem). Without rapid prototyping, missing or inappropriate requirements were hard to detect at an early stage. Systematic reviews (commonly used in non-rapid prototyping development) often detected inconsistent, or ambiguous requirements, but missing requirements generated no internal inconsistencies and often escaped detection until the system was actually built and tried in practice.

Another experimental comparison of a prototyping versus a traditional approach to software development [Boehm et al. 1984] found that both approaches

yielded approximately equivalent products, though the prototyping approach required much less effort (45% less) and generated less code. Since the products of rapid prototyping were developed incrementally, they were also considered easier to learn and use.

Based on the findings of these earlier studies, this application was implemented using a KBS approach in an attempt to reduce the software engineering problems found with the traditional implementation. The next section describes the system design. The system described made use of one KBS to implement the same system (functionality) previously implemented using traditional techniques.

## 1.4 System Design

The KBS implementation of the optional PFD information management effort was named the Task-Tailored Flight Information Manager (TTFIM), describing the functionality of the system. The overall system design is illustrated in Figure 1-1, and again with a data flow diagram in Figure 1-2.

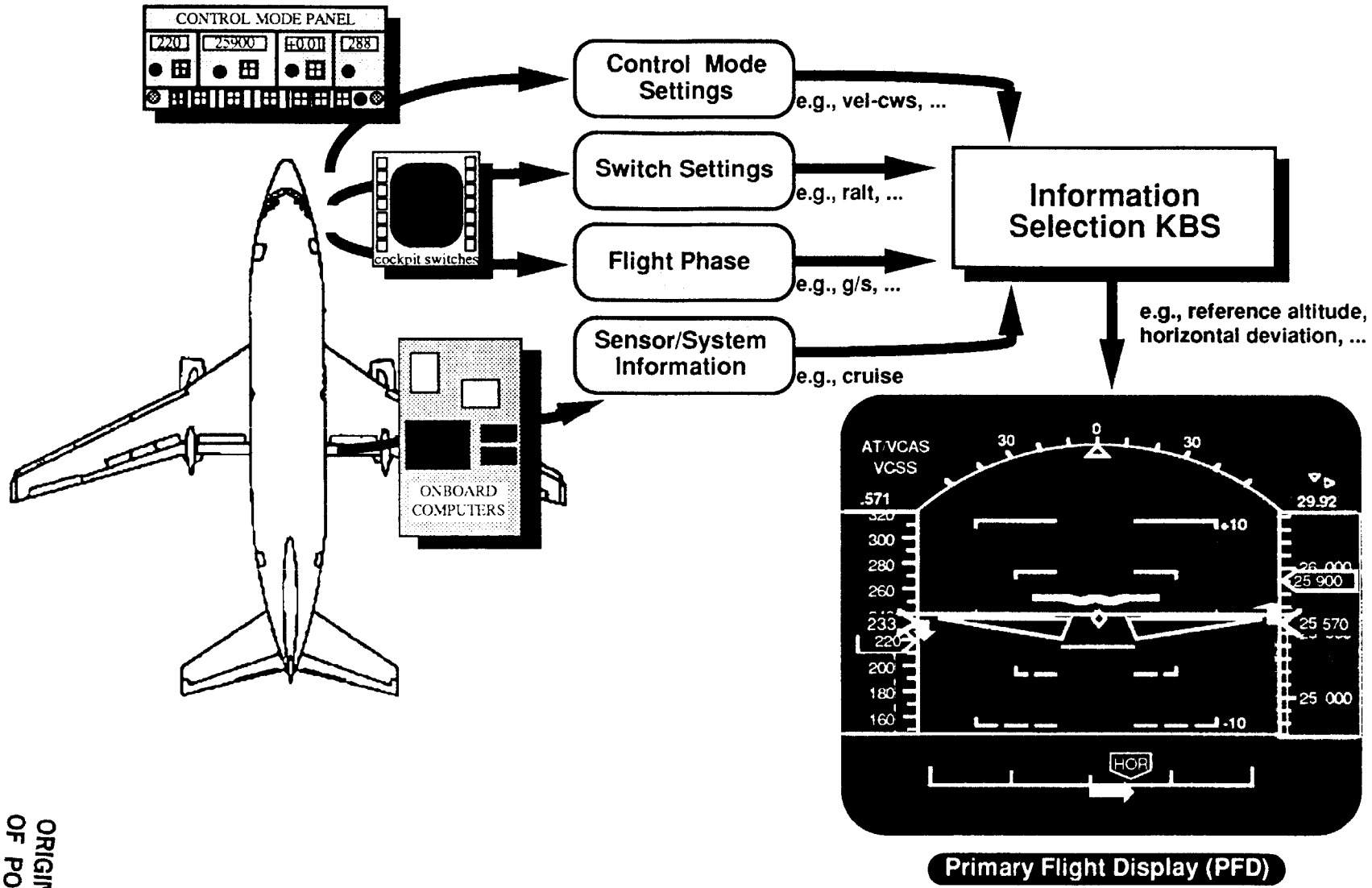



Figure 1-1 - System Design Without Automatic Flight Phase Detection

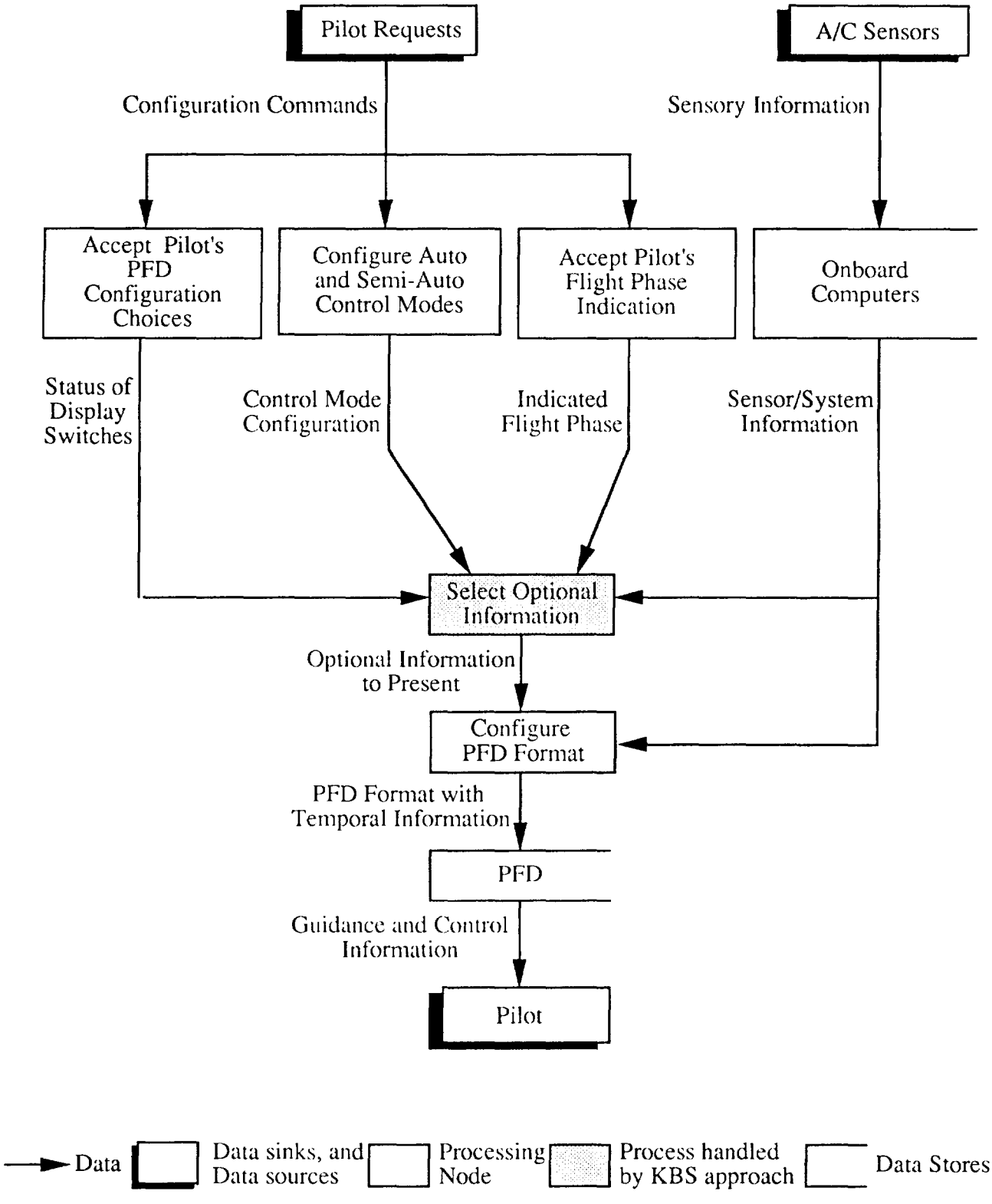



Figure 1-2 - Data-Flow Diagram of System Without Automatic Flight Phase Detection

Figure 1-2 shows that the final *control and guidance information* was provided to the pilot via the PFD. The *PFD formats* (including the dynamic information) were configured within the Displays computer (see appendix B). Configuration of the PFD was based on two sets of data. One set of data contained all the *sensor and system information* residing on the various aircraft computers. And, the second set of data identified the *optional information to present*. The optional information to present was generated by the task-tailored approach described above. The selection of optional information KBS based its decision on the *status of the display switches*, the *control mode configuration*, the *indicated flight phase*, and various *sensor/system information*. The pilot requests generated the configuration commands needed to determine the status of the display switches, control mode configuration, and phase of flight. The airplane sensors provided the data to the onboard computers.

Implementation and integration issues of the TTFIM flight software are discussed in appendix F.

# Chapter 2 Automatic Flight-Phase Detection

## 2.1 Problem

Another aspect of this thesis dealt with an improvement to the task-tailored PFD information management system's functionality. As illustrated in Figures 1-1 and 1-2, correct operation of TTFIM required indication of the current phase of flight as input. For the earlier implementations, this required that the pilot (or test engineer) track and maintain this input through bezel switches (i.e., toggle switches) in the cockpit. Additional tasks (like indicating flight phase) for the pilot were not desirable, so an effort to automate the detection of flight phase was pursued.

## 2.2 Approach

Through pilot interviews and piloted simulations in the Transport Systems Research Vehicle (TSRV) simulator [NASA SP-435, 1980] [Grove et al. 1986], a set of rules was derived for automatic flight-phase detection while in flight. The pilot interviews were conducted first to obtain a preliminary set of rules. These rules were then implemented in the TSRV simulator for further knowledge acquisition.

The initial pilot interviews were used to determine the number of different flight phases needed and to get a working set of rules which characterized these phases. As a starting point for these interviews, the current set of flight phases used by the pilots for manual entry into TTFIM was used: taxi, takeoff, climb, cruise, descend, and land. For better resolution, and for possible use of the automatic detection logic for other applications, four new phases were substituted for two of the former ones. The phases of terminal climb and enroute climb were substituted for climb. Similarly, enroute descend and terminal descend were substituted for descend.

The rules focussed on the fact that only certain phases can physically transition from one to another. For example, when the aircraft is taking off, the only possible phase transitions from takeoff were to either terminal climb, land, or

taxi. With these physical limits on the possible transitions, the rules for each phase transition only needed to be able to distinguish itself from its transition neighbors, thus minimizing the number of conditions required for each transition.

To minimize the possibility of ambiguity in the flight-phase detection logic, the rules were defined as "transition-in" rules. This meant that the conditions required for a flight phase to be active only needed to be true to start that phase, not to stay in that phase. Once in a specified phase of flight, the only way the system would transition into another phase of flight was if the conditions for another phase became true (not if the conditions of the current were no longer true). For example, one of the conditions in the rule for takeoff stated that the flight phase in the previous cycle was either taxi or land (i.e., Last-Phase = TX or LD). One cycle after the phase of flight became takeoff, the "Last-Phase" variable was bound to takeoff (i.e., T/O), thus no longer satisfying the condition stating that the last phase must be either taxi or land. However, the phase of flight remained takeoff until the conditions of another phase became true.

The vehicle used for further knowledge acquisition was the TSRV simulator. The TSRV simulator was a fixed-based cockpit configured as the research cockpit of the TSRV airplane (see appendix B). The simulation included a six-degree-of-freedom set of nonlinear equations of motion, and functionally represented the aspects of the advanced flight control configuration of the airplane. The research cockpit is characterized by eight, 9-inch diagonal, color display units.

For this study, the TSRV simulator was also connected to the Visual Landing Display System (VLDS). The VLDS was a camera/model-board system for generating a visual out-the-window scene for the pilot of a simulated aircraft. The system consisted of a dual-scaled terrain model, a series of lamps to illuminate the model, a three-degree-of-freedom translation system to position the camera, and a three-degree-of-freedom optical/rotational system mated to a color television camera. The VLDS provided non-composite RGB television signals to an external simulator cockpit window display device to give a field of view of 48 degrees horizontally, by 36 degrees vertically [Grove et al. 1986]. The VLDS provided the "out-the-window" scenes necessary for the taxi, takeoff, terminal descend, and land phases of this study.

The flight-phase detection logic was coded in a module that ran in the background of the TSRV simulation. No functionality of the simulation (e.g., display configurations) was affected by the introduction of this logic with the exception of a coded number on one of the cockpit displays to show the experimenter what phase of flight the logic detected.

For the evaluation and further knowledge acquisition of the flight-phase detection rules, seven pilot subjects were used. Three of the subjects were NASA test pilots, one subject was a pilot for the United States Navy, one subject was a Army Reserve pilot, and the remaining two were NASA employees - one with an Airline Transport rating, and the other with commercial and instrument ratings. Each subject was briefed prior to the simulation study with respect to the display formats, the aircraft cockpit systems, and the evaluation task.

The simulator evaluation began after the pilot briefing. Many of the evaluation sequences were as follows:

1. *Simulator familiarization and practice flights* - Because no demands were placed on the subjects that were specific to the simulated aircraft, and pilot performance was not a measure of concern in this study, the simulator familiarization and practice flights used the same scenario programmed for the evaluation (i.e., same flight plan).

2. *Full mission flight (i.e., from taxi at Norfolk International to taxi at Richmond International) with discrete inputs from the subjects being recorded* - The subjects were briefed regarding the flight phases identified in this study and were asked to indicate when they thought they were making a transition from one phase to another by keying the trigger on the side-stick controller. The input from the pilots were compared against the transition times generated by the automatic flight-phase detection logic.

3. *Full mission flight with the simulation being frozen at each phase transition for subjective evaluation* - At each flight-phase transition the simulation was frozen and the subject was given the opportunity to evaluate the current phase transition qualitatively.

4. *Flights consisting of aborted takeoffs, touch-and-go's, and other deviations from the flight plan* - These deviations were not pre-programmed and left up to the pilot as to how they were carried out. Most of these flights were frozen at each phase transition for further subjective evaluation.

Because the pilots' performance was not a measure, and no statistical significance was sought, deviations from the above sequence were allowed. The results sought in each subject evaluation were to either validate the set of rules being used or to identify changes that needed to be made. Valid changes to the logic were made between each subject evaluation. Errors in the logic often surfaced with one pilot and not another due to differences in their flying styles and training biases. The final set of rules used in simulation were representative of each of the evaluations.

The overall result of this evaluation process were the set of rules taken to the TSRV aircraft for flight tests. These rules are discussed in appendix G.

## 2.3 System Design

The resulting system design with the addition of the new KBS for flight-phase detection is illustrated in Figure 2-1, and again with the data flow diagram in Figure 2-2. The overall system design is the same as illustrated in Figure 1-2, and described in section 1.4 with the exception of the new KBS that *detected the phase of flight.* With this change, the pilot requests and subsequent bezel switch settings are not needed to supply the phase of flight input to the information selection KBS. Instead, the phase of flight is determined using the *sensor/system information* and the logic described above.

Implementation and integration issues of the TTFIM flight software are discussed in appendix F.

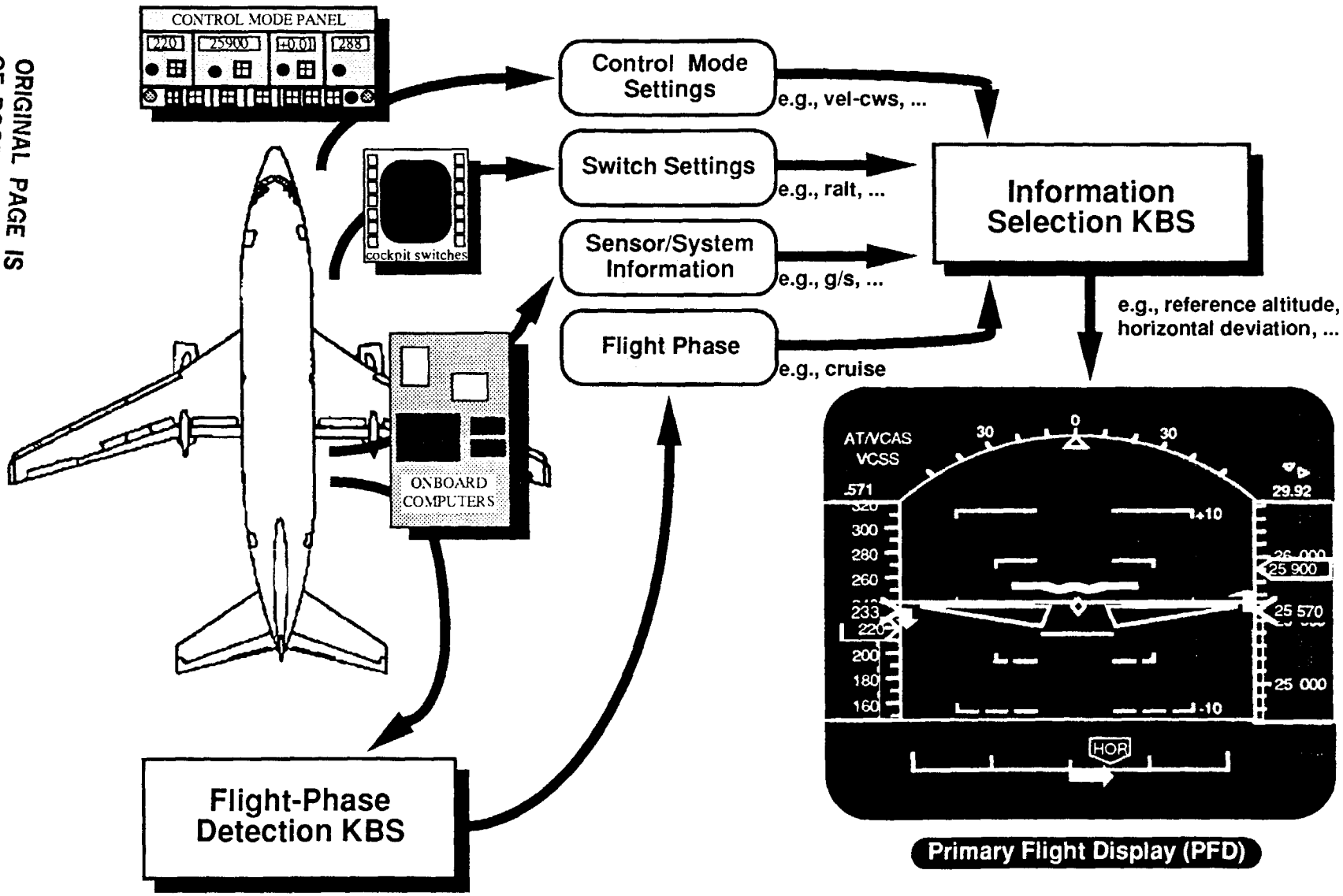



Figure 2-1 - System Design With Automatic Flight Phase Detection

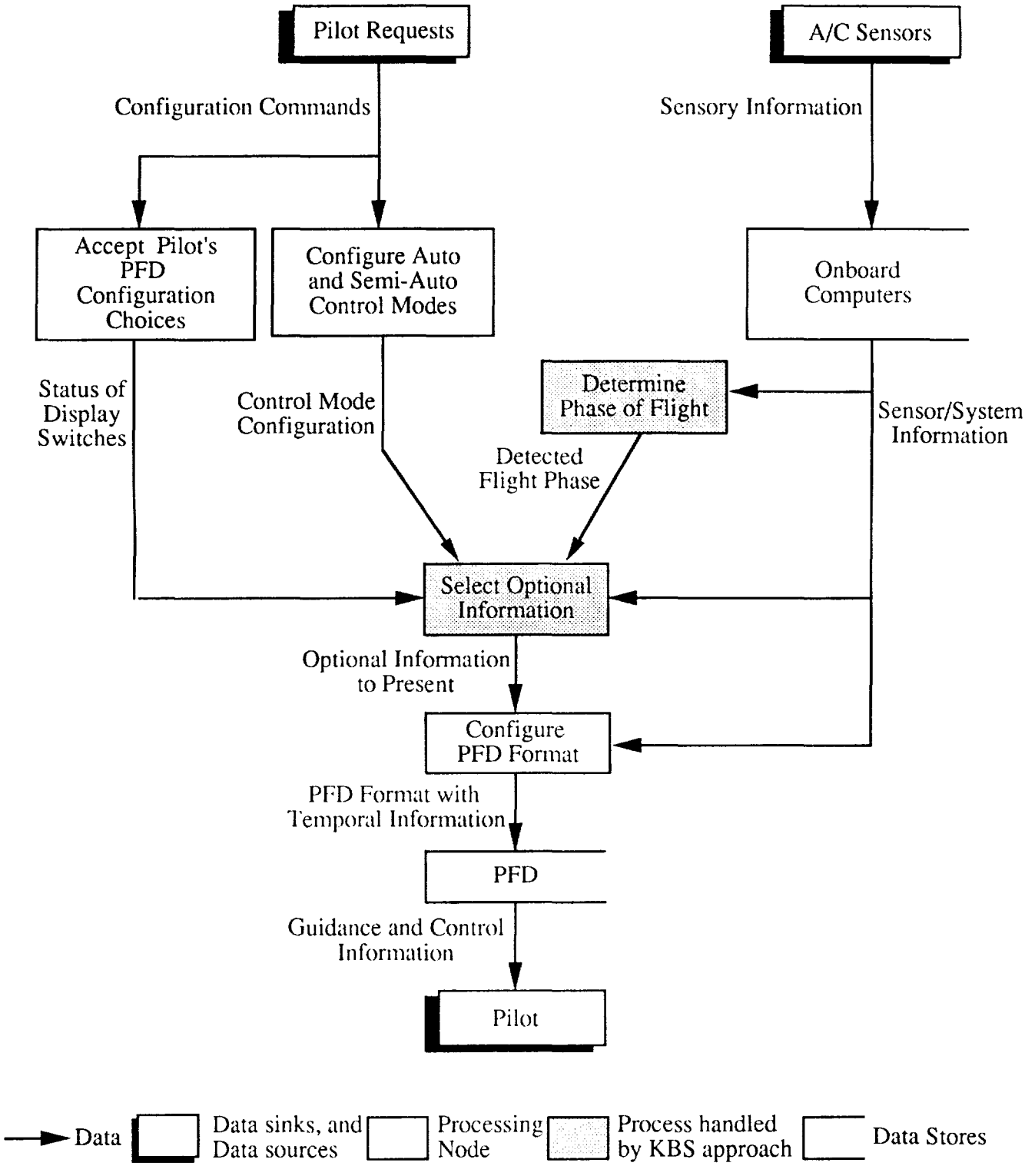



Figure 2-2 - Data-Flow Diagram of System With Automatic Flight Phase Detection

# Chapter 3 Evaluation

The task-tailored PFD information management effort was the first study involving KBS's running in LISP, and executing in real-time on the TSRV airplane. With the advent of such a drastic implementation change, feasibility issues were a concern. Therefore, a study to evaluate the feasibility of approaching the PFD information management problem with a KBS approach was performed. The implementation and integration of the software for the feasibility study were also used to validate the software engineering advantages of the KBS approach in an operational environment.

