AIAA 79-1704R

DC-9-80 Digital Flight Guidance System Monitoring Techniques

Stephen Osder*
Sperry Flight Systems, Phoenix, Ariz.

The DC-9-80 aircraft is equipped with an integrated digital flight guidance system that provides autopilot, flight director, thrust management, speed control, autothrottle, and stability augmentation within a single computer complex. The system architecture is based on two essentially autonomous computer complexes, each able to provide all of these functions, with complete fail-passive monitoring for all flight-critical functions, including category IIIA autoland. This paper describes how the multiplicity of hardware and software monitors are implemented, and how their effectiveness has been analyzed and verified. The system monitors are designed so that fault detection and appropriate shutdown of the failed elements can meet the quantitative safety criteria required by the civil aviation certificating agencies.

Introduction

IN the new generation of digital flight control products now being introduced into commercial air transports, the DC-9-80 Digital Flight Guidance System (DFGS) is somewhat unique in its architecture and monitoring concepts. Architecturally, it goes considerably farther than any contemporary production system in the extent of the functional integration provided within a single computation complex. All automatic stability and control functions, the flight-path control and thrust management functions, and an automated maintenance management capability have been integrated into one digital flight guidance computer. Each computer is designed to be 100% fail-passive for the flight critical functions, including category IIIA autoland. This system exploits self-monitoring techniques to achieve the required fault detection in several areas, including the processor and its associated bus structure. It also uses redundancy in those instances where the required safety criteria could not be met by self-monitoring alone. This paper summarizes the monitoring concepts used, with specific emphasis on self-monitoring techniques.

Block diagrams of this type of architecture are shown in Refs. 1 and 2.

The baseline system consists of two flight guidance computers, each providing autonomous, fail-passive autoland plus all functions listed in Table 1. The second computer provides the failure survivability needed when many functions are integrated within a single computation complex. The crew may select autopilot 1 or 2 for a fail-passive autoland. Each autopilot drives dual electromechanical (velocity-summed) servos. The system also includes dual flight mode annunciators (alphanumeric displays), located above the attitude director indicators, and a status maintenance annunciator mounted outside the primary instrument area. The status maintenance panel is the terminal that performs system-automated testing and interrogates the flight guidance computers to display lists of failures recorded during flight. Special sensors include redundant accelerometer units for guidance state estimation and a second pair of linear accelerometers for yaw angular acceleration measurement.

The monitoring concepts used with the DC-9-80's DFGS have evolved over the past several years to accommodate new hardware mechanizations, but the basic concepts have not changed since the early 1970's. Systems have been implemented and flown, demonstrating and verifying the crucial self-monitoring issues.[2-4] Techniques used in these systems are largely accepted in the industry, and have been incorporated in various digital flight-control applications.[5,6] However, the issues of monitoring comprehensiveness, or how close we have come to 100% fault detection, remain. We can only define a reasonable but quantitative safety requirement, and then relate the monitors to a failure mode and effects analysis (FMEA), showing that the fault-detection comprehensiveness will meet the safety criteria. This is the approach taken in this paper. However, our ability to perform such an analysis relates to the ability of the monitors to detect the so-called "what if" failures which can induce subtle but devastating consequences as a result of transient faults during a critical computation. The DC-9-80 DFGS, as well as prior applications described in Refs. 2-4, contain two types of monitors which specifically cope with transient failures. These are the dual computation techniques and the program flow constraints implemented with the so-called "ticket check" techniques. Without these two techniques, the ability of a self-monitored system to meet safety requirements would be questionable for flight-critical applications. After summarizing the monitoring strategies, an analysis of how the noise-related transient failures are detected will be presented and the results of a quantitative hazard analysis will be summarized, showing that the system monitors will meet a 10^{-9} hazard criterion. (The probability of a safety hazard resulting from a system fault, either detected or undetected, must be less than 10^{-9}.)

System Mechanization Summary

Although we have been referring to self-monitoring, each computer actually employs an internal dual structure and maintains a dual or triplex sensor interface for all flight-critical sensors. It also provides dual actuator and dual flight-

Table 1 System functions provided by each computer

Autopilot (with fail-passive autoland)
Flight director
Yaw damper and turn coordination
Mach trim compensation
Full flight regime autothrottle
Speed command (reference speed management for takeoff and go-around and angle-of-attack "floor" for approach)
Thrust rating
Auto reserve thrust
Altitude alert
Mode annunciation—alphanumeric displays
Maintenance management

Presented as Paper 79-1704 at the AIAA Guidance and Control Conference, Boulder, Colo., Aug. 6-8, 1979; submitted Nov. 2, 1979; revision received May 21, 1980. Copyright © American Institute of Aeronautics and Astronautics, Inc., 1979. All rights reserved.