TTFIM was evaluated onboard the TSRV aircraft in two stages. The objective of the first stage of flight tests was directed only at testing the KBS that selected the display elements to present, to assess the feasibility of the KBS approach. For the first stage of flight tests, no functional changes from the traditional baseline implementation were desired. For these initial flights, the flight engineer manually entered the flight-phases as they occurred (see Figure 1-2, page 11). See appendix H for a description of the flight-test envelopes used for the stage 1 evaluation.

For the second stage of flight tests, the objectives were to validate the flight-phase detection logic, and to evaluate the addition and integration of the new KBS (see Figure 2-2, page 18). These tests were done with a flight-test envelope consisting of multiple repetitions of each flight phase represented in the KBS, and multiple transitions between the flight phases. See appendix I for descriptions of the flight-test envelope used for the stage 2 evaluation.

**STAGE 1 Evaluation - Optional Information Selection KBS**

The intent of the first stage evaluation was to assess the feasibility of implementing the PFD information management application using a KBS for real-time operation onboard the TSRV airplane. For this evaluation, the KBS implementation was intended to duplicate the functionality of the traditional implementation. There were no characteristics of TTFIM in this stage that changed the functionality of the PFD information management from what the traditional implementation did on the airplane. Therefore, a successful evaluation of the stage 1 tests was defined as a KBS implementation and integration that duplicated the performance of the traditional system.

As mentioned earlier, flight tests were used to verify the implementation and integration of TTFIM onboard the TSRV. The traditional code implementation was used in the flight tests as a basis for comparison. Both pilot feedback and comparisons between KBS display elements and expected display elements were used in the post-flight analysis.

The test pilot for this study had flown many hours in the TSRV research cockpit, and was familiar with the behavior of the PFD when driven by the traditional implementation. Therefore, pilot feedback was used to note deviations on the PFD from what was expected. Pilot comments during the flight tests were manually recorded for post-flight analysis. While major irregularities did not occur, the pilot did notice that the first appearance of some of the optional information was slower with the KBS approach.

Throughout the flight tests, short delays (of a few seconds or less) were noted with the first appearance of a few optional information display elements. The increase in time needed to initiate some of the optional information formats was attributed to the slower nature of the LISP language and hardware, and the simple addition of a new module in the TSRV data communications (remember that the traditional implementation was embedded in the Displays computer graphics calls). Now extra steps were required to retrieve the input information, process the information, and then send the information to the Displays computer for formatting.

Delays were also noted in the comparison data that were recorded. Comparison data were recorded throughout the flight in the form of discrete display control words (see Table 3-1 below). Two discrete words were sufficient to represent each of the optional information elements driven by TTFIM (words 0 and 1). When a bit in a control word was set (i.e., equal to one), the relative display element was active. For example, when the second and fourth bits in control word zero were set and the remaining bits were zero, then Horizontal Deviation and Glideslope Deviation were the only active elements of word one.

TABLE 3-1

Output Display Control Words

| Display Control Word | Bit | Indication |
|---|---|---|
| 0 | 0 | Reference Altitude |
| | 1 | Waypoint Star |
| | 2 | Horizontal Deviation |
| | 3 | Glideslope Deviation |
| | 4 | Localizer Deviation |
| | 5 | CAS Reference (dial) |
| | 6 | CAS Reference (buffer) |
| 1 | 0 | Runway Image |
| | 1 | Radar Altitude |
| | 2 | Vertical Path |
| | 3 | Flare-Guide |
| | 4 | Track-Angle Error 1 |
| | 5 | Track-Angle Error 2 |
| | 6 | Track-Angle Error 3 |

The comparison data for the stage 1 tests consisted of the display elements active under both the traditional and KBS implementation - each implementation generated its own display control words. Even though the optional information on the PFD was being driven by the KBS, both the traditional and KBS implementation were generating display control words for post flight analysis. As with the pilot feedback, the only deviations noted in the post flight analysis were the small delays the KBS experienced when updating some of the active display elements.

Even though some update delays occurred with the KBS implementation, flight operations were not interrupted. Feedback from the pilot was positive, and the KBS display elements were equivalent to the expected display elements. These

results confirmed TTFIM's implementation and integration, and thus validated the feasibility of the KBS concept for implementation of PFD information management.

**STAGE 2 Evaluation - Flight-Phase Detection KBS**

The second stage of tests were directed at validating the flight-phase detection logic, and verify the implementation and integration of the KBS onboard the TSRV. The implementation and integration of the new KBS with the other KBS and TSRV systems were evaluated using the same measures as the previous study - the PFD performance (assessed again by pilot evaluation and comparison data). Since the automatic flight-phase detection KBS was not designed to change the PFD performance, but rather to eliminate the need for the pilot to enter it manually, the performance of the PFD would be the same as the traditional implementation if the flight-phase detection KBS was implemented and integrated correctly.

Validating the flight-phase detection logic was done by comparing the phases detected with those expected. Two additional control words were added to help evaluate the detection of flight phase KBS. These control words are defined in Table 3-2 below.

TABLE 3-2

TTFIM Output Control Words

| Display Control Word | Bit | Indication |
|---|---|---|
| 2 | 0 | Takeoff |
| | 1 | Terminal Climb |
| | 2 | Cruise |
| | 3 | Terminal Descent |
| | 4 | Land |
| | 5 | Taxi |
| | 6 | Enroute Climb |
| | 7 | Enroute Descend |
| 3 | 0 | Error Flag (for flight phase) |

Only one phase was true at one time, therefore only one bit in word 2 was set at one time. The error bit (word 3) was set when errors were reported by the KBS. The control word indicating the phase of flight was decoded and displayed on the screen during the flight. This presentation of flight-phase was used by the test engineer to note whether the logic was detecting current phases as it should. Video recordings of the PFD were also used in post flight analysis.

Correct PFD configurations verified the implementation and integration of the flight-phase detection KBS. The evaluation of the flight-phase detection logic was successful for all elements within the flight-test envelope (see appendix I) except one. At one point in the flight-test envelope, the test called for a "touch-and-go" where the KBS was to detect the transition from land to takeoff. However, a transition to taxi occurred due to an erroneous value given for the flap settings in

the rules. This problem was easily isolated given the KBS environment. With the correction to the flight-phase detection KBS being made, the tests were considered successful.

**Software Engineering Evaluation**

The implementation and integration of the flight test KBS's were also used to validate the software engineering advantages of KBS's identified in a previous study [Ricks & Abbott, 1987] while in an operational environment. In the development and maintenance process, frequent modifications to TTFIM's rules were needed to achieve correct performance. The KBS programming environment's built-in utilities enabled quick, easy, and efficient modifications.

The KBS environment also provided routines for explaining the execution, and producing information needed to verify performance. These features helped during the initial development and in explaining system performance during the flight tests. Positive programmer feedback and the additional data point of isolating the logic error in the flight-phase detection KBS during the flight tests was further evidence of the software engineering advantages of KBS architectures.

# Chapter 4 Concluding Remarks

This thesis resulted from a study at NASA LaRC that is exploring effective ways of managing information on the PFD of commercial aircraft cockpits. The current information management strategy being explored determines when to present information on the PFD by the tasks the pilot performs. This task-tailored approach to PFD information management requires complex logic that led to software engineering problems when traditional procedural programming techniques were used. Based on these software engineering problems, a KBS approach was chosen over the traditional programming approach. This decision was based, in part, on earlier studies that found KBS architectures easier to manage given complex logic.

While working with the PFD information management system, an improvement to the system's functionality was made. In the logic used for task-tailored information management, knowing the *phase of flight* was necessary for correct operation. In the original procedural implementation, the need for this input required that the pilot enter the phase of flight (via cockpit switches) throughout the flight. Adding this task for the pilot was not desirable, so the detection of flight phase was automated within this effort.

Since the task-tailored PFD information effort was the first study to involve KBS's running in LISP in real time on the TSRV aircraft, feasibility issues surfaced. Therefore, a preliminary study to evaluate the feasibility of the KBS concept for this flight application was performed. The objectives of the study were to test the resulting KBS's in flight, to verify their implementation and integration, and to validate the software engineering advantages in an operational environment.

The results of the flight tests verified the feasibility of using KBS's for PFD management with actual data. Correct implementation and integration of the KBS with existing aircraft systems were evident by the correct mapping of KBS-dictated

PFD formats with those generated by the traditional implementation. The only irregularity noted during the flight tests by both pilot feedback and recorded comparison data were the small delays that the KBS caused when changing some of the PFD information formats (e.g., from reference altitude guidance to glideslope guidance). However, these delays were very slight, and did not interrupt flight operations.

Flight tests were also successful in validating the logic used for flight-phase detection. Validation of the flight-phase detection logic was done by tracking the phases detected with those expected. The evaluation of the flight-phase detection logic was successful for all elements within the flight-test envelope except one. However, the cause of this one problem was easily isolated during the flight test given the KBS environment. Correct mapping of PFD formats during the validation of the flight-phase detection logic also verified the integration of the flight-phase detection KBS with the KBS system for selection of PFD formats.

The process of implementing the KBS's for flight tests provided the information necessary to confirm the software engineering advantages of KBS architectures in an operational environment. Frequent modifications to TTFIM were necessary to achieve desirable performance. The KBS's built-in utilities enabled these modifications to be done quickly and easily. This observation and positive programmer feedback validated the ability of a KBS approach to ease the task of software maintenance. Another finding in favor of the software engineering advantages of the KBS approach was the programmer's ability to more easily develop initial systems (i.e., from scratch) using the KBS shell than with traditional programming techniques. The ease with which the one logic error in the flight-phase detection KBS was isolated during the flight tests was even further evidence of the software engineering advantages of KBS architectures.

An auxiliary contribution of this thesis resulted from the process of preparing the KBS software for flight tests onboard the TSRV airplane. The experience gained during this process will ease the effort required to take future systems requiring AI-based implementation techniques to the TSRV. Another contribution was that the flight-phase detection logic used in this study can be use by other studies requiring flight-phase input. And, the KBS architecture developed for this task will also ease the future exploration of PFD information management efforts by providing an software platform more amenable to logic modifications. For future work, pilot evaluations of the task-tailored PFD information management philosophy are planned. Additionally, plans are being made to perform a sensitivity analysis and an ambiguity evaluation of the flight-phase detection logic.

# Appendix A
# Primary Flight Display (PFD)

The PFD (see Figure A-1) provides a pilot with information necessary for guidance and control of an aircraft. Studies have shown great potential in reducing pilots visual work load with electronically generated PFD's (e.g., [Steinmetz, 1986], [Abbott et al. 1987a] and [Abbott et al. 1987b]). The installation of electronically generated PFD's in some of the Boeing, McDonnell Douglas, Airbus, and other aircraft families characterize an increasing trend toward using these PFD's.

The PFD format in the TSRV shows the current aircraft attitude and provides other critical state information to the pilot. Refer to the PFD format drawing, Figure A-2. The area in the center of the screen is referred to as the PFD view window. Within the window a number of symbols appear that depict aircraft roll, pitch, yaw, flight path angle, angle of attack, and track error. Three dimensional representations of the waypoint and the destination runway are displayed in the window along with a flare guidance cue and alert messages.

Angular perspective in the window is provided by the pitch grid and horizon ticks. The pitch grid has a double solid line representing the horizon which separates the sky from the ground, along with parallel grid bars spaced in 5 degree increments. Along the horizon line, tick marks are spaced to show 10 degree steps of horizontal displacement. The other window symbology is interpreted against the grid and ticks to ascertain proper angular readings. The area from the horizon line to the top of the view window is raster filled in blue to easily distinguish the sky/ground boundary formed by the horizon line. At the top of the window along the arc is a roll scale which uses a triangular pointer to designate current aircraft roll angle. The roll angle corresponds to the amount of rotation applied to the horizon line within the view window.

On either side of the view window are gray raster filled rectangular areas called the airspeed and altitude tapes. They have tick marks and numeric values which can slide vertically giving the appearance of rolling measurement tapes.

The airspeed tape, on the left side of the view window, has the current aircraft airspeed value in the blacked out area at the center of the tape. A yellow pointer box containing a reference airspeed value may also appear at the appropriate spot on the tape when certain conditions arise. When airspeed is changing an elongated arrow will grow from the tape center vertically along the outside of the tape ticks and point to the airspeed that will be reached in ten seconds at the current rate of acceleration. Also along the same edge of the airspeed tape are a pair of wedge markers that bound a range of airspeeds suggested for the current aircraft flap settings and gross weight.

On the right hand side of the view window is the altitude tape. Similar to the airspeed tape, the current airplane altitude is shown in the blacked out area at the tape center. A yellow pointer box can also be displayed on the altitude tape representing selected altitude, glideslope deviation, or vertical deviation depending on the conditions. Flight path angle error is determined from the length of the white arrow that grows from the center of the altitude tape. When the glideslope error symbol is being displayed on the altitude tape, three magenta dots appear along the left edge of the tape to be used as a glideslope deviation scale. The last

item connected with the altitude tape is the radar altitude symbol. This red and white wedge shaped symbol is placed on the tape at the point that corresponds to the height of the ground above sea level.

On the top of the display area above the tapes is the phase of flight indicator on the left, and the baroset value and wheel/column detent pointers on the right. Below the altitude tape is a rectangular box containing the decision height value. Directly under the view window is the horizontal deviation scale and the flight information and status boxes. The horizontal scale has two reference markers. The white arrow that expands along the bottom of the scale shows track angle error while the pointer box on the top is used for horizontal deviation, localizer deviation, or crosstrack.

Along either side of the display screen are small squares adjacent to the sixteen bezel buttons. The squares indicate the current state of the bezel buttons, small green squares indicate "ON" and smaller magenta ones for "OFF". Normally (however experiments may redefine) the top six bezel buttons on the left hand side are used to choose the phase of flight mode for the PFD format. The bottom two bezels are the "cas" and "message" buttons which enable the display of the commanded airspeed and alert messages respectively. The eight buttons on the right hand area are also used to enable certain symbols. The first six enable the following pointers; reference altitude, vertical deviation, glideslope, horizontal deviation, crosstrack, and localizer. The last two are used to select the perspective runway symbols and waypoint star.

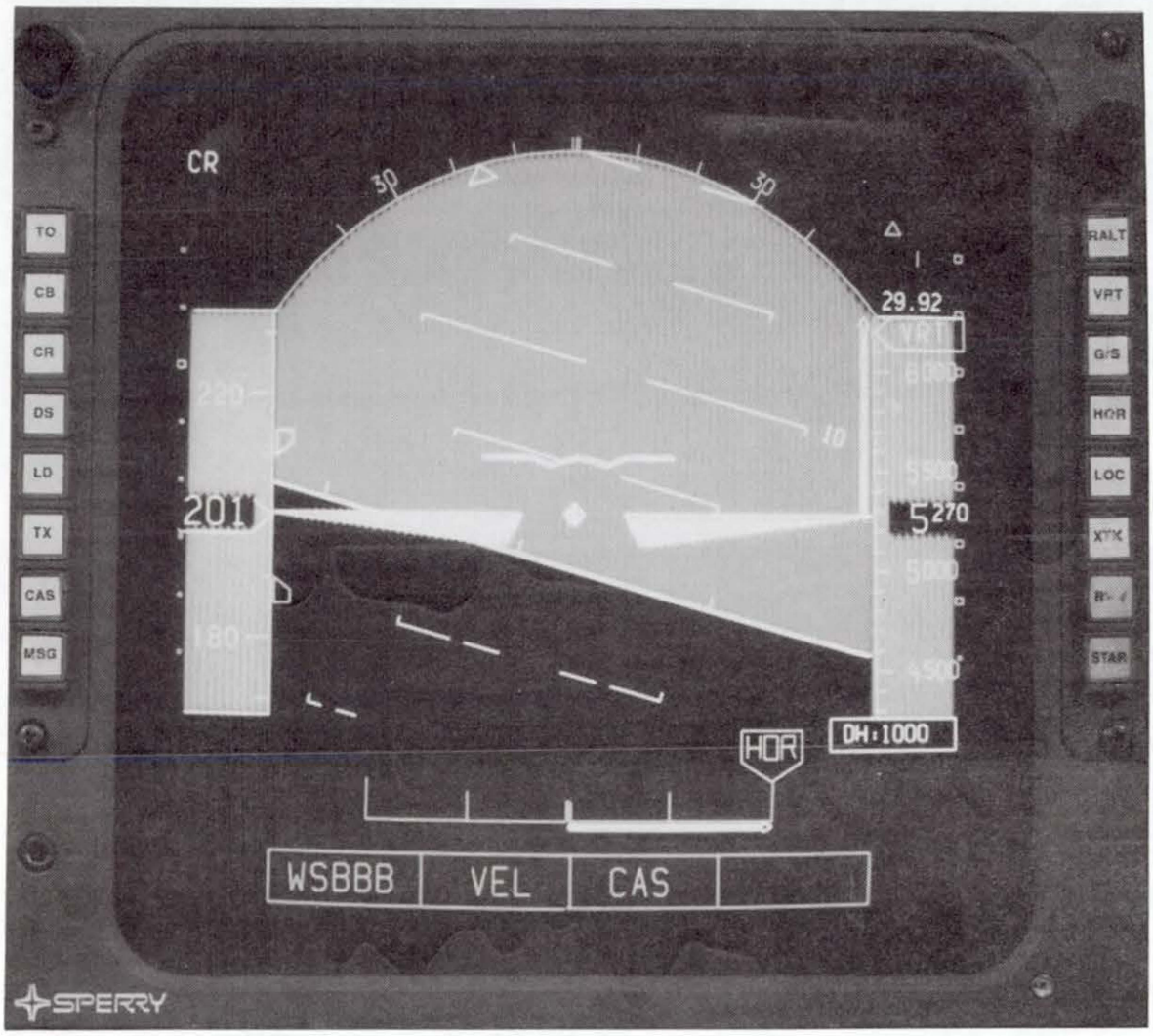
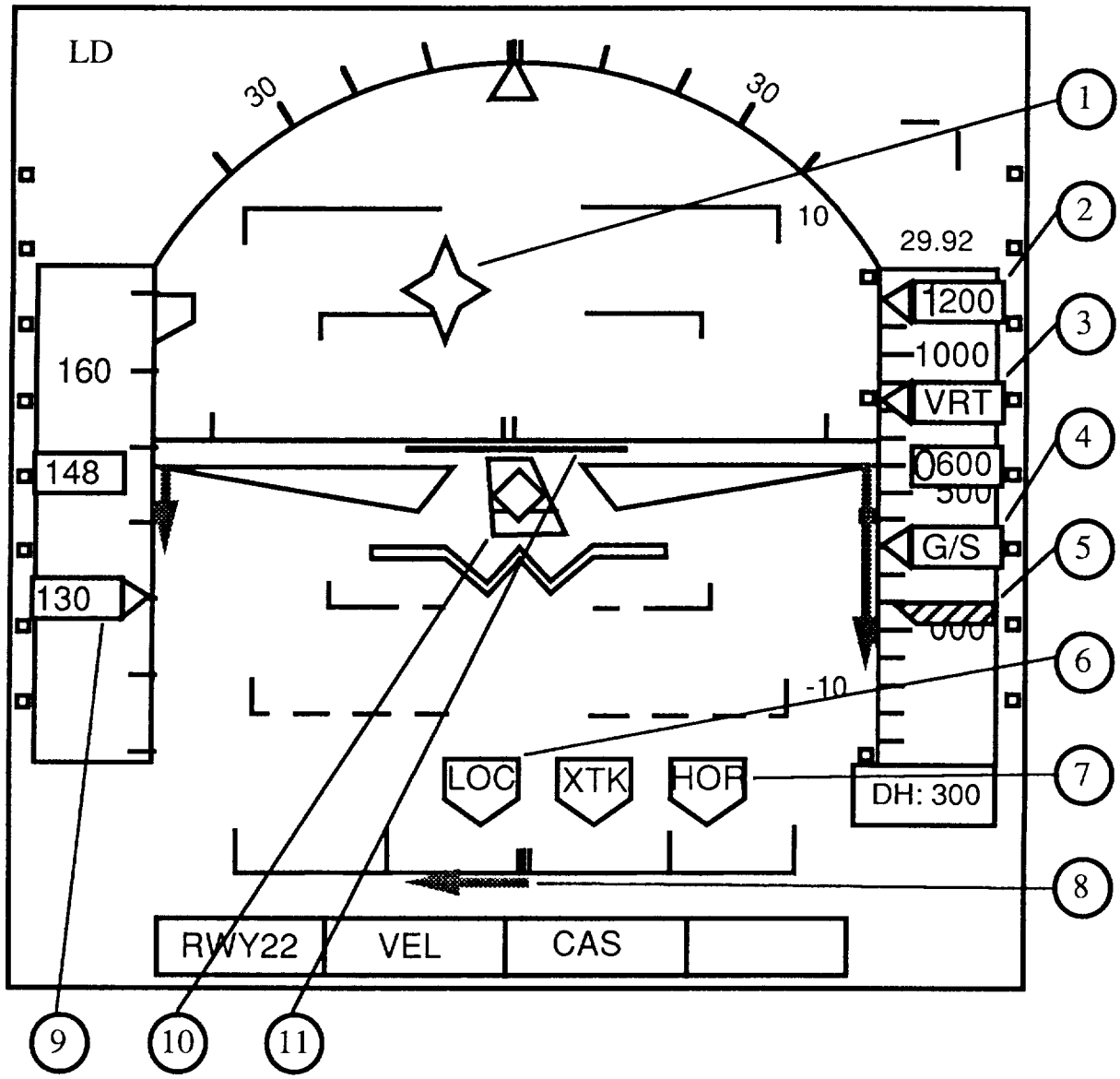



Figure A-1 - The Primary Flight Display




1 - Waypoint Star
2 - Reference Altitude
3 - Verticle Path
4 - Glideslope
5 - Radar Altitude
6 - Localizer
7 - Horizontal Path
8 - Track-Angle Error
9 - Reference Air Speed
10- Runway
11- Flare Guidance

Figure A-2 - PFD With Optional Information Formats

# Appendix B
# Transport Systems Research Vehicle (TSRV)

The TSRV [NASA NR87-48, 1987] is a specially instrumented Boeing 737 (a twin-engine subsonic commercial jet transport), with two flight decks (see Figures B-1 and B-2). The forward flight deck is a conventional Boeing 737 flight deck used for operational support and safety backup. The aft flight deck is a fully operational research flight deck positioned in the aircraft's cabin.