*Director, Research and Development. Member AIAA.

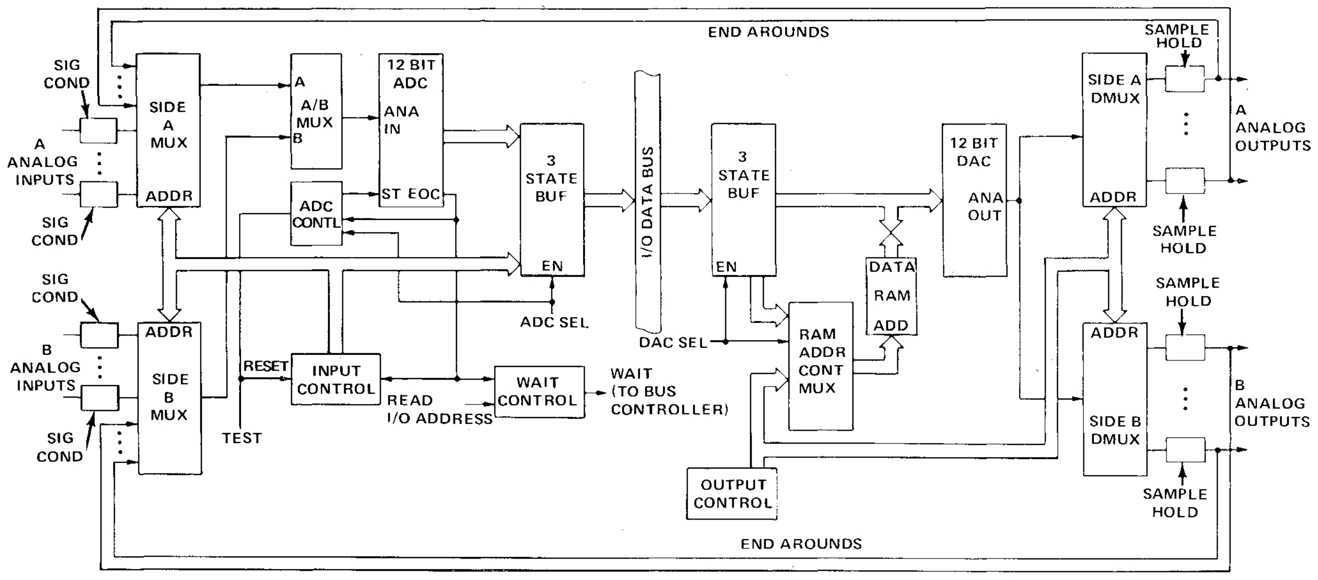

Fig.1 Analog I/O.

director outputs. This dual flow permits comparison monitoring for all critical signal processing until the signals reach the internal nodes or "single-thread" operation where fault detection can be accomplished without resorting to redundant elements. Figure 1, the analog input and output block diagram, illustrates this dual internal structure. The A and B channels, using separate multiplexers, converge to a common analog-to-digital (A/D) converter. The A/D converter interfaces with the input/output (I/O) data bus. This bus in turn interfaces with the internal memory data and address buses. The central processing unit (CPU) also interfaces with the memory and data buses. All dual analog inputs are transferred to channel A and channel B memory locations. Likewise, all dual outputs are derived from separate A and B memory locations. All serial data is converged on a single, continuously tested internal data bus which interfaces with the memory data bus (Fig. 2).

The I/O data bus also interfaces the discrete inputs and outputs with the computer memory. Flight-critical discretes are channelized via the A and B sides, with physical separation maintained until the signals reach the "testable" nodes. Just as the analog outputs are end-around tested via the A/D converter, the flight-critical discrete outputs are fed back to discrete input conditioning circuitry from which they are transferred to memory via the I/O bus.

Dual storage of the critical measurements occurs in location A and location B as shown in Fig. 3, and best estimates of the variable are made by averaging A and B, except when the discrepancy between A and B exceeds specified criteria. Two channels of computation are created internally, and two output control commands are generated. Before an averaged signal is used in a computation, it is passed through an update screen. Any discrepancies in sensor data can activate the update screen. Thus, if the comparison monitor exceeds a threshold, the screen is shut down and control law computations are thereby forced to use prior data. Validity testing includes examination of "valid" discretes for the sensors and rate-of-change constraints (associated with the reasonableness criteria for that sensor). If system test criteria (such as dual computation comparisons) are not satisfied, the outputs are not updated.