The experimental systems consist of triplex digital flight control system, a digital navigation and guidance system, and an electronic CRT display system located in the aft (research) flight deck [Knox & Cannon, 1980]. The full-scale research flight deck is located in the airplane cabin just forward of the wing as shown in Figure B-2. Figure B-3 shows the instrument panel of the research flight deck.

The triply redundant digital flight control system is driven by the controls computer. It provides both automatic and fly-by-wire control wheel steering options. One advance control mode, velocity vector control wheel (stick) steering mode, has the flight control computers (see Figure B-2) vary pitch attitude and heading to maintain flight-path angle and track angle, respectively [Knox & Cannon, 1980].

The navigation computer (see Figure B-2) is a general-purpose digital computer designed for airborne computations and data processing tasks. Major software routines in the navigation computer include navigation position estimate, flight route definition, guidance commands to the flight control computer system, and flight data storage for navigation purposes.


Figure B-1 - Transport Systems Research Vehicle

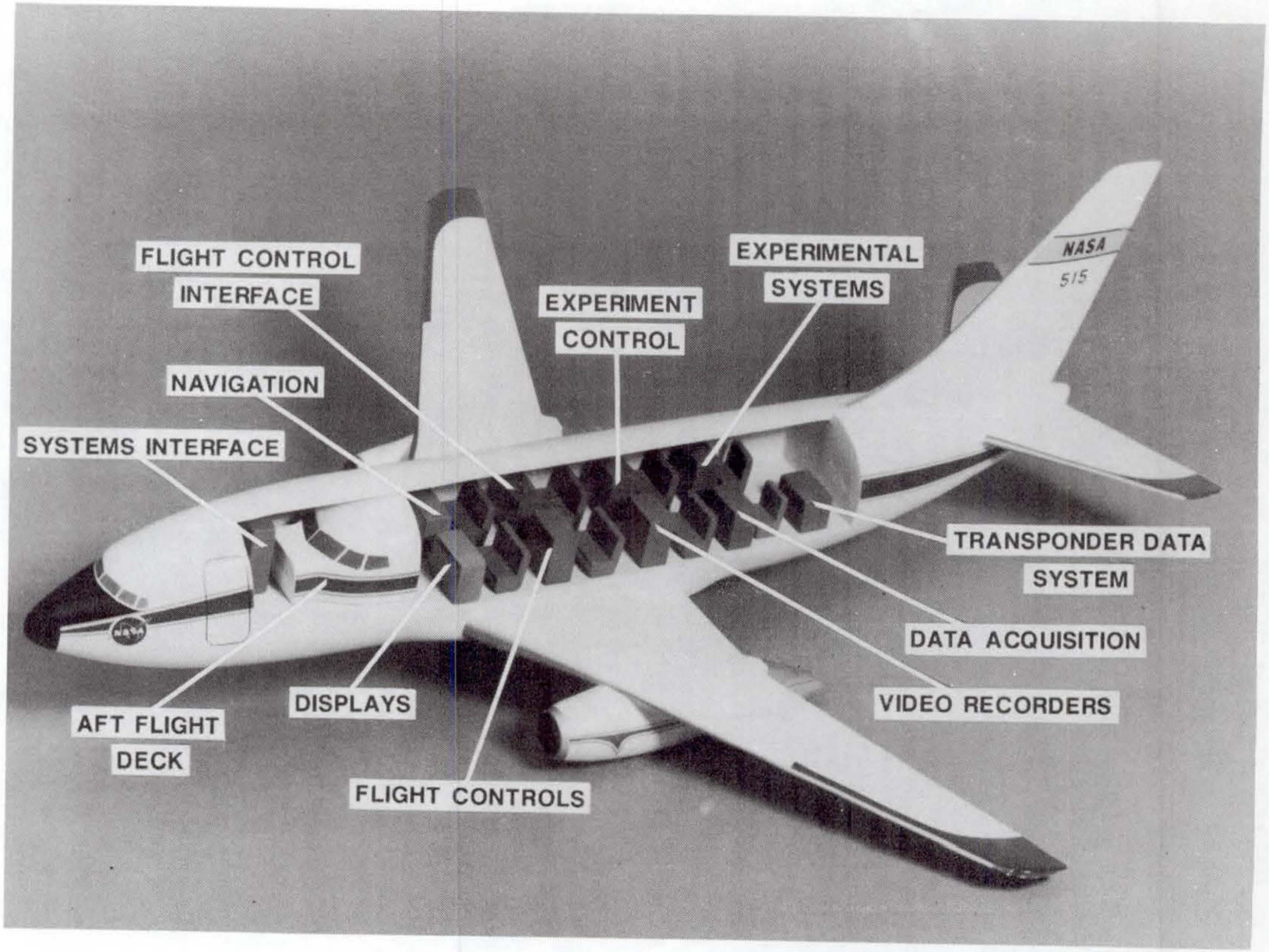



Figure B-2 - Layout of the Transport Systems Research Vehicle

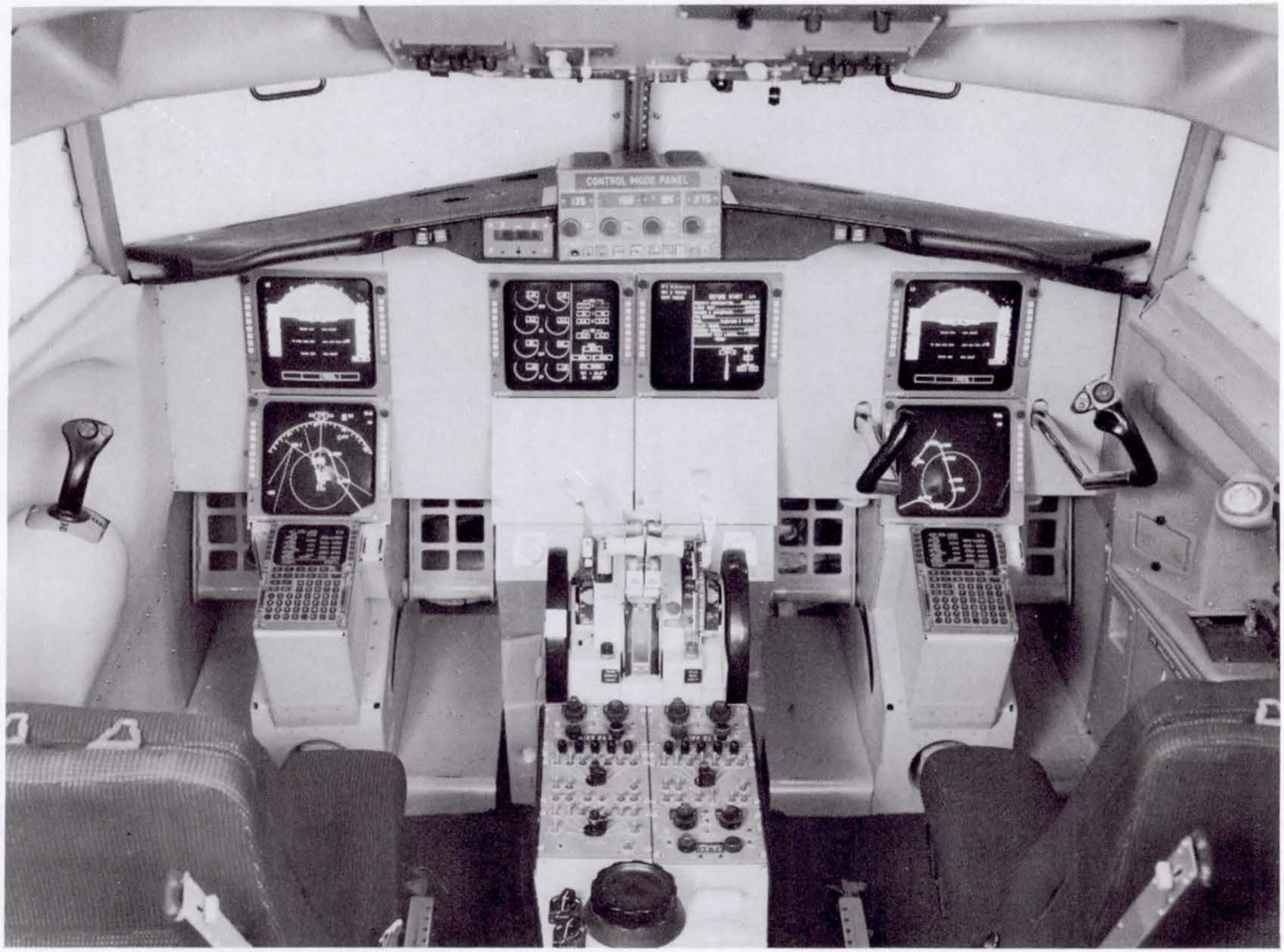



Figure B-3 - Instrument Panel of the TSRV Research Flight Deck

# Appendix C
# Optional PFD Information

This appendix lists the optional information managed by TTFIM along with a description of each piece of information. The optional information controlled by this effort were Localizer Deviation, Horizontal Deviation, Track Error 1-3, Vertical Path, Reference Altitude, Glideslope Deviation, Radar Altitude, Runway Image, Waypoint Star, Flare Guide, and Commanded Airspeed 1-2.

**Localizer Deviation**, located at the bottom of the PFD, is used for horizontal guidance where a localizer signal is used for the horizontal track. The Localizer Deviation symbol indicates horizontal deviation from the localizer beam in degrees.

**Horizontal Deviation**, located at the bottom of the PFD, was horizontal guidance where the reference path was either commanded by the pilot (via the CMP) or from the navigation computer (pre-programmed). The Horizontal Deviation pointer indicated horizontal flight path deviation in feet.

**Track Error** located at the bottom of the PFD, was horizontal guidance using an arrow to indicate the difference between actual track and either the dialed in track, the track in the navigation computer, or the runway. **Track Error 1** used the runway heading as the path reference. **Track Error 2** used the track commanded by the pilot (via the CMP). And, **Track Error 3** used the path in navigation computer.

**Vertical Path** symbology was located on the right hand side of the view window on the altitude tape. Vertical Path was for vertical guidance where the reference path was either commanded by the pilot (via the CMP) or in the navigation computer.

**Reference Altitude** symbology was located on the altitude tape. Reference Altitude was vertical guidance where the reference was either commanded by the pilot (via the CMP) or in the navigation computer.

**Glideslope Deviation** symbology was located on the altitude tape. Glideslope Deviation guidance was like Vertical Path guidance with the exception of the path being defined by a glideslope signal.

**Radar Altitude** was another item connected with the altitude tape. This symbol was placed on the tape at the point that corresponded to the height of the ground. If desired by the pilot, Radar Altitude can be displayed whenever valid. Radar Altitude is sometimes called Runway Altitude since it shows the placement of the runway above sea level.

**Runway Image** was displayed as a three dimensional image in the middle of PFD view window. Runway Image was used as secondary vertical and horizontal guidance. The runway symbol has a horizontal line across it to indicate the touchdown point on the runway. Like Radar Altitude, Runway Image can be displayed whenever valid and desired by the pilot.

**Waypoint Star** was displayed as an image in the middle of the PFD view window. Waypoint Star was a three dimensional visual reference of the destination waypoints within the PFD view window as defined in the navigation computer.

Waypoint Star guidance was used for both vertical and horizontal guidance. Waypoint Star guidance was displayed whenever valid and desired by the pilot.

**Flare Guide** symbol was displayed in the middle of the PFD view window when the appropriate conditions occur during the landing phase. The flare guidance cue rose from the bottom of the screen toward the bore sight of the gamma wedge symbol. Once the cue reached the bore sight, the pilot began the flare maneuver and in doing so, kept the cue and sight joined as a single unit. To clean up the screen when the flare guide symbol was used, no horizontal or track angle error guidance was displayed.

**Commanded Airspeed** was located on the left hand side of the view window on the airspeed tape. Commanded Airspeed provided a reference for (as the name implies) airspeed. **Commanded Airspeed 1** used as a reference, the input from the pilot (i.e., the CMP). **Commanded Airspeed 2** used as a reference, the speeds given for waypoints in the navigation computer. The only competition with Commanded Airspeed guidance is between the source of the reference. Either the pilots reference will be used, or that in the navigation computer.

# Appendix D
# Input for Information Selection KBS

This appendix lists the input information used by TTFIM to select the optional information to present on the PFD, along with a brief description. The input information consisted of control mode configurations, cockpit switches, sensor and system information, and the current flight phase. At the end of this appendix is a brief listing of the information used to determine the phase of flight.

**Control mode configurations** were any of the many combinations available on the control mode panel (see Figure D-1 and [NASA SP-435, 1978]). The lower left-hand section of the CMP provided selection of attitude control-stick steering, velocity vector control-stick steering, or automatic flight path control. The lower center and right-hand sections provided selection of the type of automatic path guidance. The four top sections provided hold select, and preselect operation of automatic airspeed, altitude, flight-path-angle, and track-angle modes. Most buttons on the CMP were 4-level buttons indicating either off, preselect, arm, or engaged status. The dials on the CMP were used to input reference airspeeds, altitudes, flight path angles, or horizontal track. The combinations of buttons statuses and dialed input, relative to themselves and each other define the control mode configuration.

**Cockpit switches** were any of the bezel switches located in the cockpit. The switches useful to the TTFIM system were the ones the pilot used to impose his choice of display configuration overrides, and were located on both sides of the PFD screen. The switches corresponding to the optional information formats (total of 9) were two-value bezel switches that indicated whether the pilot wanted the particular piece of information or not. These switches reflected the pilot's preference regarding the following symbols: Reference Altitude, Vertical Path, Glideslope Deviation, Horizontal Deviation, Localizer Deviation, Crosstrack Deviation, Runway Image, Waypoint Star, and Commanded Airspeed. The remaining switches corresponded to the flight phases and were either on or off, and only one could be on at any given time.

**Sensor and system information** consisted of information in the navigation computer (e.g., signal availability), sensory values, and various numerical and boolean data residing in other systems onboard the TSRV. The sensor and system information looked at by TTFIM were whether not a glideslope signal was valid, whether or not a localizer signal was valid, whether or not another waypoint existed to be displayed, whether or not the waypoint was within the displayable range, whether or not runway information was available, aircraft altitude (both barometric and radar), decision height, aircraft offset from horizontal path, runway length, aircraft heading, runway heading, and aircraft track.

**Current flight phase** was the most interesting of the inputs to TTFIM. Needing to know the flight phase meant that either the pilot would have to input the information as it changed, or some means of automating the detection needed to be pursued. Both methods were employed. For the first stage of the implementation, the pilot had to manually input the phases using the bezel switches on the left side of the screen, then the flight-phase detection was automated to achieve the "human-centered" objectives discussed in this paper.

When requiring the pilot to input the phase of flight, TTFIM only used six phases: taxi, takeoff, climb, cruise, descend, and land (see the bezel switches on the left of the PFD - Figure A-1). For automatic flight-phase detection, eight phases were used: taxi, takeoff, terminal climb, enroute climb, cruise, enroute descend, terminal descend, and land. The data used to detect flight phases were the squat switch status, gear status, epr value, gamma value, flaps settings, reverser status, radar altitude, barometric altitude, and the previous flight phase.

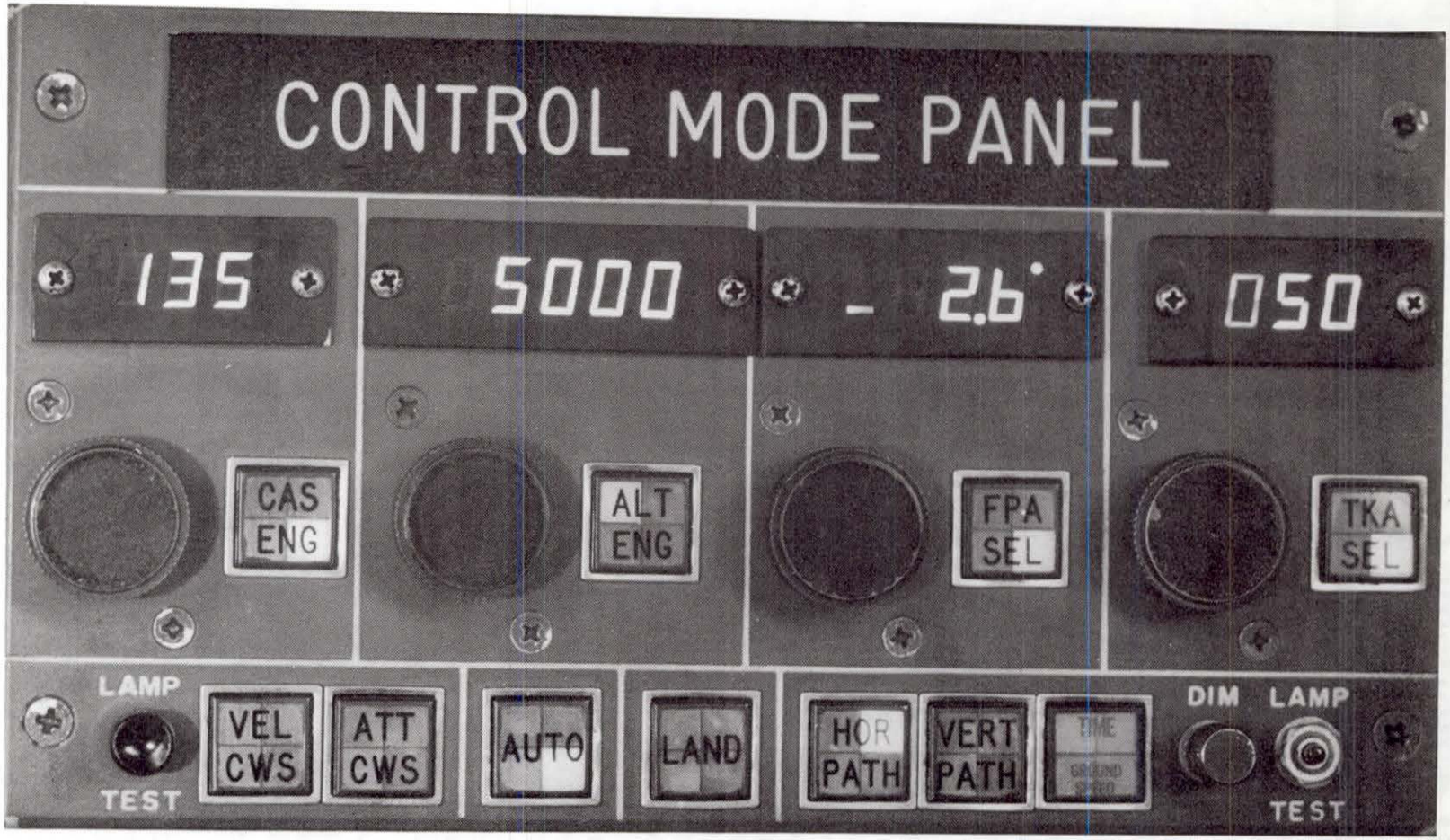



Figure D-1 - Control Mode Panel

# Appendix E
# Information Management Logic

The following logic represents that used in the TTFIM flight tests for determining the optional information to present. The intent of these rules were to duplicate the logic used in the traditional baseline implementation. Therefore, even when simplifications could be made, they were not. The rules appear here as they were in the flight tests.

HORIZONTAL DEVIATION
if flight phase is not TX
HOR display switch is on
horizontal track in navigation computer is valid
FLARE GUIDE symbol is not active
TRACK-ANGLE ERROR {2} symbol is not active
LOCALIZER DEVIATION symbol is not active
radar altitude is greater than 260'

HORIZONTAL DEVIATION
if flight phase is not TX
HOR display switch is on
horizontal track in navigation computer is valid
FLARE-GUIDE symbol is not active
TRACK-ANGLE ERROR {2} symbol is not active
LOCALIZER DEVIATION symbol is not active

TRACK-ANGLE ERROR {2}
if flight phase is TX or T/O
XTK display switch is on
automatic track angle mode is preselected or engaged
automatic horizontal path mode is not armed
FLARE GUIDE symbol is not active

TRACK-ANGLE ERROR {2}
if automatic track angle mode is preselected or engaged
XTK display switch is on
automatic horizontal path mode is not armed
FLARE-GUIDE symbol is not active
radar altitude is greater than 260'

TRACK-ANGLE ERROR {4}
if flight phase is LD
XTK display switch is on
localizer is valid
FLARE-GUIDE symbol is not active
radar altitude is grater than 260'

TRACK-ANGLE ERROR { }
if flight phase is T/O
XTK display switch is on
horizontal track in navigation computer is valid

FLARE-GUIDE symbol is not active
TRACK-ANGLE ERROR {2} symbol is not active
TRACK-ANGLE ERROR {4} symbol is not active

TRACK-ANGLE ERROR { }
if flight phase is not TX
XTK display switch is on
horizontal track in navigation computer is valid
FLARE-GUIDE symbol is not active
TRACK-ANGLE ERROR {2} symbol is not active
TRACK-ANGLE ERROR {4} symbol is not active
radar altitude is greater than 260'

VERTICAL PATH
if VRT display switch is on
vertical path in navigation computer is valid
GLIDESLOPE DEVIATION symbol is not active
REFERENCE ALTITUDE symbol is not active

REFERENCE ALTITUDE
if flight phase is not TX
automatic altitude hold mode is preselected, armed, or engaged
automatic land mode is armed or engaged
GLIDESLOPE DEVIATION symbol is not active

REFERENCE ALTITUDE
if flight phase is not TX
RALT display switch is on
automatic altitude hold mode is preselected, armed, or engaged
automatic enable mode is engaged
automatic track angle mode is engaged
automatic flight path angle mode is engaged
GLIDESLOPE DEVIATION symbol is not active

REFERENCE ALTITUDE
if flight phase is not TX
RALT display switch is on
automatic altitude hold mode is preselected, armed, or engaged
automatic attitude control (a-cws) mode is engaged
or automatic velocity vector control (v-cws) mode is engaged
GLIDESLOPE DEVIATION symbol is not active

REFERENCE ALTITUDE
if flight phase is not TX
RALT display switch is on
automatic track angle mode is engaged or h-path mode is engaged
automatic enable mode is engaged
automatic altitude hold mode is engaged
GLIDESLOPE DEVIATION symbol is not active

GLIDESLOPE DEVIATION
if flight phase is LD or TD
G/S display switch is on
automatic attitude control (a-cws) mode is engaged

or automatic velocity vector control (v-cws) mode is engaged
glideslope signal is valid

GLIDESLOPE DEVIATION
if flight phase is LD or TD
G/S display switch is on
glideslope signal is valid
automatic enable mode is engaged
automatic flight path angle mode is engaged

GLIDESLOPE DEVIATION
if flight phase is LD or TD
G/S display switch is on
glideslope signal is valid
automatic enable mode is engaged
automatic land mode is armed or engaged

RADAR ALTITUDE
if flight phase is TD, TC, EC, ED, or LD
radar altitude is less than 1300'

RUNWAY IMAGE
if flight phase is TD or LD
RWY display switch is on
runway is within coverage cone
runway in navigation computer is valid
a/c is within coverage cone
altitude is less than or equal to 5000'

WAYPOINT STAR
if STAR display switch is on
horizontal track in navigation computer is valid
there is another waypoint in the navigation computer
waypoint is within range

FLARE GUIDE
if flight phase is LD
automatic velocity vector control (v-cws) mode is engaged
radar-alt is less than the decision height

FLARE GUIDE
if flight phase is LD
automatic velocity vector control (v-cws) mode is engaged
radar-alt is less than 200'

LOCALIZER DEVIATION
if flight phase is LD or TD
LOC display switch is on
localizer signal is valid
FLARE-GUIDE symbol is not active
radar-alt is greater than 260'

COMMANDED AIR SPEED {1}
if automatic commanded airspeed hold mode is preselected or engaged

COMMANDED AIR SPEED {2}
if automatic time path mode is engaged
CAS display switch is on
last waypoint in navigation computer is false
COMMANDED AIR SPEED symbol {1} is not active

# Appendix F
# Implementation and Integration Issues

TTFIM's final system design was implemented and integrated for flight tests onboard NASA, Langley's TSRV aircraft (see appendix B) using a commercial knowledge-based expert system development shell . Implementation and integration issues of the flight software are discussed below in sections that describe the software development environment, the system's knowledge base, the system's inference mechanism, the operation protocol, and integration modifications that were necessary to the existing software modules.

## The Software Development Environment

Both KBS's (refer to Figure 2-2, page 18) onboard the TSRV were developed and maintained with the Gold Hill Computers' GoldWorks (TM) expert system shell. GoldWorks was a knowledge-based expert system development environment integrated with Gold Hill's Golden Common LISP Developer software. The GoldWorks system had two interfaces to accommodate different user needs and areas of expertise. The menu-based interface allowed non-LISP programmers to develop the system without using LISP. Whereas, more experienced developers were able to work with GoldWorks' open architecture through the Developer interface - a functional interface.

## The Knowledge Base

As with most KBS's, the primary implementation concern was the knowledge base. Everything the KBS knew about the application was contained in the knowledge base. TTFIM's knowledge base consisted of both *passive* and *active* knowledge pertaining to task-tailored PFD information management (both information selection and flight-phase detection). Passive knowledge were **facts** known to be true *a priori*, while active knowledge were any methods (e.g., rules, daemon functions, etc.) used to make, delete, and modify facts during run-time.

In TTFIM, passive knowledge was used for initializations. For example, the assertions on page 77 were passive knowledge that initialized, among other things, the phase of flight. And, in TTFIM, one use of active knowledge was the rules for detecting the phase of flight. For example, the rule for takeoff on page 81 would assert the fact (phase next takeoff) when the facts supporting the following conditions were true: in auto-detect mode; the previous phase of flight was taxi or land; the engine reversers are not engaged; epr greater than 1.8; flaps are set less than or equal to 30 degrees; and, radar altitude is less than or equal to 400 feet.

All facts (resulting from both passive and active knowledge) were represented in GoldWorks with *assertions*. In addition to the facts (also called *patterns*), an assertion contained the fact's dependency information. The dependency information of an assertion recorded how the assertion was put into the assertion base (the derivation) and why the assertion was currently in the assertion base (the justification or logical support). At any one time, the assertion base contained all the current factual information about TTFIM (see Figure F-1 below for an example snapshot of a partial TTFIM assertion base during run-time).

(TRUE)
(EQUIV TAKEOFF 1)
(EQUIV TERM-CB 2)
(EQUIV CRUISE 4)
...
(EQUIV ENR-DS 128)
(DETECT AUTO)
(SHIFT OUTPUT)
(G/S VALID-IS ON)
(LOC VALID-IS ON)
(NAV-PATH VALID-IS ON)
(NAVPATH2 VALID-IS ON)
(RALT SWITCHED ON)
(VERT SWITCHED ON)
(G/S SWITCHED ON)
...
(CAS SWITCHED ON)
(LOC SWITCHED ON)
(ALT MODE-IS 1)
(TKA MODE-IS 1)
(A-CWS MODE-IS 1)
(AUTO MODE-IS 1)
...
(LAND MODE-IS 1)
(FPA MODE-IS 1)
(V-CWS MODE-IS 4)
(CAS MODE-IS 2)
(DEC-HEIGHT IS 1000)
...
(ALTITUDE IS 100)
(EPR IS 185)
(CAS-REF-BUF SYMBOL OFF)
(FLARE-GUIDE SYMBOL OFF)
(RAD-ALT SYMBOL OFF)
(RWY-IMAGE SYMBOL OFF)
(LOC-DEV SYMBOL OFF)
...
(G/S-DEV SYMBOL OFF)
(REF-ALT SYMBOL OFF)
(VERT-PATH SYMBOL ON)
(WP-STAR SYMBOL ON)
(XTK-DEV SYMBOL ON)
(HOR-DEV SYMBOL ON)
(CAS-REF-DIAL SYMBOL ON)
...
(GEAR DISCRETE-IS ON)
(NOW-IS IN-PHASE TAKEOFF)
(NOW-IS NOT-IN-PHASE TAXI)

Figure F-1 - Example Snapshot of a portion of TTFIM Assertion Base

The justification and logical dependency portion of the assertions were used to support assertion *retraction*. When an assertion was removed from the assertion base, the system used the justifications and logical dependency information of the assertion and removed or retracted all assertions that logically depended upon it from the assertion base. When the retraction of one assertion was to cause the retraction of another assertion, the rule that led from one assertion to the other was defined with a *dependency* value of "t". For example, the rule for horizontal deviation on page 87 of appendix J had a dependency value of t. So, when any of the assertions needed to fire the horizontal deviation rule were retracted (e.g., **(now-is in-phase takeoff)**), the assertion **(hor-dev symbol on)** would be retracted. On the other hand, if an assertion was not to be retracted when conditions that put the assertion in the assertion base were retracted, it was defined with the dependency value of nil. An example of this was the rule for slave-phase on page 79 of appendix J, in which the dependency value was nil. When the assertion **(shift detect)** of this rule was retracted, it did not retract the assertion generated by the rule (i.e., **(now-is in-phase ?phname)**, **(shift output)**, or **(phase-out is ?phnum)**).

Assertion retraction was also enabled when an assertion relation was defined as *functional*. The special characteristic of a functional assertions was that when one functional assertion had the same elements as another functional assertion except for the last element in the list, it caused the first assertion to be retracted. For example, DETECT (page 76, appendix J) was defined as a functional assertion relation. If **(detect auto)** was asserted first, then **(detect manual)** was asserted later, the assertion **(detect auto)** was retracted.

TTFIM's assertion base was also modified by daemon functions. TTFIM made use of GoldWorks daemon capability to perform overhead operations (e.g., re-initializing values). For example, the when-modified facets of frame instances allowed a LISP function to be attached. Whenever a slot value associated with the daemon function was modified (asserted, retracted or modified), the system evaluated the LISP functions in the order listed. For example, the instance SYMBOL (page 74 of appendix J) defines when-modified daemons to each of its slot values to indicate a modification to the output module (new show) and to re-initialize the symbols to off (off-set).

## The Inference Engine

While the knowledge base contained the information specific to TTFIM, the inference engine contained the facilities that caused the system to make inferences about the data. The inference engine was responsible for applying the active knowledge to the factual data (i.e., assertions) when searching for solutions. Pattern-matching was used to match the antecedents and/or consequents of defined methods (e.g., rules, daemon functions, etc.) to assertions in the assertion base.

Several inferencing techniques were available in GoldWorks (i.e., forward chaining, backward chaining, or a combination of forward and backward chaining). In addition, GoldWorks allowed the control of the inference process to be altered by the use of priorities. TTFIM consisted primarily of data-driven production rules. The forward-chaining of TTFIM was used to infer solutions from assertions that existed in the assertion base. The forward-chaining was initiated when the antecedent, or "if," portion of a forward rule matched a set of assertions in the knowledge base. When the rule was matched and ready to fire, the inference engine created an agenda item. When the agenda item fired, the consequent of the rule

entered new assertions into the knowledge base. These assertions in turn could cause more agenda items to be created, continuing the forward chaining.

As mentioned above, GoldWorks allowed the control of the inference to be altered by using priorities. The higher the priority, the sooner the rule was applied to the agenda list. TTFIM used priorities and the SHIFT relation (appendix J, page 76) to control the ordering of agenda items.

**Operation Protocol**

During flight, the TTFIM KBS software operated in interpreted-LISP mode on a Gold Hill Computers' HummingBoard card, installed in an 80286-based Personal Computer (PC) clone. The HummingBoard was an 80386-based CPU that could be housed in any 8088-based PC, 80286-based PC, or compatible. TTFIM's process communicated with the display processes on the Norden computer, by sending display symbol control words, via the common data transfer system called the Digital Autonomous Terminal Access Communication (DATAC) bus.

All of the various functions of the TSRV were networked through the DATAC bus. The DATAC bus was also the means for retrieving the massive amount of input data necessary to TTFIM. The DATAC bus was a 1 megahertz serial bus which operated in broadcast mode - every terminal on the bus had access to the transmissions of all other terminals on the bus. All aircraft parameters from sensor interface pallets, and all flight commands, were distributed over the DATAC bus for access from any of the various stations depicted in Figure B-2 (page 36).

GoldWorks' built-in low-level functions permitted TTFIM to communicate with the DATAC bus by accessing certain areas of the host PC's memory. When TTFIM read from or wrote to the special area of the PC memory, a PC resident program did the appropriate transfer (i.e., read or write) on the DATAC bus.

All of TTFIM's data were formatted into the low byte of the specified DATAC address. The upper bytes of the DATAC words were not used since the TTFIM hardware was not able to address the upper byte. TTFIM outputs consisted of four packed discrete bytes. The bytes were put on the DATAC bus as the low bytes of four DATAC words. The first two words indicated the state of the 14 configurations of optional PFD symbols controlled by TTFIM. The third word indicated the detected phase of flight. And, the fourth word was a one bit error flag.

TABLE F-1
TTFIM Output Control Words

| Display Control Word | Bit | Indication |
|---|---|---|
| 0 | 0 | Reference Altitude |
| | 1 | Waypoint Star |
| | 2 | Horizontal Deviation |
| | 3 | Glideslope Deviation |
| | 4 | Localizer Deviation |
| | 5 | CAS Reference (dial) |
| | 6 | CAS Reference (buffer) |
| 1 | 0 | Runway Image |
| | 1 | Radar Altitude |
| | 2 | Vertical Path |
| | 3 | Flare-Guide |
| | 4 | Track-Angle Error 1 |
| | 5 | Track-Angle Error 2 |
| | 6 | Track-Angle Error 3 |
| 2 | 0 | Takeoff |
| | 1 | Terminal Climb |
| | 2 | Cruise |
| | 3 | Terminal Descent |
| | 4 | Land |
| | 5 | Taxi |
| | 6 | Enroute Climb |
| | 7 | Enroute Descend |
| 3 | 0 | Error Flag (for flight phase) |

The formatting and final control of all displays in the TSRV were the responsibilities of the Displays computer (shown in Figure B-2, page 36). The Displays computer produced the high-level commands needed by the display systems in the aft cockpit (i.e., graphics commands). The logical decision of which optional information display elements to present on the PFD was formerly embedded in these high-level display commands. However, with the KBS in operation, the logic portion of the Displays computer software was disabled. The Display computer looked for the display control words from the KBS to determine which optional information display elements to present.

**Baseline Software Modifications**

In addition to the KBS development, this effort required modification of the displays input/output (I/O) handler as well as the I/O data common. I/O routines were written to format the input discretes used by the GoldWorks software, and to unpack the output discretes which controlled the display of optional display symbols on the PFD (see Table F-1) and dictate which phase of flight to indicate in the top left corner of the PFD. Display modules were modified to accept the output discretes from the GoldWorks software. The modules were modified in such a way

that control of the optional display symbols could be determined by either the present logic or the KBS software. Control was switched via a configuration word set interactively by the experimenter.

Implementing the TTFIM software also included the redefinition of several bezels on the PFD. The eight bezels on the left of the PFD were used for manual selection of phase of flight (when not in automatic detection mode) and the seventh bezel on the right was used to select manual or automatic phase of flight configuration.

# Appendix G
# Flight-Phase Detection Rules

The following logic represents that used in the TTFIM flight tests for determining the phase of flight. As with the information management logic, the flight-phase detection logic appears as it did in the flight tests, even though simplifications can now be made.

The flight-phase detection rules were "transition-in" rules, meaning that the conditions required for a flight-phase to be active only needed to be true to start that phase, not to stay in that phase. Once in a specified phase of flight, the only way the system would transition into another phase of flight was if the conditions for another phase became true (not if the conditions of the current were not true). For example, one of the conditions in the rule for takeoff stated that the flight phase in the previous cycle was either taxi or land (i.e., Last-Phase = TX or LD). One cycle after the phase of flight became takeoff, the "last- phase" variable was set to takeoff, thus not satisfying the condition stating that the last-phase must be either taxi or land. However, the phase of flight remained takeoff until the conditions of another phase became true.

TAXI (TX)
- Last-Phase = T/O or LD
- Squat-Switch = GROUND
- Gear = DOWN
- EPR < 1.8
- -1.0° < GAMMA < 1.0°
- Radar-Altitude < = 10'
- FLAPS < 5°

TAKEOFF (T/O)
- Last-Phase = TX or LD
- Reversers = OFF
- EPR > 1.8
- Radar-Altitude < = 400'
- Flaps < = 30°

TERMINAL-CLIMB (TC)
- Last-Phase = T/O or TD
- Squat-Switch = AIR
- Gear = UP
- Gamma > = 1.0°
- Radar-Altitude > = 400'
- Barometric-Altitude < 5000'
- Flaps < = 15°

ENROUTE-CLIMB (EC)
- Last-Phase = TC or CR or ED
- Squat-Switch = AIR
- Gear = UP
- EPR > 1.2
- Gamma > = 1.0°
- Barometric-Altitude > = 10000

CRUISE (CR)
Last-Phase = EC or ED
Squat-Switch = AIR
Gear = UP
EPR > 1.2
-1.0° < Gamma < 1.0°
Flaps = 0°
Barometric-Altitude > = 10000

ENROUTE-DESCENT (ED)
Last-Phase = CR or ED
Squat-Switch = AIR
Gear = UP
EPR < 1.4
Gamma < -1.0°
Barometric-Altitude > = 10000

TERMINAL-DESCENT (TD)
Last-Phase = TC or ED
Squat-Switch = AIR
EPR < 1.4
Gamma < -1.0°
Barometric-Altitude < 10000

LAND (LD)
Last-Phase = T/O or TD or TC
Gear = DOWN
EPR < 1.8
Gamma < = 0°
Flaps > = 15°

# Appendix H
# Stage 1 Flight-Test Envelopes

The stage 1 flight tests of TTFIM were done during the baseline verification flights on June 7 and 13, 1989. The functionality of TTFIM at this stage, was to duplicate the functionality of the traditional implementation in the baseline, therefore transparent to the pilot. So, dedicated flight tests were not necessary. The flight-test envelope for the June 7 and 13, 1989, are given in the tables below.