Monitor Criteria and Descriptions

The following is a summary of failure detection criteria and the detection techniques used to identify and/or isolate failures in the sensors, electronics/computation, actuators, linkage, and associated wiring.

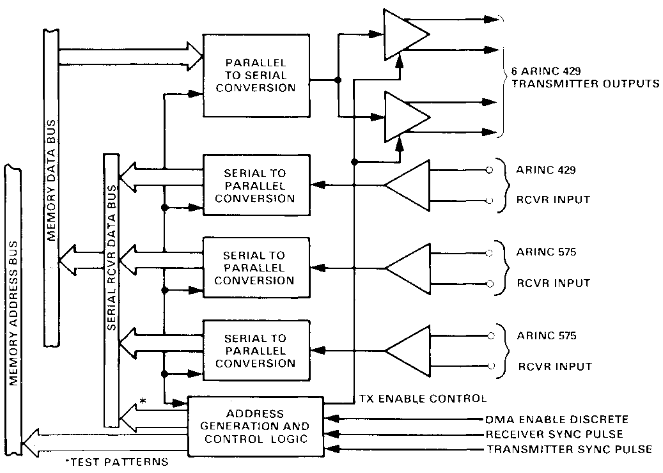

Fig. 2 Serial data I/O.

Sensors and Associated Signal Conditioning

The comparison-monitoring algorithms generate three outputs. They set specific bits in failure words to designate that two specific sensors have indicated a fault. The algorithm sets bits in a "screen" word to prevent updates or computations from using the suspect sensors. It also outputs the computed sensor rate information for the A and B sensors. This rate information is used in the reasonableness algorithms that are part of the fault isolation sequence which follows the detection of a failure by the comparison monitors.

If the comparison algorithms detect a failure, the fault isolation routines are entered. If a validity bit is in the failed state, this constitutes a fault isolation. The system's reversionary logic functions will allow modes and procedures that will accept single-sensor operation, but will disallow the modes if dual sensors are needed.

The reasonableness routines are activated if the validity bit examination did not isolate the failure. Each sensor type has its own reasonableness algorithms. Synchro-type sensors use synchro leg voltage ratio criteria. All sensors use rate-of-change criteria. That rate is compared with a predicted maximum derived from reasonableness considerations or other aircraft state estimates. If sensor A or sensor B fails a reasonableness test, it is considered isolated. As in the case of the validity bit isolation test, the screen word is reset, thereby allowing continuation of the computation process with a