TABLE H-1
June 7, 1989, TSRV Flight-Test Envelope

| Flt.Deck | Configuration | Purpose | Procedure |
|---|---|---|---|
| RFD | Cruise-Trimmed<br>Sidearm control | VCWS mode check<br>Display system check | Engage RFD<br>Select VCWS<br><br>Check pitch roll & yaw |
| FFD | STAR WFB13 (see Figure H-1)<br>Altitude 4000'<br><br>210 kts<br><br>ADIRS<br><br>Autothrottle | MLS autoland evaluation<br><br>New throttle | Engage auto HOR & VERT path<br>Enter star at first waypoint<br>Enable MLS when MLS is valid<br>Pilot call out alt & cross track errors at MLS engage<br>Arm land mode in final turn<br>Continue landing through rollout |
| RFD | STAR WFB13 (see Figure H-1)<br>Altitude 4000'<br><br>210 kts<br><br>SAC config 1.0<br><br>ADIRS | Manual MLS<br><br>Approach & landing<br><br>Pilot evaluation of SAC configuration | Engage VCWS & autothrottles<br>Enter star at first waypoint<br>Enable MLS when valid<br>Pilot call out alt & crosstrack errors at MLS engage<br>Continue approach to go-around |
| RFD | SAC config 2.0 | Pilot evaluation of SAC alternate configuration | Repeat previous procedure<br>Exercise Lateral Trim Switch |

TABLE H-1 (continued)

| Flt.Deck | Configuration | Purpose | Procedure |
|---|---|---|---|
| RFD | SAC config 3.0 | Same as previous | Repeat previous procedure |
| RFD | Gear up<br>Flaps up<br>CAS = 210 kts<br>Altitude 10,000' | Stick shaker check<br>AOA vane check | Engage RFD<br>Select altitude hold<br>Set idle thrust - FFD safety pilot should check that throttles are on AFT stop<br>FFD Pilot should call out $CL/CL_{max}$ values in .1 inc<br>RFD pilot initiates recovery |
| RFD | Gear up<br>Flaps 1<br>CAS = 200 kts | Same as previous | Repeat previous procedure |
| RFD | Gear up<br>Flaps 5<br>CAS = 195 kts | Same as previous | Repeat previous procedure |
| RFD | Gear up<br>Flaps 10<br>CAS = 180 kts | Same as previous | Repeat previous procedure |
| RFD | Gear up<br>Flaps 15<br>CAS = 165 kts | Same as previous | Repeat previous procedure |
| RFD | Gear up<br>Flaps 25<br>CAS = 160 kts | Same as previous | Repeat previous procedure |

TABLE H-1 (continued)

| Flt.Deck | Configuration | Purpose | Procedure |
|---|---|---|---|
| RFD | Gear down<br>Flaps 25<br>CAS = 160 kts | Same as previous | Repeat previous procedure |
| RFD | Gear down<br>Flaps 30<br>CAS = 155 kts | Same as previous | Repeat previous procedure |
| RFD | Gear down<br>Flaps 40<br>CAS = 140 kts | Same as previous | Repeat previous procedure |
| RFD | Cruise-trimmed<br>Altitude 10,000'<br>CAS = 250 kts<br>SAC Config X.1p* | VCWS performance Longitudinal axis<br>Display evaluation | Engage RFD VCWS<br>Apply 1/4 PMC step input up allowing 2 deg increase in att<br>Release & allow to stabilize<br>Return to level<br>Displays verify: flight path angle command, symbol & position, pitch attitude, VCWS indication, drift angle indication |
| RFD | same as previous | same as previous | Repeat previous procedure for 1/4 PMC down & 2 deg attitude decrease |

* - either 1, 2, or 3, depending on results of previous runs

TABLE H-2
June 13, 1989, TSRV Flight-Test Envelope

| Flt.Deck | Configuration | Purpose | Procedure |
|---|---|---|---|
| RFD | Cruise-trimmed<br>Altitude 10,000'<br>CAS = 250 kts<br>SAC Config 4 | VCWS performance<br>Longitudinal axis<br>Display evaluation | Engage RFD VCWS<br>Apply 1/4 PMC step input up allowing 2 deg increase in att<br>Release & allow to stabilize<br>Return to level<br>Displays verify: flight path angle command, symbol & position, pitch attitude, VCWS indication, drift angle indication |
| RFD | same as previous with CAS = 300 kts | same as previous | same as previous |
| RFD | same as previous with CAS = 250 kts | same as previous | same as previous with 1/4 PMC down & 2 deg attitude decrease |
| RFD | same as previous | same as previous | same as previous with 1/4 PMC up & 5 deg attitude increase |
| RFD | same as previous with CAS = 300 kts | same as previous | same as previous |
| RFD | same as previous | same as previous | same as previous |
| RFD | same as previous | same as previous | same as previous |
| RFD | same as previous | same as previous | same as previous |
| RFD | Cruise-Trimmed<br>Altitude 10,000'<br>CAS = 250 kts | VCWS performance<br>Longitudinal axis<br>Precise control<br>Display symbology checks | RFD engage VCWS<br>Increase flight path angle in 1 deg steps to 5 deg, stabilizing at each step<br>Return to level |

TABLE H-2 (continued)

| Flt.Deck | Configuration | Purpose | Procedure |
|---|---|---|---|
| RFD | same as previous | same as previous | same as previous with decrease in 1 deg steps to -5 deg, stabilizing at each |
| RFD | same as previous | same as previous | same as previous with increases and using manual click trim switch |
| RFD | Cruise-Trimmed<br><br>Altitude > 10,000'<br><br>SAC config 4<br><br>Manual Throttle fixed<br>CAS = 250 kts | Manual electric pitch stability check | Disconnect RFD<br>Trim a/c<br>Reengage RFD<br>Apply 1/4 PMC step input up allowing 2 deg increase in alt<br>Release, allow to stabilize<br>Return to level<br>Display verify:<br>flight path angle command, symbol & position, pitch attitude, VCWS indication, drift angle indication |
| RFD | same as previous with CAS = 300 kts | same as previous | same as previous |

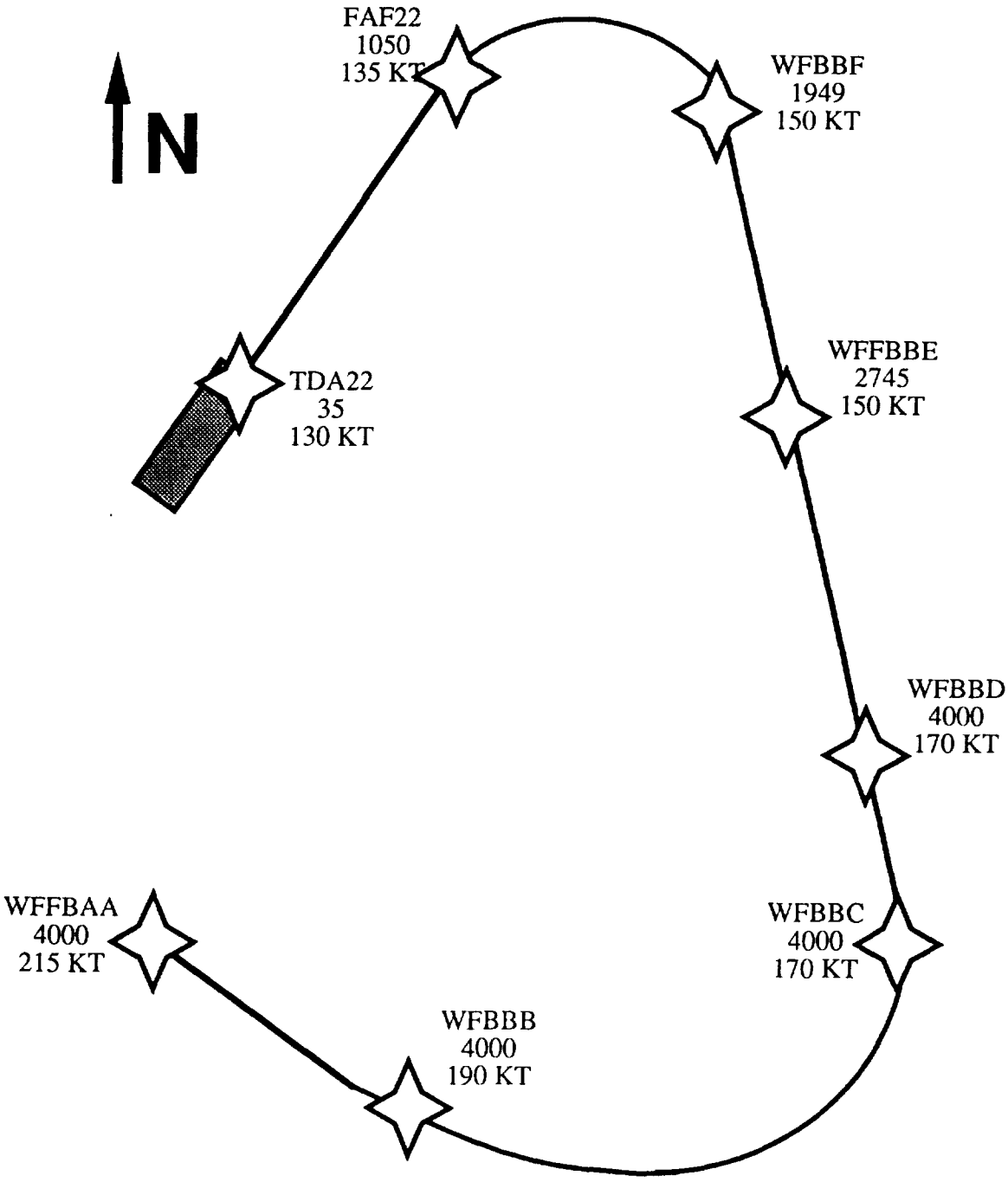



Figure H-1 - STAR WFB13

# Appendix I
# Stage 2 Flight-Test Envelope

One objective of the November 13, 1989 flight test was to functionally test and check out the operation of the flight-phase detection KBS and the integration of the KBS with the information selection KBS and other onboard systems (primarily to check out the automatic flight-phase detection). The flight-test envelope for this flight test is given in the Table below.

TABLE I-1
November 13, 1989, Portion of the TSRV Flight-Test Envelope

| Flt.Deck | Configuration | Purpose | Procedure |
|---|---|---|---|
| RFD | Altitude < 10,000' | TTFIM automatic flight-phase detection check in terminal area | TX<br>T/O<br>TC - then level off<br>TD - then level off |
| RFD | same as previous | same as previous | TC - no level off<br>TD - no level off |
| RFD | Altitude > 10,000' | TTFIM automatic flight-phase detection check outside terminal area | TC<br>EC<br>CR - level off, vary gamma (-1 and 1)<br>ED<br>CR<br>EC<br>CR<br>EC - no level off, straight to next run |
| RFD | Touch and go<br>STAR WFB13<br>(see Figure H-1) | same as previous for touch and go | ED<br>TD<br>LD - touch and go<br>T/O |
| RFD | Full mission phase detection | same as previous for full mission | TC<br>EC<br>CR<br>ED<br>TD<br>LD<br>TX |

# Appendix J
# GoldWorks Code

The following are the GoldWorks files used in the flight tests. Minor changes were made to make the code more readable (e.g., documentation). Note that "Track-Angle Error" is consistently referred to as Crosstrack Deviation in the following code. No effort was made to correct this conflict in the following code. The "Crosstrack Deviation" output from this code was interpreted by the Display computer as meaning "Track-Angle Error."

```
;;
;; defframe.lsp -- dolm: 1-12-90
;;
;; Defines all frames. Each of the frames will be described in detail using in-line
;; documentation.
;;

(in-package 'gw)

;; CURRENT is a top-frame used to categorize the instances NOT-IN-PHASE, and
;; IN-PHASE. The fields defined here that will be inherited by all instances of
;; CURRENT are: NOW-IS -- to specify the what the current phase is or is not and,
;; ERR-FLG -- to be set when an error in the logic occurs.

(DEFINE-FRAME CURRENT
  (:print-name "CURRENT"
   :doc-string ""
   :is TOP-FRAME)
  (NOW-IS
     :default-values  (TAXI)
     :constraints (:ONE-OF (TAXI TAKEOFF TERM-CB TERM-DS ENR-CB
                    ENR-DS CRUISE LAND)))
  (ERR-FLG
     :default-values  (0)
     :constraints (:ONE-OF (0 1))))

;; DSPLAY is a top-frame used to categorize the instance of SYMBOL (just one
;; instance at the present time). The fields defined here that will be inherited by
;; all instances of DSPLAY are each symbols on the PFD that will be either on or
;; off.

(DEFINE-FRAME DSPLAY
  (:print-name "DSPLAY"
   :doc-string ""
   :is TOP-FRAME)
  (REF-ALT
        :default-values  (OFF)
     :constraints (:ONE-OF (OFF ON)))
  (G/S-DEV
     :default-values  (OFF)
     :constraints (:ONE-OF (OFF ON)))
  (XTK-DEV
```

```
    :default-values (OFF)
    :constraints (:ONE-OF (OFF ON)))
  (XTK-DEV2
    :default-values (OFF)
    :constraints (:ONE-OF (OFF ON)))
  (XTK-DEV4
    :default-values (OFF)
    :constraints (:ONE-OF (OFF ON)))
  (WP-STAR
    :default-values (OFF)
    :constraints (:ONE-OF (OFF ON)))
  (LOC-DEV
    :default-values (OFF)
    :constraints (:ONE-OF (OFF ON)))
  (RWY-IMAGE
    :default-values (OFF)
    :constraints (:ONE-OF (OFF ON)))
  (RAD-ALT
    :default-values (OFF)
    :constraints (:ONE-OF (OFF ON)))
  (HOR-DEV
    :default-values (OFF)
    :constraints (:ONE-OF (OFF ON)))
  (FLARE-GUIDE
    :default-values (OFF)
    :constraints (:ONE-OF (OFF ON)))
  (CAS-REF-DIAL
    :default-values (OFF)
    :constraints (:ONE-OF (OFF ON)))
  (CAS-REF-BUF
    :default-values (OFF)
    :constraints (:ONE-OF (OFF ON)))
  (VERT-PATH
    :default-values (OFF)
    :constraints (:ONE-OF (OFF ON))))

;; CONTROL is a top-frame used to categorize the instances of SHOW and INIT.
;; The fields defined here that will be inherited by all instances of CONTROL are:
;; DLOAD -- to specify whether or not to download from the DATAC bus;
ULOAD --
;; to specify whether or not to upload to the DATAC bus; and, STATUS -- to
indicate
;; that an update to the display control words had occurred.

(DEFINE-FRAME CONTROL
  (:print-name "CONTROL"
   :doc-string ""
   :is TOP-FRAME)
  (DLOAD
    :default-values (NO)
    :constraints (:ONE-OF (NO YES)))
  (ULOAD
    :default-values (NO)
    :constraints (:ONE-OF (NO YES)))
```

```
  (STATUS
    :default-values  (0)))

;; A/C-STATUS is a top-frame used only for categorization. There are no
;; inheritance fields. The next level in this lattice is still a frame. All instances will
;; be of the sub-frames grouped under this top-frame. The sub-frames are BEZEL,
;; CMS, VALIDS, BOOLS, and ANALOG.

(DEFINE-FRAME A/C-STATUS
  (:print-name "A/C-STATUS"
   :doc-string ""
   :is TOP-FRAME))

;; BEZEL is a sub-frame of A/C-STATUS. The instance of BEZEL are used to
;; assert which selection switches have been set. There are 10 fields defined in
;; BEZEL and inherited by SWITCHED -- each can be either ON or OFF.

(DEFINE-FRAME BEZEL
  (:print-name "BEZEL"
   :doc-string ""
   :is A/C-STATUS)
  (RALT
    :default-values  (OFF)
    :constraints (:ONE-OF (OFF ON 0 1)))
  (VRT
    :default-values  (OFF)
    :constraints (:ONE-OF (OFF ON 0 1)))
  (G/S
    :default-values  (OFF)
    :constraints (:ONE-OF (OFF ON 0 1)))
  (XTK
    :default-values  (OFF)
    :constraints (:ONE-OF (OFF ON 0 1)))
  (STAR
    :default-values  (OFF)
    :constraints (:ONE-OF (OFF ON 0 1)))
  (HOR
    :default-values  (OFF)
    :constraints (:ONE-OF (OFF ON 0 1)))
  (RWY
    :default-values  (OFF)
    :constraints (:ONE-OF (OFF ON 0 1)))
  (CAS
    :default-values  (OFF)
    :constraints (:ONE-OF (OFF ON 0 1)))
  (LOC
    :default-values  (OFF)
    :constraints (:ONE-OF (OFF ON 0 1)))
  (AUTO-PHASE
    :default-values  (OFF)
    :constraints (:ONE-OF (OFF ON 0 1))))

;; CMS is a sub-frame of A/C-STATUS. The instance of CMS is MODE-IS which
;; is used to indicate which automatic control mode the airplane is configured for (if
```

```
;; any). There are 11 fields defined in CMS mapping to each of the switches on the
;; mode control panel. While the selections are more limited for some than others,
;; each of the fields will range from 1 to 4 (the most).

(DEFINE-FRAME CMS
  (:print-name "CMS"
   :doc-string ""
   :is A/C-STATUS)
  (CAS
     :default-values  (1)
     :constraints (:RANGE (1 4)))
  (ALT
     :default-values  (1)
     :constraints (:RANGE (1 4)))
  (TKA
     :default-values  (1)
     :constraints (:RANGE (1 4)))
  (V-CWS
     :default-values  (1)
     :constraints (:RANGE (1 4)))
  (A-CWS
     :default-values  (1)
     :constraints (:RANGE (1 4)))
  (AUTO
     :default-values  (1)
     :constraints (:RANGE (1 4)))
  (H-PATH
     :default-values  (1)
     :constraints (:RANGE (1 4)))
  (V-PATH
     :default-values  (1)
     :constraints (:RANGE (1 4)))
  (T-PATH
     :default-values  (1)
     :constraints (:RANGE (1 4)))
  (LAND
     :default-values  (1)
     :constraints (:RANGE (1 4)))
  (FPA
     :default-values  (1)
     :constraints (:RANGE (1 4))))

;; VALIDS is a sub-frame of A/C-STATUS. The instance of VALIDS is VALID-IS
;; which is used to assert which signals are currently valid in the airplane's
;; computer. Each of the inheritance fields can be either ON or OFF (i.e., valid or
;; not valid).

(DEFINE-FRAME VALIDS
  (:print-name "VALIDS"
   :doc-string ""
   :is A/C-STATUS)
  (G/S
     :print-name ""
     :default-values  (OFF)
```

```
    :constraints (:ONE-OF (OFF ON)))
  (LOC
    :default-values  (OFF)
    :constraints (:ONE-OF (OFF ON)))
  (NAV-PATH
    :default-values  (OFF)
    :constraints (:ONE-OF (OFF ON)))
  (NAVPATH2
    :default-values  (OFF)
    :constraints (:ONE-OF (OFF ON))))

;; BOOLS is a sub-frame of A/C-STATUS. The instance of BOOLS is
;; DISCRETE-IS. The inheritance fields of BOOLS are used for asserting boolean
;; information pertaining to the aircraft's current configuration

(DEFINE-FRAME BOOLS
  (:print-name "BOOLS"
   :doc-string ""
   :is A/C-STATUS)
  (LAST-WP
    :default-values  (OFF)
    :constraints (:ONE-OF (OFF ON)))
  (RWY-IN-NAV
    :default-values  (OFF)
    :constraints (:ONE-OF (OFF ON)))
  (IN-COVERAGE
    :default-values  (OFF)
    :constraints (:ONE-OF (OFF ON)))
  (WP-ALERT
    :default-values  (OFF)
    :constraints (:ONE-OF (OFF ON)))
  (WP-DISPLAYABLE
    :default-values  (OFF)
    :constraints (:ONE-OF (OFF ON)))
  (GEAR
    :default-values  (OFF)
    :constraints (:ONE-OF (OFF ON)))
  (SQUAT
    :default-values  (OFF)
    :constraints (:ONE-OF (OFF ON)))
  (TREVERSE
    :default-values  (OFF)
    :constraints (:ONE-OF (OFF ON))))

;; ANALOG is a sub-frame of A/C-STATUS. The instance of ANALOG is IS
;; which is used to assert information about the aircraft that takes on analog values
;; (e.g., like altitude). Likewise, the inheritance fields of the ANALOG refer to
;; analog values needed by TTFIM.

(DEFINE-FRAME ANALOG
  (:print-name "ANALOG"
   :doc-string ""
   :is A/C-STATUS)
  (ALTITUDE
```

```
    :default-values  (0)
    :constraints (:RANGE (-100 25000)))
  (DEC-HEIGHT
    :default-values  (1000)
    :constraints (:RANGE (0 2500)))
  (FLAPS)
  (EPR)
  (GAMMA)
  (RADAR-ALT)
  (PHASE-IN
    :default-values  (0)
    :constraints (:RANGE (0 128)))
  (PHASE-OUT
    :default-values  (0)
    :constraints (:RANGE (0 128)))
  (RWY-HEADING
    :default-values  (0)
    :constraints (:RANGE (-180 180)))
  (A/C-TRACK
    :default-values  (0)
    :constraints (:RANGE (-180 180))))

;;
;; definstn.lsp -- dolm: 1-16-90
;;
;; Defines (and makes) all instances.  Each of the instances will be described in
;; detail using in-line documentation.
;;

;; NOT-IN-PHASE inherits CURRENT fields with nothing extra.

(DEFINE-INSTANCE NOT-IN-PHASE
 (:print-name "NOT-IN-PHASE"
  :doc-string ""
  :is  CURRENT)
 (NOW-IS TAXI)
 (ERR-FLG 0)
 )

;; IN-PHASE inherits CURRENT fields with 2 additions.  When the phase of flight
;; is modified, the when-modified daemon calls NEW-SHOW to set the system flag
;; noting that a new phase needs to be displayed.  If an error occurs in the automatic
;; flight-phase detection logic, the when-modified daemon calls ZERO-SET to
;; change the value back to 0.

(MAKE-INSTANCE 'IN-PHASE
 :print-name "IN-PHASE"
 :doc-string ""
 :is  'CURRENT
 :slots
  '(
 (NOW-IS :VALUE TAXI :WHEN-MODIFIED (NEW-SHOW))
 (ERR-FLG :VALUE 0 :WHEN-MODIFIED (ZERO-SET))
 ))
```

```
;; DISCRETE-IS inherits the slots of frame BOOLS with all slots initialized to
;; OFF.

(DEFINE-INSTANCE DISCRETE-IS
 (:print-name "DISCRETE-IS"
  :doc-string ""
  :is  BOOLS)
 (LAST-WP OFF)
 (RWY-IN-NAV OFF)
 (IN-COVERAGE OFF)
 (WP-ALERT OFF)
 (WP-DISPLAYABLE OFF)
 (GEAR OFF)
 (SQUAT OFF)
 (TREVERSE OFF)
 )

;; SHOW inherits the fields of CONTROL. The fields are used to control
;; execution characteristics of TTFIM. DLOAD is used to control the downloading
;; from the DATAC card. ULOAD controls the uploading to the DATAC card.
;; And, STATUS is used to indicate a change has occurred.

(DEFINE-INSTANCE SHOW
 (:print-name "SHOW"
  :doc-string ""
  :is  CONTROL)
 (DLOAD NO)
 (ULOAD NO)
 (STATUS 0)
 )

;; INIT inherits the fields of CONTROL with 2 additions. DLOAD is to control the
;; downloading from the DATAC card to the TTFIM software. When DLOAD is
;; modified, the daemon calls FULLOAD to signify a full loading of all DATAC
;; values. ULOAD works as the inverse of DLOAD with an inverse daemon
;; function SEND-DSP-CMD.

(MAKE-INSTANCE 'INIT
 :print-name "INIT"
 :doc-string ""
 :is  'CONTROL
 :slots
  '(
 (DLOAD :VALUE NO :WHEN-MODIFIED (FULLOAD))
 (ULOAD :VALUE NO :WHEN-MODIFIED (SEND-DSP-CMD))
 (STATUS :VALUE 0)
 ))

;; SYMBOL is an instance of DSPLAY. Each of the display symbols are initialized
;; to OFF with when-modified daemons calling function NEW-SHOW to notify the
;; system that a change has been made, and function OFF-SET to change it back to
;; OFF. Note, how the rule are stated will only turn a symbol ON, not OFF.
;; Therefore, they are cut off each time (in this system only) and turned on again if
```

```
;; they are still active on the next cycle.

(MAKE-INSTANCE 'SYMBOL
 :print-name "SYMBOL"
 :doc-string ""
 :is 'DSPLAY
 :slots
  '(
(REF-ALT :VALUE OFF :WHEN-MODIFIED (NEW-SHOW OFF-SET))
(G/S-DEV :VALUE OFF :WHEN-MODIFIED (NEW-SHOW OFF-SET))
(XTK-DEV :VALUE OFF :WHEN-MODIFIED (NEW-SHOW OFF-SET))
(XTK-DEV2 :VALUE OFF :WHEN-MODIFIED (NEW-SHOW OFF-SET))
(XTK-DEV4 :VALUE OFF :WHEN-MODIFIED (NEW-SHOW OFF-SET))
(WP-STAR :VALUE OFF :WHEN-MODIFIED (NEW-SHOW OFF-SET))
(LOC-DEV :VALUE OFF :WHEN-MODIFIED (NEW-SHOW OFF-SET))
(RWY-IMAGE :VALUE OFF :WHEN-MODIFIED (NEW-SHOW OFF-SET))
(RAD-ALT :VALUE OFF :WHEN-MODIFIED (NEW-SHOW OFF-SET))
(HOR-DEV :VALUE OFF :WHEN-MODIFIED (NEW-SHOW OFF-SET))
(FLARE-GUIDE :VALUE OFF :WHEN-MODIFIED (NEW-SHOW OFF-SET))
(CAS-REF-DIAL :VALUE OFF :WHEN-MODIFIED (NEW-SHOW OFF-SET))
(CAS-REF-BUF :VALUE OFF :WHEN-MODIFIED (NEW-SHOW OFF-SET))
(VERT-PATH :VALUE OFF :WHEN-MODIFIED (NEW-SHOW OFF-SET))
 ))

;; IS is an instance of ANALOG and is used to initialize the value of ALTITUDE to
;; 0, DEC-HEIGHT to 1000, PHASE-IN to 0, PHASE-OUT to 0,
;; RWY-HEADING to 0, and A/C-TRACK to 0. Note that throughout this system,
;; all variables (slots or assertions) that refer to altitude take on a half value due to
;; restrictions of the systems integer values.

(DEFINE-INSTANCE IS
 (:print-name "IS"
  :doc-string ""
  :is ANALOG)
(ALTITUDE 0)
(DEC-HEIGHT 1000)
(PHASE-IN 0)
(PHASE-OUT 0)
(RWY-HEADING 0)
(A/C-TRACK 0)
 )

;; MODE-IS is an instance of CMS in which each of the control mode switches are
;; initialized to 1.

(DEFINE-INSTANCE MODE-IS
 (:print-name "MODE-IS"
  :doc-string ""
  :is CMS)
(CAS 1)
(ALT 1)
(TKA 1)
(V-CWS 1)
(A-CWS 1)
```

```
(AUTO 1)
(H-PATH 1)
(V-PATH 1)
(T-PATH 1)
(LAND 1)
(FPA 1)
)

;; SWITCHED is an instance of BEZEL in which each of the bezel switches
;; associated to display options are initialized to OFF. Note, the last bezel switch
;; mentioned, AUTO-PHASE, is the switch the flight engineer used to toggle
;; between automatic flight-phase detection, and manual flight phase entry.

(DEFINE-INSTANCE SWITCHED
 (:print-name "SWITCHED"
  :doc-string ""
  :is BEZEL)
(RALT OFF)
(VRT OFF)
(G/S OFF)
(XTK OFF)
(STAR OFF)
(HOR OFF)
(RWY OFF)
(CAS OFF)
(LOC OFF)
(AUTO-PHASE OFF)
)

;; VALID-IS is an instance of VALIDS in which it initializes each slot to OFF (or
;; not valid).

(DEFINE-INSTANCE VALID-IS
 (:print-name "VALID-IS"
  :doc-string ""
  :is VALIDS)
(G/S OFF)
(LOC OFF)
(NAV-PATH OFF)
(NAVPATH2 OFF)
)


;;
;; defreltn.lsp -- dolm: 1-16-90
;;
;; Defines all relations. Each relation will be described in more detail using in-line
;; documentation
;;

(in-package 'gw)

;; EQUIV is used to define table-lookups. For example
;;                (EQUIV takeoff 2)
```

```
;; can be used like (EQUIV ?p 2) to bind ?p to takeoff, or like (EQUIV takeoff ?p)
;; to bind ?p to the numerical value of takeoff. This becomes more powerful when
;; all phases are considered and the EQUIV relation is used in rules where the
;; phase is known in either the numerical or symbolic form and not both.

(DEFINE-RELATION EQUIV
  (:print-name "EQUIV"
   :doc-string ""
   :explanation-string ""
   :LISP-function NIL
   :relation-type :ASSERTION)
  NIL)

;; SHIFT is used to help dictate the flow of TTFIM. SHIFT adds some
;; procedurality to TTFIM. SHIFT probably could have been left out all together
;; and its role handled by priorities. However, SHIFT makes it easier to
;; understand. Certain rules will change the value of SHIFT to direct the inference
;; to other sections of rules. Shift is a functional-assertion and therefore only one
;; assertion of SHIFT can be true at one time.

(DEFINE-RELATION SHIFT
  (:print-name "SHIFT"
   :doc-string ""
   :explanation-string ""
   :LISP-function NIL
   :relation-type :FUNCTIONAL-ASSERTION)
  NIL)

;; DETECT is used to dictate whether the automatic detection of flight phases is on
;; or not. DETECT is a functional assertion.

(DEFINE-RELATION DETECT
  (:print-name "DETECT"
   :doc-string ""
   :explanation-string ""
   :LISP-function NIL
   :relation-type :FUNCTIONAL-ASSERTION)
  NIL)

;; PHASE is used in the flight-phase detection KBS to store the valid flight phases.
;; It's not functional because the rules allow more than one phase to be detected
;; (however, more than one is never detected).

(DEFINE-RELATION PHASE
  (:print-name "PHASE"
   :doc-string ""
   :explanation-string ""
   :LISP-function NIL
   :relation-type :ASSERTION)
  NIL)

;;
;; defassrt.lsp -- dolm: 1-16-90
;;
```

```
;; Defines initial assertions. Descriptions of theses assertions will be described
;; using in-line documentation.
;;

(in-package 'gw)

;; Some of these assertions can be made by changing intial values of slots in the
;; appropriate instances. However, it's easier to keep track of them here. The table
;; lookup for the flight phase relations are also defined here.

(DEFINE-ASSERTION
 (and
   (NOW-IS IN-PHASE TAXI)
   (SHIFT TOP)
   (MONITOR DISPLAY ON)
   (EQUIV TAKEOFF 1)
   (EQUIV TERM-CB 2)
   (EQUIV CRUISE 4)
   (EQUIV TERM-DS 8)
   (EQUIV LAND 16)
   (EQUIV TAXI 32)
   (EQUIV ENR-CB 64)
   (EQUIV ENR-DS 128)))

;;
;; r-top.lsp  --  dolm:  1-17-90
;;
;; This group of rules are to fire first. The order within the group is dictated by the
;; priority value. The fact that this group fires first is dictated by the (SHIFT TOP)
;; which is set during the loading of the system, and after the output of the results.
;; The appropriate rules within this group will fire, then make the assertion (SHIFT
;; DETECT) so that the phase can be detected (either manually or automatically.
;; Each rule's purpose will be described using in-line documentation.
;;

(in-package 'gw)

;; DOWNLOAD sets the DLOAD slot of INIT (an instance of CONTROL) to YES
;; so that the values from the DATAC bus can be loaded into the system.

(DEFINE-RULE DOWNLOAD
  (:print-name "DOWNLOAD"
   :doc-string ""
   :dependency NIL
   :direction :FORWARD
   :certainty 1.0
   :explanation-string ""
   :priority 900
   :sponsor TOP-SPONSOR)
  (SHIFT TOP)
 THEN
  (INSTANCE INIT IS CONTROL WITH DLOAD YES)  )
```

```
;; QUERY-AUTO determines if the user (i.e., pilot) wants to have the system
;; automatically detect the phase of flight. If the user wants the automatic
;; detection, the auto-phase switch will be on. Therefore, if the system is recycling
;; to the top (i.e., SHIFT TOP), and the auto-phase switch is on, then assert
;; (DETECT AUTO) so that the rules for automatic detection will fire, and turn
;; control over to the detection section (i.e., SHIFT DETECT).

(DEFINE-RULE QUERY-AUTO
  (:print-name "QUERY AUTO"
   :doc-string ""
   :dependency NIL
   :direction :FORWARD
   :certainty 1.0
   :explanation-string ""
   :priority 800
   :sponsor TOP-SPONSOR)
  (SHIFT TOP) (AUTO-PHASE SWITCHED ON)
 THEN
  (DETECT AUTO) (SHIFT DETECT) )

;; QUERY-MANUAL determines if the user (i.e., pilot) wants to manually indicate
;; the phase of flight. If the user wants manual entry, the auto-phase switch will be
;; off. Therefore, if the system is recycling to the top (i.e., SHIFT TOP), and the
;; auto-phase switch is off, then assert (DETECT MANUAL) so that the rules for
;; automatic detection will not fire, and turn control over to the detection section
;; (i.e., SHIFT DETECT).

(DEFINE-RULE QUERY-MANUAL
  (:print-name "QUERY MANUAL"
   :doc-string ""
   :dependency NIL
   :direction :FORWARD
   :certainty 1.0
   :explanation-string ""
   :priority 800
   :sponsor TOP-SPONSOR)
  (SHIFT TOP) (AUTO-PHASE SWITCHED OFF)
 THEN
  (DETECT MANUAL) (SHIFT DETECT) )


;;
;; r-phases.lsp  --  dolm: 1-19-90
;;
;; This group of rules fire after TOP, and before the symbol rules. There is no
;; reason to dictate the transition to the symbol rules with a shift command since the
;; priorities can handle that. The ultimate purpose of these rules is to set the
;; assertion (NOW-IS IN-PHASE ?). The first rule handles the case when manual
;; input of flight phase has been chosen. There is then a series of rules that handle
;; the automatic detection of all valid phases of flights. The last set of rules work
;; with the automatic phase selection by picking one phase out of multiple, none, or
;; one choice given by the previous rules. Transition to the last set of rules is done
;; with (SHIFT READY). After the phase of flight has been chosen, (SHIFT
;; OUTPUT) is set to allow the output section of code to fire. However, the next
```

```
;; set of rules that will fire are the symbol rules since their priority is higher.
;;

;; SLAVE-PHASE is used to determine NOW-IS IN-PHASE when manual
;; selection has been selected. While in manual selection, PHASE-IN IS is
;; asserted with the numerical equivalence of the flight phase as dictated by the
;; bezel switches beside the PFD. Then using the EQUIV assertions as a table
;; lookup, NOW-IS IN-PHASE is bound to the symbolic equivalent of the
;; numerical representation of the flight phase. SHIFT is then changed to
;; OUTPUT for the reasons described above.

(DEFINE-RULE SLAVE-PHASE
  (:print-name "SLAVE-PHASE"
   :doc-string ""
   :dependency NIL
   :direction :FORWARD
   :certainty 1.0
   :explanation-string ""
   :priority 800
   :sponsor TOP-SPONSOR)
  (DETECT MANUAL) (SHIFT DETECT)
  (PHASE-IN IS ?PHNUM)
  (EQUIV ?PHNAME ?PHNUM)
 THEN
  (NOW-IS IN-PHASE ?PHNAME)
  (SHIFT OUTPUT) (PHASE-OUT IS ?PHNUM)    )

;; PHASE-TDS
;;      if  current-phase = terminal-climb or enroute-descent, and
;;          squat-switch = off (i.e., in the air), and
;;          gamma < -1.0 degrees, and
;;          epr < 1.4, and
;;          baro-altitude < 10000'
;;      then
;;          next-phase = terminal-descent

(DEFINE-RULE PHASE-TDS?
  (:print-name "Terminal Descent Test"
   :doc-string ""
   :dependency T
   :direction :FORWARD
   :certainty 1.0
   :explanation-string ""
   :priority 100
   :sponsor TOP-SPONSOR)
  (DETECT AUTO)
  (OR
   (NOW-IS IN-PHASE TERM-CB) (NOW-IS IN-PHASE ENR-DS))
  (SQUAT DISCRETE-IS OFF)
  (GAMMA IS ?G) (EPR IS ?E) (ALTITUDE IS ?BALT)
  (< ?G -10) (< ?E 140) (< ?BALT 5000)
 THEN
  (PHASE NEXT TERM-DS))
```

```
;; PHASE-TCB
;;      if  current-phase = takeoff or terminal-descent, and
;;          squat-switch = off (i.e., in the air), and
;;          gear = off (i.e., up), and
;;          gamma > = 1.0 deg., and
;;          flaps < = 15. deg., and
;;          baro-alt < 10000', and
;;          radar-alt > 400'
;;      then
;;          next-phase = terminal-climb

(DEFINE-RULE PHASE-TCB?
  (:print-name "Terminal Climb"
   :doc-string ""
   :dependency T
   :direction :FORWARD
   :certainty 1.0
   :explanation-string ""
   :priority 100
   :sponsor TOP-SPONSOR)
  (DETECT AUTO)
  (OR
   (NOW-IS IN-PHASE TAKEOFF) (NOW-IS IN-PHASE TERM-DS))
  (SQUAT DISCRETE-IS OFF) (GEAR DISCRETE-IS OFF)
  (GAMMA IS ?G) (FLAPS IS ?F) (RADAR-ALT IS ?RA)
  (ALTITUDE IS ?BALT) (> = ?G 10) (< = ?F 15)
  (> = ?RA 200) (< ?BALT 5000)
 THEN
  (PHASE NEXT TERM-CB))

;; PHASE-TO
;;      if  current-phase = taxi or land, and
;;          flaps < = 30. deg., and
;;          reversers = off, and
;;          epr > 1.8, and
;;          radar-alt < = 400'
;;      then
;;          next-phase = takeoff


(DEFINE-RULE PHASE-TO?
  (:print-name "Takeoff Test"
   :doc-string ""
   :dependency T
   :direction :FORWARD
   :certainty 1.0
   :explanation-string ""
   :priority 100
   :sponsor TOP-SPONSOR)
  (DETECT AUTO)
  (OR
   (NOW-IS IN-PHASE TAXI) (NOW-IS IN-PHASE LAND))
  (TREVERSE DISCRETE-IS OFF) (EPR IS ?E) (FLAPS IS ?F)
  (RADAR-ALT IS ?RA) (> ?E 180) (< = ?F 30) (< = ?RA 200)
```

```
 THEN
  (PHASE NEXT TAKEOFF))

;; PHASE-TAXI
;;      if  current-phase = takeoff or land, and
;;          squat-switch = on (i.e., on the ground), and
;;          gear = on (i.e., down), and
;;          flaps < = 15 deg., and
;;          -1.0 < gamma < 1.0
;;          epr < 1.8, and
;;          radar-alt < = 10'
;;      then
;;          next-phase = taxi

(DEFINE-RULE PHASE-TAXI?
  (:print-name "TAXI-TEST"
   :doc-string ""
   :dependency T
   :direction :FORWARD
   :certainty 1.0
   :explanation-string ""
   :priority 100
   :sponsor TOP-SPONSOR)
  (DETECT AUTO)
  (OR
   (NOW-IS IN-PHASE TAKEOFF) (NOW-IS IN-PHASE LAND))
  (SQUAT DISCRETE-IS ON) (GEAR DISCRETE-IS ON)
  (EPR IS ?E) (GAMMA IS ?G) (FLAPS IS ?F)
  (RADAR-ALT IS ?RA) (< ?E 180) (< ?G 10) (> ?G -10)
  (< = ?F 15) (< = ?RA 5)
 THEN
  (PHASE NEXT TAXI))

;; PHASE-LAND
;;      if  current-phase = takeoff or land or terminal-descent, and
;;          gear = on (i.e., down), and
;;          flaps > = 15 deg., and
;;          gamma < 0.0, and
;;          epr > 1.8
;;      then
;;          next-phase = land

(DEFINE-RULE PHASE-LAND?
  (:print-name "LAND-TEST"
   :doc-string ""
   :dependency T
   :direction :FORWARD
   :certainty 1.0
   :explanation-string ""
   :priority 100
   :sponsor TOP-SPONSOR)
  (DETECT AUTO)
  (OR
   (NOW-IS IN-PHASE TAKEOFF) (NOW-IS IN-PHASE TERM-DS)
```

```
   (NOW-IS IN-PHASE TERM-CB)) (GEAR DISCRETE-IS ON)
  (EPR IS ?E) (GAMMA IS ?G) (FLAPS IS ?F) (< ?E 180)
  (< = ?G 0) (> = ?F 15)
 THEN
  (PHASE NEXT LAND))

;; PHASE-EDS
;;      if  current-phase = cruise or enroute-climb, and
;;          squat-switch = off (i.e., in the air), and
;;          gear = off (i.e., up), and
;;          gamma < -1.0, and
;;          epr < 1.4, and
;;          baro-altitude > = 10000'
;;      then
;;          next-phase = enroute-descent

(DEFINE-RULE PHASE-EDS?
  (:print-name "Enroute Descent"
   :doc-string ""
   :dependency T
   :direction :FORWARD
   :certainty 1.0
   :explanation-string ""
   :priority 100
   :sponsor TOP-SPONSOR)
  (DETECT AUTO)
  (OR
   (NOW-IS IN-PHASE CRUISE) (NOW-IS IN-PHASE ENR-CB))
  (SQUAT DISCRETE-IS OFF) (GEAR DISCRETE-IS OFF)
  (GAMMA IS ?G) (EPR IS ?E) (ALTITUDE IS ?BALT)
  (< ?G -10) (< ?E 140) (> = ?BALT 5000)
 THEN
  (PHASE NEXT ENR-DS))

;; PHASE-CRUISE
;;      if  current-phase = enroute-climb or enroute-descent, and
;;          squat-switch = off (i.e., in the air), and
;;          gear = off (i.e., up), and
;;          flaps = 0
;;          -1.0 < gamma < 1.0, and
;;          epr > 1.2, and
;;          baro-altitude > = 10000'
;;      then
;;          next-phase = cruise

(DEFINE-RULE PHASE-CRUISE?
  (:print-name "CRUISE-TEST"
   :doc-string ""
   :dependency T
   :direction :FORWARD
   :certainty 1.0
   :explanation-string ""
   :priority 100
   :sponsor TOP-SPONSOR)
```

```
  (DETECT AUTO)
  (OR
   (NOW-IS IN-PHASE ENR-CB) (NOW-IS IN-PHASE ENR-DS))
  (SQUAT DISCRETE-IS OFF) (GEAR DISCRETE-IS OFF)
  (FLAPS IS 0) (EPR IS ?E) (GAMMA IS ?G)
  (ALTITUDE IS ?BALT) (> ?E 120) (> ?G -10) (< ?G 10)
  (> = ?BALT 5000)
 THEN
  (PHASE NEXT CRUISE))

;; PHASE-ECB
;;      if  current-phase = terminal-climb or enroute-descent or
;;                    cruise, and
;;          squat-switch = off (i.e., in the air), and
;;          gear = off (i.e., up), and
;;          gamma > = 1.0, and
;;          epr > 1.2, and
;;          baro-altitude > = 10000'
;;      then
;;          next-phase = cruise

(DEFINE-RULE PHASE-ECB?
  (:print-name "Enroute Climb"
   :doc-string ""
   :dependency T
   :direction :FORWARD
   :certainty 1.0
   :explanation-string ""
   :priority 100
   :sponsor TOP-SPONSOR)
  (DETECT AUTO)
  (OR
   (NOW-IS IN-PHASE TERM-CB) (NOW-IS IN-PHASE CRUISE)
   (NOW-IS IN-PHASE ENR-DS)) (SQUAT DISCRETE-IS OFF)
  (GEAR DISCRETE-IS OFF) (EPR IS ?E) (GAMMA IS ?G)
  (ALTITUDE IS ?BALT) (> ?E 120) (> = ?G 10)
  (> = ?BALT 5000)
 THEN
  (PHASE NEXT ENR-CB))

;; TEST-PHASE-IN sets SHIFT READY if the system is in automatic detection
;; mode, if its still time for DETECT rules, and if a new phase (i.e, PHASE NEXT
;; ?) exists in the assertion lists.  A SHIFT READY allows the system to fire the
;; prioritization rules below that address the possibility of multiple choices of flight
;; phases being selected.

(DEFINE-RULE TEST-PHASE-IN
  (:print-name "PHASE DETECTED"
   :doc-string ""
   :dependency NIL
   :direction :FORWARD
   :certainty 1.0
   :explanation-string ""
   :priority 50
```

```
   :sponsor TOP-SPONSOR)
  (DETECT AUTO) (SHIFT DETECT) (PHASE NEXT ?)
 THEN
  (SHIFT READY))

;; NO-PDETECT takes care of the no phase TRANSITION being detected. Since
;; all of the phase rules are "transition-in" rules, this rule is true most of the time.
;; When this rule is true, the previous phase is still valid. This rule then sets SHIFT
;; OUTPUT.

(DEFINE-RULE NO-PDETECT
  (:print-name "NO-PDETECT"
   :doc-string ""
   :dependency NIL
   :direction :FORWARD
   :certainty 1.0
   :explanation-string ""
   :priority 50
   :sponsor TOP-SPONSOR)
  (DETECT AUTO) (SHIFT DETECT) (UNKNOWN (PHASE NEXT ?))
 THEN
  (SHIFT OUTPUT))

;; EXCESS-NEXTS takes care of retracting NEXT phases when more than one
;; exists that are not equal to the previous phase. NOTE: If multiple next phases
;; exists, this rule will always be true and will always retract each of the NEXT
;; phases since none will ever equal the previous phase (see note in rule above).
;; The rule is somewhat valid since the phase detection rules should be written
;; where this case can never be true. And, if this case is ever true (but it's not) then
;; the retraction of all NEXT which forces it to stay in the previous phase is valid.

(DEFINE-RULE EXCESS-NEXTS
  (:print-name "EXCESS-NEXTS"
   :doc-string ""
   :dependency NIL
   :direction :FORWARD
   :certainty 1.0
   :explanation-string ""
   :priority 45
   :sponsor TOP-SPONSOR)
  (SHIFT READY) (PHASE NEXT ?X) -> ?S (PHASE NEXT ?Y) -> ?R
  (LAST PHASE ?Z) (NOT-EQUAL ?X ?Y) (NOT-EQUAL ?Z ?X)
  (NOT-EQUAL ?Z ?Y)
 THEN
  (RETRACT ?R) (RETRACT ?S))

;; ESCAPE works with the other rules above as follows. If all the next phases have
;; been retracted, then there is no longer a NEXT PHASE. This rule asserts the
;; warning and shifts control to ERROR. Note: No errors of this nature were
;; reported during the flight tests.

(DEFINE-RULE ESCAPE
  (:print-name "ESCAPE"
   :doc-string ""
```

```
   :dependency NIL
   :direction :FORWARD
   :certainty 1.0
   :explanation-string ""
   :priority 42
   :sponsor TOP-SPONSOR)
  (SHIFT READY) (UNKNOWN (PHASE NEXT ?))
 THEN
  (PROHIBITED PHASE WARNING) (SHIFT ERROR))

;; GET-NEXT takes care of asserting the one-and-only PHASE NEXT (the higher
;; priority rule have ascertained that if it gets here, there is only one PHASE
;; NEXT) as the new NOW-IS IN-PHASE. It also sets SHIFT SET.

(DEFINE-RULE GET-NEXT
  (:print-name "Get Next Phase"
   :doc-string ""
   :dependency NIL
   :direction :FORWARD
   :certainty 1.0
   :explanation-string ""
   :priority 20
   :sponsor TOP-SPONSOR)
  (SHIFT READY) (PHASE NEXT ?PH) -> ?R
 THEN
  (NOW-IS IN-PHASE ?PH) (SHIFT SET)
  (RETRACT ?R)   )

;; P-CONVERT uses the EQUIV lookup table to assert the numerical value of the
;; flight phase in the INSTANCE IS ANALOG. SHIFT is then set to OUTPUT.

(DEFINE-RULE P-CONVERT
  (:print-name "P-CONVERT"
   :doc-string ""
   :dependency NIL
   :direction :FORWARD
   :certainty 1.0
   :explanation-string ""
   :priority 20
   :sponsor TOP-SPONSOR)
  (SHIFT SET) (NOW-IS IN-PHASE ?PH) (EQUIV ?PH ?NUM)
 THEN
  (SHIFT OUTPUT)
  (INSTANCE IS IS ANALOG WITH PHASE-OUT ?NUM))

;;
;; r-symbls.lsp -- dolm: 1-19-90
;;
;; Contains all of the PFD symbol selection logic. There is no SHIFT control
;; checks in these rules. All the rule's dependency are set. This set of rules fire
after
;; the phase detection rules. Note that the error logic also transitions here.

;; NOT-TAXI just sets the assertion NOW-IS NOT-IN-PHASE TAXI if the a/c is
```

```
;; any other phase.  This is a poor substitution for the lack of a "not" logical in the
;; GW package.

(DEFINE-RULE NOT-TAXI
  (:print-name "Not in Taxi phase"
   :doc-string ""
   :dependency T
   :direction :FORWARD
   :certainty 1.0
   :explanation-string ""
   :priority 10
   :sponsor TOP-SPONSOR)
  (OR
    (NOW-IS IN-PHASE TAKEOFF)
    (NOW-IS IN-PHASE TERM-CB)
    (NOW-IS IN-PHASE TERM-DS)
    (NOW-IS IN-PHASE ENR-CB)
    (NOW-IS IN-PHASE ENR-DS)
    (NOW-IS IN-PHASE CRUISE)
    (NOW-IS IN-PHASE LAND) )
  THEN
   (NOW-IS NOT-IN-PHASE TAXI) )

;; HORIZONTAL DEVIATION {A} - if  nav. path = valid
;;                            hor switch = on
;;                            not in phase taxi
;;                            FLARE GUIDE symbol = off
;;                            XTK-DEV2 symbol = off
;;                            LOC-DEV symbol = off
;;                            radar-alt > 260'
;;                         then
;;                            HOR-DEV symbol = on

(DEFINE-RULE TTFIM-HOR-DEV{A}
  (:print-name "Display Hor Dev"
   :doc-string ""
   :dependency T
   :direction :FORWARD
   :certainty 1.0
   :explanation-string ""
   :priority 0
   :sponsor TOP-SPONSOR)
  (NAVPATH2 VALID-IS ON) (HOR SWITCHED ON)
  (NOW-IS NOT-IN-PHASE TAXI)(FLARE-GUIDE SYMBOL OFF)
  (XTK-DEV2 SYMBOL OFF) (LOC-DEV SYMBOL OFF)
  (RADAR-ALT IS ?H) (> ?H 130)
 THEN
  (HOR-DEV SYMBOL ON))

;; HORIZONTAL DEVIATION {B} - if  not in phase taxi
;;                            nav. path = valid
;;                            hor switch = on
;;                            FLARE-GUIDE symbol = off
;;                            XTK-DEV2 symbol = off
```

```
;;                    LOC-DEV symbol = off
;;                 then
;;                    HOR-DEV symbol = on

(DEFINE-RULE TTFIM-HOR-DEV{B}
  (:print-name "Display Hor Dev"
   :doc-string ""
   :dependency T
   :direction :FORWARD
   :certainty 1.0
   :explanation-string ""
   :priority 0
   :sponsor TOP-SPONSOR)
  (NOW-IS IN-PHASE TAKEOFF) (NAVPATH2 VALID-IS ON)
  (HOR SWITCHED ON) (FLARE-GUIDE SYMBOL OFF)
  (XTK-DEV2 SYMBOL OFF) (LOC-DEV SYMBOL OFF)
 THEN
  (HOR-DEV SYMBOL ON))

;; CROSSTRACK DEVIATION {2} - if  phase = TAXI or TAKEOFF
;;                    tka mode = 2 or 4
;;                    xtk switch = on
;;                    FLARE GUIDE symbol = off
;;                    horizontal path mode < > 3
;;                 then
;;                    XTK-DEV2 symbol = on

(DEFINE-RULE TTFIM-XTK-DEV2{SEL-TX-TO}
  (:print-name "XTK-DEV2"
   :doc-string "(TKsel - TKa/c) mid-priority"
   :dependency T
   :direction :FORWARD
   :certainty 1.0
   :explanation-string ""
   :priority 0
   :sponsor TOP-SPONSOR)
  (OR (NOW-IS IN-PHASE TAXI) (NOW-IS IN-PHASE TAKEOFF))
  (OR (TKA MODE-IS 2) (TKA MODE-IS 4)) (XTK SWITCHED ON)
  (FLARE-GUIDE SYMBOL OFF) (H-PATH MODE-IS ?H)
  (NOT-EQUAL 3 ?H)
 THEN
  (XTK-DEV2 SYMBOL ON))

;; CROSSTRACK DEVIATION {2} - if  tka mode = 2 or 4
;;                    xtk switch = on
;;                    FLARE-GUIDE symbol = off
;;                    horizontal path mode < > 3
;;                    radar-alt > 260'
;;                 then
;;                    XTK-DEV2 symbol = on

(DEFINE-RULE TTFIM-XTK-DEV2{SEL-ANY}
  (:print-name "XTK-DEV2 (TKsel"
   :doc-string "(TKsel - TKa/c)  mid priority"
```

```
   :dependency T
   :direction :FORWARD
   :certainty 1.0
   :explanation-string ""
   :priority 0
   :sponsor TOP-SPONSOR)
  (OR (TKA MODE-IS 2) (TKA MODE-IS 4))
  (XTK SWITCHED ON) (FLARE-GUIDE SYMBOL OFF)
  (H-PATH MODE-IS ?M) (NOT-EQUAL 3 ?M) (RADAR-ALT IS ?HT)
  (> ?HT 130)
 THEN
  (XTK-DEV2 SYMBOL ON))

;; CROSSTRACK DEVIATION {4} - if  phase = land
;;                        xtk switch = on
;;                        localizer = valid
;;                        FLARE-GUIDE symbol = off
;;                        radar-alt > 260'
;;                     then
;;                        XTK-DEV4 symbol = on

(DEFINE-RULE TTFIM-XTK-DEV4{LAND}
  (:print-name "XTK-DEV4"
   :doc-string "(TKa/c - rwy-hdg) max-priority"
   :dependency T
   :direction :FORWARD
   :certainty 1.0
   :explanation-string ""
   :priority 0
   :sponsor TOP-SPONSOR)
  (NOW-IS IN-PHASE LAND) (XTK SWITCHED ON)(LOC VALID-IS ON)
  (FLARE-GUIDE SYMBOL OFF) (XTK-DEV2 SYMBOL OFF)
  (RADAR-ALT IS ?Q) (> ?Q 130)
 THEN
  (XTK-DEV4 SYMBOL ON))

;; CROSSTRACK DEVIATION { } - if  phase = takeoff
;;                        xtk switch = on
;;                        navigation path = valid
;;                        FLARE-GUIDE symbol = off
;;                        XTK-DEV2 symbol = off
;;                        XTK-DEV4 symbol = off
;;                     then
;;                        XTK-DEV symbol = on

(DEFINE-RULE TTFIM-XTK-DEV{NAV2}
  (:print-name "XTK-DEV"
   :doc-string "(TKnav - TKa/c)  min-priority"
   :dependency T
   :direction :FORWARD
   :certainty 1.0
   :explanation-string ""
   :priority 0
   :sponsor TOP-SPONSOR)
```

```
  (NOW-IS IN-PHASE TAKEOFF) (XTK SWITCHED ON)
  (NAVPATH2 VALID-IS ON) (FLARE-GUIDE SYMBOL OFF)
  (XTK-DEV2 SYMBOL OFF) (XTK-DEV4 SYMBOL OFF)
 THEN
  (XTK-DEV SYMBOL ON))

;; CROSSTRACK DEVIATION { } - if  phase = taxi
;;                        xtk switch = on
;;                        navigation path = valid
;;                        FLARE-GUIDE symbol = off
;;                        XTK-DEV2 symbol = off
;;                        XTK-DEV4 symbol = off
;;                        radar-altitude > 260'
;;                     then
;;                        XTK-DEV symbol = on

(DEFINE-RULE TTFIM-XTK-DEV{NAV}
  (:print-name "XTK-DEV"
   :doc-string "(TKnav - TKa/c)  min-priority"
   :dependency T
   :direction :FORWARD
   :certainty 1.0
   :explanation-string ""
   :priority 0
   :sponsor TOP-SPONSOR)
  (XTK SWITCHED ON) (NAVPATH2 VALID-IS ON)
  (NOW-IS NOT-IN-PHASE TAXI) (FLARE-GUIDE SYMBOL OFF)
  (XTK-DEV2 SYMBOL OFF) (XTK-DEV4 SYMBOL OFF)
  (RADAR-ALT IS ?H) (> ?H 130)
 THEN
  (XTK-DEV SYMBOL ON))

;; VERTICAL PATH          - if  vrt switch = on
;;                        navigation path = valid
;;                        G/S symbol = off
;;                        REF-ALT = off
;;                     then
;;                        VERT-PATH symbol = on

(DEFINE-RULE TTFIM-VERT-PATH{CWS}
  (:print-name "Vertical Path - CWS Mode"
   :doc-string ""
   :dependency T
   :direction :FORWARD
   :certainty 1.0   :explanation-string ""
   :priority 0
   :sponsor TOP-SPONSOR)
  (VRT SWITCHED ON) (NAV-PATH VALID-IS ON)
  (G/S-DEV SYMBOL OFF)
  (REF-ALT SYMBOL OFF)
 THEN
  (VERT-PATH SYMBOL ON))

;; REFERENCE ALTITUDE     - if  alt mode = 2, 3, or 4
```

```
;;                      land mode = 3 or 4
;;                      phase < > taxi
;;                      G/S DEV symbol = off
;;                  then
;;                      REF-ALT symbol = on

(DEFINE-RULE TTFIM-REF-ALT{LAND}
  (:print-name "Ref-Altitude in LAND Mode"
   :doc-string ""
   :dependency T
   :direction :FORWARD
   :certainty 1.0
   :explanation-string ""
   :priority 0
   :sponsor TOP-SPONSOR)
  (OR (ALT MODE-IS 2) (ALT MODE-IS 3) (ALT MODE-IS 4))
  (OR (LAND MODE-IS 3) (LAND MODE-IS 4))
  (RALT SWITCHED ON) (AUTO MODE-IS 4)
  (NOW-IS NOT-IN-PHASE TAXI) (G/S-DEV SYMBOL OFF)
 THEN
  (REF-ALT SYMBOL ON))

;; REFERENCE ALTITUDE    - if  alt mode = 2, 3, or 4
;;                      ralt switch = on
;;                      auto mode = 4
;;                      tka mode = 4
;;                      fpa mode = 4
;;                      phase < > taxi
;;                      G/S-DEV symbol = off
;;                  then
;;                      REF-ALT symbol = on

(DEFINE-RULE TTFIM-REF-ALT{FPA}
  (:print-name "Ref-Altitude in FPA Mode"
   :doc-string ""
   :dependency T
   :direction :FORWARD
   :certainty 1.0
   :explanation-string ""
   :priority 0
   :sponsor TOP-SPONSOR)
  (OR (ALT MODE-IS 2) (ALT MODE-IS 3) (ALT MODE-IS 4))
  (RALT SWITCHED ON) (AUTO MODE-IS 4) (TKA MODE-IS 4)
  (FPA MODE-IS 4) (NOW-IS NOT-IN-PHASE TAXI)
  (G/S-DEV SYMBOL OFF)
 THEN
  (REF-ALT SYMBOL ON))

;; REFERENCE ALTITUDE    - if  alt mode = 2, 3, or 4
;;                      a-cws mode = 4 or v-cws mode = 4
;;                      ralt switch = on
;;                      phase < > taxi
;;                      G/S-DEV symbol = off
;;                  then
```

```
;;                        REF-ALT symbol = on

(DEFINE-RULE TTFIM-REF-ALT{CWS}
  (:print-name "Ref-Altitude in CWS Mode"
   :doc-string ""
   :dependency T
   :direction :FORWARD
   :certainty 1.0
   :explanation-string ""
   :priority 0
   :sponsor TOP-SPONSOR)
  (OR (ALT MODE-IS 2) (ALT MODE-IS 3) (ALT MODE-IS 4))
  (OR (A-CWS MODE-IS 4) (V-CWS MODE-IS 4))
  (RALT SWITCHED ON) (NOW-IS NOT-IN-PHASE TAXI)
  (G/S-DEV SYMBOL OFF)
 THEN
  (REF-ALT SYMBOL ON))

;; REFERENCE ALTITUDE      - if  tka mode = 4 or h-path mode = 4
;;                             ralt switch = on
;;                             auto mode = 4
;;                             alt mode = 4
;;                             phase < > taxi
;;                             G/S-DEV symbol = off
;;                         then
;;                             REF-ALT symbol = on

(DEFINE-RULE TTFIM-REF-ALT{ALT}
  (:print-name "Ref-Altitude in AltEng Mode"
   :doc-string ""
   :dependency T
   :direction :FORWARD
   :certainty 1.0
   :explanation-string ""
   :priority 0
   :sponsor TOP-SPONSOR)
  (OR (TKA MODE-IS 4) (H-PATH MODE-IS 4))
  (RALT SWITCHED ON) (AUTO MODE-IS 4) (ALT MODE-IS 4)
  (NOW-IS NOT-IN-PHASE TAXI) (G/S-DEV SYMBOL OFF)
 THEN
  (REF-ALT SYMBOL ON))

;; GLIDESLOPE DEVIATION    - if  a-cws mode = 4 or v-cws mode = 4
;;                             g/s switch = on
;;                             glideslope = valid
;;                             phase = land or terminal descent
;;                         then
;;                             G/S-DEV symbol = on

(DEFINE-RULE TTFIM-G/SLOPE{CWS}
  (:print-name "GlideSlope - CWS Mode"
   :doc-string ""
   :dependency T
   :direction :FORWARD
```

```
   :certainty  1.0
   :explanation-string ""
   :priority 0
   :sponsor TOP-SPONSOR)
  (OR (A-CWS MODE-IS 4) (V-CWS MODE-IS 4))
  (G/S SWITCHED ON)  (G/S VALID-IS ON)
  (OR (NOW-IS IN-PHASE LAND) (NOW-IS IN-PHASE TERM-DS))
 THEN
  (G/S-DEV SYMBOL ON))

;;  GLIDESLOPE DEVIATION    - if  phase = land or terminal-descent
;;                          g/s switch = on
;;                          glideslope = valid
;;                          auto mode = 4
;;                          fpa mode = 4
;;                        then
;;                          G/S-DEV symbol = on

(DEFINE-RULE TTFIM-G/SLOPE{FPA}
  (:print-name "GlideSlope - FPA Mode"
   :doc-string ""
   :dependency T
   :direction :FORWARD
   :certainty  1.0
   :explanation-string ""
   :priority 0
   :sponsor TOP-SPONSOR)
  (OR (NOW-IS IN-PHASE LAND) (NOW-IS IN-PHASE TERM-DS))
  (G/S SWITCHED ON) (G/S VALID-IS ON)
  (AUTO MODE-IS 4) (FPA MODE-IS 4)
 THEN
  (G/S-DEV SYMBOL ON))

;;  GLIDESLOPE DEVIATION    - if  phase = land or terminal-descent
;;                          g/s switch = on
;;                          glideslope = valid
;;                          auto mode = 4
;;                          land mode = 3 or 4
;;                        then
;;                          G/S-DEV symbol = on

(DEFINE-RULE TTFIM-G/SLOPE{LAND}
  (:print-name "GlideSlope - LAND Mode"
   :doc-string ""
   :dependency T
   :direction :FORWARD
   :certainty  1.0
   :explanation-string ""
   :priority 0
   :sponsor TOP-SPONSOR)
  (OR (NOW-IS IN-PHASE LAND)(NOW-IS IN-PHASE TERM-DS))
  (OR (LAND MODE-IS 4) (LAND MODE-IS 3))
  (AUTO MODE-IS 4) (G/S SWITCHED ON)  (G/S VALID-IS ON)
 THEN
```

```
  (G/S-DEV SYMBOL ON))

;; RADAR ALTITUDE        - if  phase = tds, tclb, eclb, eds, or land
;;                           radar-alt < 1300'
;;                       then
;;                           RAD-ALT symbol = on

(DEFINE-RULE TTFIM-RADAR-ALT
  (:print-name "Display RAD ALT Symbol"
   :doc-string ""
   :dependency T
   :direction :FORWARD
   :certainty 1.0
   :explanation-string ""
   :priority 0
   :sponsor TOP-SPONSOR)
  (OR (NOW-IS IN-PHASE TERM-CB) (NOW-IS IN-PHASE TERM-DS)
      (NOW-IS IN-PHASE ENR-CB) (NOW-IS IN-PHASE ENR-DS)
      (NOW-IS IN-PHASE LAND) )
  (RADAR-ALT IS ?Q) (< ?Q 650)
 THEN
  (RAD-ALT SYMBOL ON))

;; RUNWAY IMAGE          - if  phase = term-descent or land
;;                           rwy switch = on
;;                           in-coverage discrete = on
;;                           rwy in nav computer = true
;;                           a/c is w/in coverage cone
;;                           alt < = 5000'
;;                       then
;;                           RWY-IMAGE symbol = on

(DEFINE-RULE TTFIM-RWAY-IMAGE
  (:print-name "Runway Image"
   :doc-string ""
   :dependency T
   :direction :FORWARD
   :certainty 1.0
   :explanation-string ""
   :priority 0
   :sponsor TOP-SPONSOR)
  (OR (NOW-IS IN-PHASE TERM-DS) (NOW-IS IN-PHASE LAND))
  (RWY SWITCHED ON) (IN-COVERAGE DISCRETE-IS ON)
  (RWY-IN-NAV DISCRETE-IS ON) (A/C-TRACK IS ?T)
  (RWY-HEADING IS ?H) (< (- ?T ?H) 41)
  (< (- ?H ?T) 41) (ALTITUDE IS ?A) (< = ?A 2500)
 THEN
  (RWY-IMAGE SYMBOL ON))

;; WAYPOINT STAR         - if  star switch = on
;;                           nav-path = valid
;;                           last-waypoint = false
;;                           waypoint is w/in range
;;                       then
```

```
;;                    WP-STAR symbol = on

(DEFINE-RULE TTFIM-WP-STAR
  (:print-name "WaypointStar"
   :doc-string ""
   :dependency T
   :direction :FORWARD
   :certainty 1.0
   :explanation-string ""
   :priority 0
   :sponsor TOP-SPONSOR)
  (STAR SWITCHED ON) (NAV-PATH VALID-IS ON)
  (LAST-WP DISCRETE-IS OFF)
  (WP-DISPLAYABLE DISCRETE-IS ON)
;;  (WP-ALERT DISCRETE-IS OFF)
  THEN
  (WP-STAR SYMBOL ON))

;; FLARE GUIDE          - if  phase = land
;;                        v-cws mode = 4
;;                        radar-alt < decision height
;;                      then
;;                        FLARE-GUIDE symbol = on

(DEFINE-RULE TTFIM-FLARE{<DEC}
  (:print-name "Display FlareGuide below DecisionHeight"
   :doc-string ""
   :dependency T
   :direction :FORWARD
   :certainty 1.0
   :explanation-string ""
   :priority 0
   :sponsor TOP-SPONSOR)
  (NOW-IS IN-PHASE LAND) (V-CWS MODE-IS 4)
  (RADAR-ALT IS ?H) (DEC-HT IS ?D) (< = ?H ?D)
  THEN
  (FLARE-GUIDE SYMBOL ON))

;; FLARE GUIDE          - if  phase = land
;;                        v-cws mode = 4
;;                        radar-alt < = 200'
;;                      then
;;                        FLARE-GUIDE symbol = on


(DEFINE-RULE TTFIM-FLARE{<200}
  (:print-name "Display FlareGuide below 200' "
   :doc-string ""
   :dependency T
   :direction :FORWARD
   :certainty 1.0
   :explanation-string ""
   :priority 0
   :sponsor TOP-SPONSOR)
```

```
  (NOW-IS IN-PHASE LAND) (V-CWS MODE-IS 4)
  (RADAR-ALT IS ?Q) (< = ?Q 100)
 THEN
  (FLARE-GUIDE SYMBOL ON))

;; LOCALIZER DEVIATION     - if  phase = land or term-ds
;;                             loc switch = on
;;                             localizer = valid
;;                             FLARE-GUIDE symbol = off
;;                             RADAR-ALT > 260'
;;                         then
;;                             LOC-DEV symbol = on

(DEFINE-RULE TTFIM-LOC-DEV
  (:print-name "Display LOC DEV"
   :doc-string ""
   :dependency T
   :direction :FORWARD
   :certainty 1.0
   :explanation-string ""
   :priority 0
   :sponsor TOP-SPONSOR)
  (OR (NOW-IS IN-PHASE LAND)(NOW-IS IN-PHASE TERM-DS))
  (LOC SWITCHED ON)  (LOC VALID-IS ON)
  (FLARE-GUIDE SYMBOL OFF) (XTK-DEV2 SYMBOL OFF)
  (RADAR-ALT IS ?Q) (> ?Q 130)
 THEN
  (LOC-DEV SYMBOL ON))

;; COMMANDED AIR SPEED     - if  cas mode = 2 or 4
;;                         then
;;                             CAS-REF-DIAL symbol = on

(DEFINE-RULE TTFIM-CAS{DIAL}
  (:print-name "Display CAS"
   :doc-string ""
   :dependency T
   :direction :FORWARD
   :certainty 1.0
   :explanation-string ""
   :priority 0
   :sponsor TOP-SPONSOR)
  (OR (CAS MODE-IS 2) (CAS MODE-IS 4))
 THEN
  (CAS-REF-DIAL SYMBOL ON))

;; COMMANDED AIR SPEED     - if  t-path mode = 4
;;                             cas switch = on
;;                             last waypoint = false
;;                             CAS-REF-SYMBOL = off
;;                         then
;;                             CAS-REF-BUF symbol = on

(DEFINE-RULE TTFIM-CAS{WP-BUFF}
```

```
  (:print-name "CAS-SELECT{WP-BUFF}"
   :doc-string ""
   :dependency T
   :direction :FORWARD
   :certainty 1.0
   :explanation-string ""
   :priority 0
   :sponsor TOP-SPONSOR)
  (T-PATH MODE-IS 4) (CAS SWITCHED ON)
  (LAST-WP DISCRETE-IS OFF) (CAS-REF-DIAL SYMBOL OFF)
 THEN
  (CAS-REF-BUF SYMBOL ON))

;;
;; r-output.lsp -- dolm: 1-19-90
;;
;; Contains the rules for initiating uploads, displaying info on development screen,
;; and starting the recycle. This is the last set of rules to fire. After setting SHIFT to
;; OUTPUT, the system falls through the symbol rules (they have higher priority)
;; then control comes here. To start the recycle, SHIFT TOP is set.
;;

;; UPLOAD -- If SHIFT EXIT has been set and a slot has been change (i.e.,
;; STATUS SHOW 1) or an error has occurred, let DATAC code know an upload
;; needs to take place by setting slot ULOAD to YES.

(DEFINE-RULE UPLOAD
  (:print-name "UPLOAD"
   :doc-string ""
   :dependency NIL
   :direction :FORWARD
   :certainty 1.0
   :explanation-string ""
   :priority 0
   :sponsor TOP-SPONSOR)
  (SHIFT EXIT) (OR (STATUS SHOW 1) (ERR-FLG IN-PHASE 1) )
 THEN
  (INSTANCE INIT IS CONTROL WITH ULOAD YES))

;; ERROR-HANDLER -- The only time the error handler will be used is when
;; multiple NEXT PHASEs occur when automatically detecting flight phases. This
;; itself is unlikely since the rule were designed so that this will not occur.

(DEFINE-RULE ERROR-HANDLER
  (:print-name "ERROR-CODE"
   :doc-string ""
   :dependency NIL
   :direction :FORWARD
   :certainty 1.0
   :explanation-string ""
   :priority -200
   :sponsor TOP-SPONSOR)
  (SHIFT ERROR)
 THEN
```

```
  (SHIFT EXIT) (INSTANCE IN-PHASE IS CURRENT WITH ERR-FLG 1)

;; EXIT-ESCAPE -- The exit route used when no changes have to be reported to
;; the output screen.  Sets SHIFT EXIT.

(DEFINE-RULE EXIT-ESCAPE
  (:print-name "NORMAL EXIT ROUTE"
   :doc-string ""
   :dependency NIL
   :direction :FORWARD
   :certainty 1.0
   :explanation-string ""
   :priority -800
   :sponsor TOP-SPONSOR)
  (SHIFT OUTPUT)
 THEN
  (SHIFT EXIT) )

;; RECYCLE-RULE -- Resets the DLOAD, ULOAD, ERR-FLG, and STATUS
;; assertions for the next cycle.  Also sends control back to the top by SHIFT TOP.

(DEFINE-RULE RECYCLE-RULE
  (:print-name "RECYCLE-RULE"
   :doc-string ""
   :dependency NIL
   :direction :FORWARD
   :certainty 1.0
   :explanation-string ""
   :priority -900
   :sponsor TOP-SPONSOR)
  (SHIFT EXIT)
 THEN
  (INSTANCE INIT IS CONTROL WITH DLOAD NO WITH ULOAD NO)
  (INSTANCE IN-PHASE IS CURRENT WITH ERR-FLG 0)
  (INSTANCE SHOW IS CONTROL WITH STATUS 0)
  (SHIFT TOP)  )


;;
;; f-utils.lsp  --  dolm: 1-22-90
;;
;; All daemon functions.  These functions handle overhead operations at what
;; would be considered the system level.
;;

;; ZERO-SET is used as a daemon function to return values to 0 when they have
;; been retracted.  It would have been nice if the system had a feature like this --
;; instead of retracting the assertion, it returned the assertion to its default value.

(defun zero-set (inst slot old new)
 (cond  ( (eq *no-value* (multiple-value-bind (x y) (slot-value inst slot) y ) )
   (setf (slot-value inst slot) 0 ) ) )
 )
```

```
;; OFF-SET is used as a daemon function to return values to OFF when they have
;; been retracted. It would have been nice if the system had a feature like this --
;; instead of retracting the assertion, it returned the assertion to its default value.

(defun off-set (inst slot old new)
 (cond ( (eq *no-value* (multiple-value-bind (x y) (slot-value inst slot) y ) )
  (setf (slot-value inst slot) 'OFF ) ) )
 )

;; TF-SET is a slot fixing daemon. If a true/false slot = 1 then it sets it to 'T. And,
;; if it = 0, it sets it to 'F.

(defun TF-SET (inst slot old new)
 (cond ( (equal 1 (car new)) (setf (slot-value inst slot) 'T ) )
     ( (equal 0 (car new)) (setf (slot-value inst slot) 'F) ) )
)

;; OFF-ON-SET is a slot fixing daemon. If an OFF/ON slot = 1 then it
;; sets it to 'ON. And, if it = 0, it sets it to 'OFF.

(defun OFF-ON-SET (inst slot old new)
 (cond ( (equal 1 (car new)) (setf (slot-value inst slot) 'ON) )
     ( (equal 0 (car new)) (setf (slot-value inst slot) 'OFF) ) )
)

;; f-iobas.lsp -- dolm: 1-22-90
;;
;; TTFIM functions for io operations (i.e., port calls, low-memory access, etc.
;;

;; ----- Download ----
;;
;; For complete load of TTFIM module of data from ioports. Called as daemon.

(defun FULLOAD (inst slot old new)
  (cond ( (equal 'YES (car new)) (do-load))) )

(defun DO-LOAD ()
  (mode-fix M1LIST 'MODE-IS 4 #x133)
  (mode-fix M2LIST 'MODE-IS 4 #x134)
  (mode-fix M3LIST 'MODE-IS 3 #x135)
  (on-off-fix S1LIST 'SWITCHED 7 #x131)   ; switches
  (on-off-fix S2LIST 'SWITCHED 3 #x132)
  (on-off-fix VLIST 'VALID-IS 4 #x136)    ; valids
  (on-off-fix BLIST 'DISCRETE-IS 5 #x137)    ; booleans
  (slot-fix 'PHASE-IN 'IS (read-byte #x130)) ;; new for Phase2x
  (slot-quan 'RADAR-ALT 'IS #x13C)
  (slot-quan 'DEC-HEIGHT 'IS #x13E)
  (slot-quan 'RWY-HEADING 'IS #x148)
  (slot-quan 'A/C-TRACK 'IS #x146)
  (setf SLIST '(TREVERSE SQUAT GEAR))
  (on-off-fix SLIST 'DISCRETE-IS 3 #x14F)
  (slot-quan 'GAMMA 'IS #x140)
  (slot-quan 'EPR 'IS #x142)
```

```
  (slot-quan 'FLAPS 'IS #x14A)
  (slot-quan 'ALTITUDE 'IS #x138) )

;; Final load function for the MODE instance. Accepts a slot-name list as argument
;; -- accepts 'number' & 'addr' as arguments for called function 'make-nlist - where
;; 'number is # of items in N-LIST

(defun MODE-FIX (list inst number addr)
  (mapcar 'SLOT-FIX  list
          (make-inst-list inst number)
          (make-nlist number addr) ) )

;; ON-OFF-FIX is the ultimate level function for single digit (1/0 - t/f) loads.
;; Accepts arguments list = slot name list, number = # items in list, addr = ioport
;; addr.

(defun ON-OFF-FIX (list inst number addr)
  (mapcar 'SLOT-FIX list
          (make-inst-list inst number)
          (make-on/off-list number addr) ) )

;; sets a given slot value in the current GW frame/instance/slot environment

(defun SLOT-FIX (slot inst  value)
  (setf  (slot-value inst slot) value) )

;; load of analog slot 'inst   directly from  port addr.

(defun SLOT-QUAN (slot inst addr)
  (setf (slot-value inst slot) (read-word addr)) )

;; this is called by OTHER-FIX & MODE-FIX to create a list of 'number' items all
;; of 'inst'

(defun MAKE-INST-LIST (inst number)
 (DO (( COUNT number) (I-LIST NIL))
     ((ZEROP COUNT) I-LIST)
   (setf I-LIST (cons inst I-LIST))
   (setf COUNT (- COUNT 1))))

;; MAKE-NLIST (called by MODE-FIX) - gets word from ioport (addr) & converts
;; input (byte) into a list of numbers - each from 1-4 in value using 2 bits per
;; number.

(defun MAKE-NLIST (number addr)      ;; number = # items, addr = ioport
  (setf N-LIST () )                  ;; declare N-LIST - this function
  (setf D-WORD (read-byte addr))
  (DO ( (XNUM number (- XNUM 1) ) );; NUMBER initialized and bound
      ( (= 0 XNUM) N-LIST)           ;; terminate when NUMBER = 0
    (setf N-LIST (cons (nib-to-num D-WORD) N-LIST) );; add to list
    (setf D-WORD (truncate D-WORD 4) ) ) ;; chop 2 bits
   N-LIST )                          ;; return N-List

;; MAKE-ON/OFF-LIST accepts arguments 'number' & 'addr' - passing addr to
```

```
;; read-byte, and 'number' to WORD-TO-LIST function. It creates a list of 1's &
;; 0's in the list TFLIST - bound here but returned as output to caller.

(defun MAKE-ON/OFF-LIST (number addr)
  (setf WORD (read-byte addr) )
  (setf TFLIST (WORD-TO-LIST NUMBER WORD) )
 TFLIST )

;; WORD-TO-LIST. Subroutine converting an input number (byte) into a list of n
;; 1's & 0's, where n may be up to 8 (16) depending on the input number. The
;; 'number' argument is the # of items in the list. The 'input' argument is the data
;; number.

(defun  WORD-TO-LIST (NUMBER INPUT)
 (DO  ( (I-LIST ()) (VALUE INPUT) (INDEX NUMBER (- INDEX 1) ) )
     ( (ZEROP INDEX) I-LIST)
 (SETQ I-LIST (CONS (COND ( (ODDP VALUE) 1) ( (EVENP VALUE) 0))
                    I-LIST) )
     (SETQ VALUE (TRUNCATE VALUE 2) ) )
)

;; NIB-TO-NUM accepts an input number from the caller and converts lowest two
;; bits into a value 1-4 which it returns

(defun NIB-TO-NUM (input)       ;; input is a 2bit number 0-3
  (setq I-NUM 1)            ;; for each bit in input, incr I-NUM
  (cond  (  (oddp INPUT) (setq I-NUM (+ I-NUM 1)) ))
  (setq INPUT (truncate INPUT 2) )
  (cond (  (oddp INPUT) (setq I-NUM (+ I-NUM 2)) ) )
  I-NUM )                   ;; return a number 1-4

(defun READ-BYTE (addr)
  (sys:%ioport  addr nil nil))

(defun READ-WORD (addr)
  (sys:%ioport  addr nil t))
;; Lists naming to read into slots from ioports for TTFIM
;; downloading.

(setf M1LIST '(TKA FPA ALT CAS))
(setf M2LIST '(LAND AUTO A-CWS V-CWS))
(setf M3LIST '(T-PATH V-PATH H-PATH))
(setf VLIST  '(NAVPATH2 NAV-PATH LOC G/S))
(setf BLIST  '(IN-COVERAGE RWY-IN-NAV WP-ALERT WP-DISPLAYABLE
               LAST-WP ))
(setf S1LIST '(CAS XTK LOC HOR G/S VRT RALT))
(setf S2LIST '(AUTO-PHASE STAR RWY))


;; ----- UpLoad -----
;;
;; For uploading data that has changed.  Called as a daemon.
;;
```

```
;; New TTFIM upload daemon.

(defun SEND-DSP-CMD (inst slot old new)
   (cond (  (equal 'YES (car new) ) (write-word #x110 (make-dsp-cmd))  ))
   (cond (  (equal 'YES (car new) )
          (write-byte #x112 (slot-value 'IS 'PHASE-OUT)) ))
   (cond (  (equal 'YES (car new) )
          (write-byte #x113 (slot-value 'IN-PHASE 'ERR-FLG)) ))    )

;; For TTFIM upload .. originally in file \larc\io99bas.lsp

(defun MAKE-DSP-CMD ()
  (setf DSPSLOT '(REF-ALT WP-STAR HOR-DEV G/S-DEV LOC-DEV
        CAS-REF-DIAL  CAS-REF-BUF RWY-IMAGE RAD-ALT VERT-PATH
        FLARE-GUIDE XTK-DEV XTK-DEV2 XTK-DEV4) )
  (setf BINLST '( 1  2  4  8  16  32  64 256 512 1024 2048 4096 8192 16384))
  (setf OUTNUM 0)
  (mapcar 'BIT-SET DSPSLOT BINLST)
   OUTNUM
)

;; Primitives for upload of display command word (bytes) where OUTNUM is
;; unbound here and references a global var from caller.

(defun BIT-SET (slot bnum )
  (cond  ((equal 'ON (slot-value 'SYMBOL slot) )
        (setf OUTNUM ( + bnum OUTNUM) ))
  ( t OUTNUM)) )

;; With these defns. must use hex addr & value

(defun WRITE-BYTE (addr value)
  (sys:%ioport  addr  value nil))

(defun WRITE-WORD (addr value)
  (sys:%ioport  addr  value t))
```

# Bibliography


[**Abbott et al. 1987a**] Abbott, T.S.; Nataupsky, M.; Steinmetz, G.G., Integration of Altitude and Airspeed Information Into a Primary Flight Display vial Moving-Tape Formats, NASA TM-4010, 1987.

[**Abbott et al. 1987b**] Abbott, T.S.; Nataupsky, M.; Steinmetz, G.G., Effects of Combining Vertical and Horizontal Information Into a Primary Flight Display, NASA TP-2783, 1987.

[**Boehm et al. 1984**] Boehm, B.W.; Gray, T.E.; and Seewaldt, T, *Prototyping versus specifying: a multiproject experiment*, IEEE Transactions on Software Engineering, SE-10(3):290-202, May 1984.

[**Ford, 1982**] Ford, T.E.; *Indication and Alerting*, Aircraft Engineering, April 1984.

[**Galitz, 1989**] Galitz, W.O., Handbook of Screen Format Design, third edition, Wellesley, MA: QED Information Science, 1989

[**Gilmore, 1985**] Gilmore, W.E., Human Engineering Guidelines for the Evaluation and Assessment of Video Display Units, NUREG/CR-4227, July 1985.

[**Grove et al. 1986**] Grove, R.D.; White, E.J.; Pollock, K.S.; Farmer, S.W., Real-Time Simulation User's Guide, NASA Langley Research Center Central Scientific Computing Complex Document R-1, June 1986.

[**Knox & Cannon, 1980**] Knox, C.E.; Cannon, D.G., Development and Test Results of a Flight Management Algorithm for Fuel-Conservative Descents in a Time-Based Metered Traffic Environment, NASA TP-1717, 1980.

[**NASA NR87-48, 1987**] NASA News Release, No. 87-48, 1987.

[**NASA SP-435, 1978**] *Advance Guidance and Control System (AGCS) Control Mode Panel*, pp18-22, Terminal Configured Vehicle Program Test Facilities Guide, NASA SP-43, 1978.

[**NASA SP-435, 1980**] Terminal Configured Vehicle Program - Test Facilities Guide, NASA SP-435, 1980.

[**Ricks & Abbott, 1987**] Ricks, W.R.; Abbott, K.H., Traditional Versus Rule-Based Programming Techniques: Application to the Control of Optional Flight Information, NASA TM-89161, 1987.

[**Ropelewski, 1982**] Ropelewski, R.R.; *Boeing's New 767 Eases Crew Workload*, Aviation Week & Space Technology, Pilot Report, August 23, 1982.

[**Rushby, 1988**] Rushby, J., Quality Measures and Assurance for AI Software, NASA Contractor Report 4187, October 1988.

[**Steinmetz, 1986**] Steinmetz, G.G.; Development and Evaluation of an Airplane Electronic Display Format Aligned With the Inertial Velocity Vector, NASA TP-2648, 1986.

NASA
National Aeronautics and Space Administration

# Report Documentation Page

| 1. Report No. | 2. Government Accession No. | 3. Recipient's Catalog No. |
|---|---|---|
| NASA TM-102685 | | |

| 4. Title and Subtitle | 5. Report Date |
|---|---|
| Knowledge-Based System for Flight Information Management | June 1990 |
| | 6. Performing Organization Code |
| 7. Author(s) | 8. Performing Organization Report No. |
| Wendell R. Ricks | |
| | 10. Work Unit No. |
| | 505-67-21-07 |
| 9. Performing Organization Name and Address | 11. Contract or Grant No. |
| NASA Langley Research Center<br>Hampton, VA 23665-52225 | |
| | 13. Type of Report and Period Covered |
| 12. Sponsoring Agency Name and Address | Technical Memorandum |
| National Aeronautics and Space Administration<br>Washington, DC 20546 | 14. Sponsoring Agency Code |

15. Supplementary Notes

16. Abstract

This thesis describes the use of knowledge-based system (KBS) architectures to manage information on the primary flight display (PFD) of commercial aircraft. The PFD information management strategy used in this study tailored the information on the PFD to the tasks the pilot performed. This thesis describes the KBS design and the implementation of the task-tailored PFD information management application.

This thesis also describes the knowledge acquisition and subsequent system design of a flight-phase-detection KBS. The flight-phase output of this KBS was used as input to the task-tailored PFD information management KBS. This thesis describes the implementation and integration of this KBS with existing aircraft systems and the other KBS.

In conclusion, this thesis covers the flight tests of both KBS's, collectively called the Task-Tailored Flight Information Manager (TTFIM), which verified their implementation and integration, and validated the software engineering advantages of the KBS approach in an operational environment.

| 17. Key Words (Suggested by Author(s)) | | 18. Distribution Statement |
|---|---|---|
| Knowledge-Based Systems | Primary Flight Display | Unclassified-Unlimited |
| Artificial Intelligence | Flight Test | Subject Category: 59 |
| Information Management | Simulation | |
| Flight-Phase Detection | | |

| 19. Security Classif. (of this report) | 20. Security Classif. (of this page) | 21. No. of pages | 22. Price |
|---|---|---|---|
| Unclassified | Unclassified | 89 | A05 |