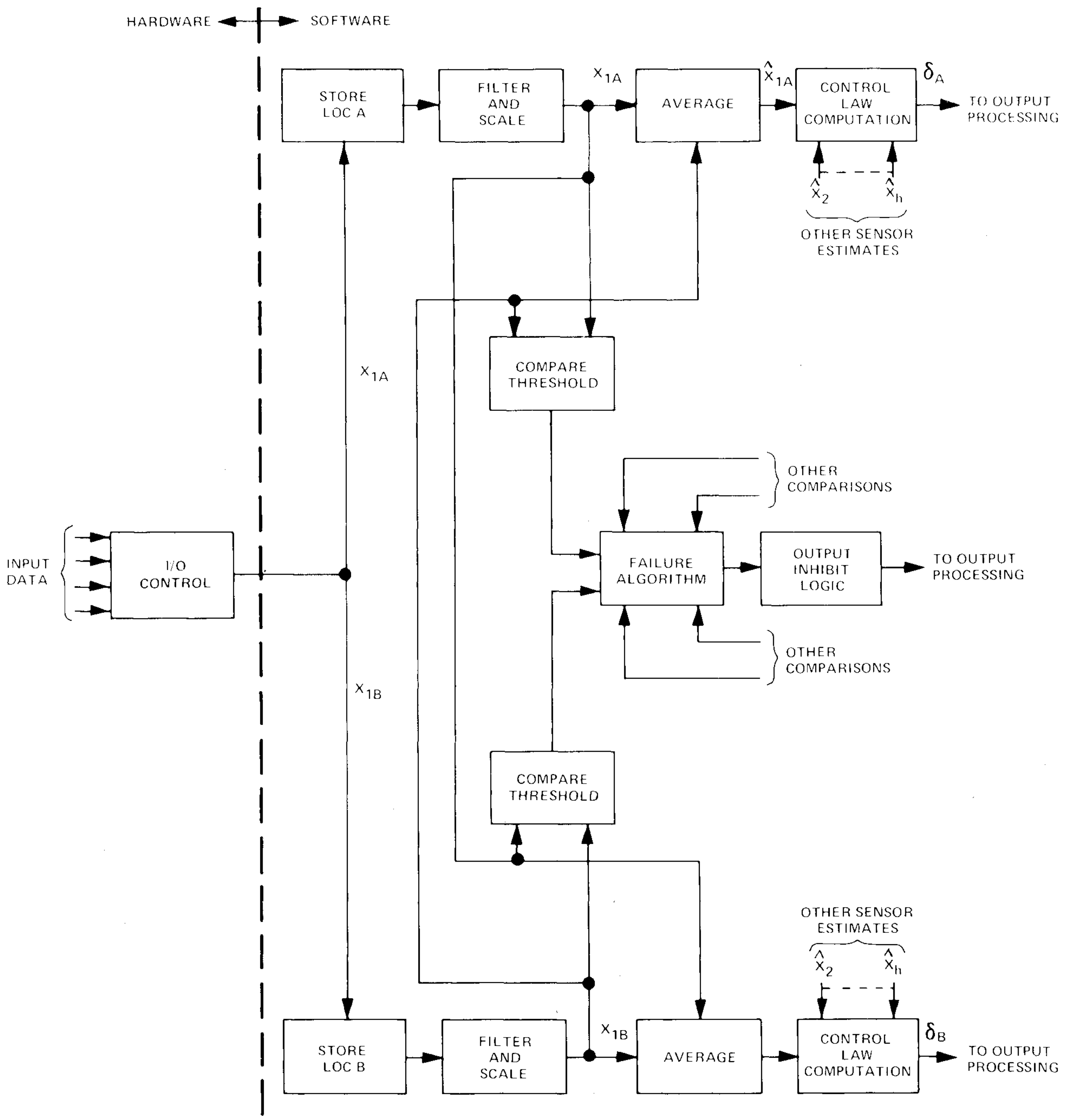

Fig. 3 Dual input and computation flow.

single sensor. If isolation cannot be achieved, then both sensor A and sensor B are considered failed after completion of a timing cycle that started when the failure was first detected. The comparison monitor will allow recovery of a faulty sensor.

All dual sensors are processed through a universal comparison algorithm. There are 46 signal pairs monitored in this manner. This algorithm generates the following information for sensor i.

Estimate of i:

$$\hat{i}=(i_A+i_B)/2\cdot 1/(\tau_i S+1) \tag{1}$$

where i_A is the A-channel measurement of variable i and i_B is the B-channel measurement; τ_i is a noise filter appropriate to each measurement category.

Comparison error:

$$\epsilon_i=\tfrac{1}{2}(i_A-i_B)/(\tau_i S+1) \tag{2}$$

Comparison threshold:

$$T_i=k_i\,|\hat{i}|+\epsilon_{\text{bias}}+F_i \tag{3}$$

where k_i is a scale factor tolerance, ϵ_{bias} represents an allowable bias or offset error, and F_i is a computed function unique to certain sensors. This deterministic error function is based on other aircraft measured states. For example, it would represent a computed gimbal error by a directional gyro and a computed model of a vertical gyro's erection system in the presence of uncertain erection cutoff thresholds and gyro drifts (for the vertical gyro sensors).

When the threshold T_i is exceeded, the sensor screen is activated and the time-magnitude fault detector is initiated. Thus, when the threshold is exceeded, the equation

$$\int_{t=0}^{t_x}|\epsilon_i|\,\mathrm{d}t>T_{M_i} \tag{4}$$

is used as the fault indicator to the failure logging system. The integration interval t_x is controlled by the detection of the threshold

$$|\epsilon_i| > T_i \tag{5}$$

When the error drops below the threshold, the time-magnitude error integrator changes to $\int - |\hat{i}|_L dt$ where $|\hat{i}|_L$ is a deadzone limited value of $|\hat{i}|$. This deadzone limit is equal to another threshold T_i'. The effect of this negative integration term is to allow the time-magnitude fault detection to recover in the event of a temporary failure. The deadzone is needed in the recovery loop to prevent a zero-type failure from automatically recovering when the sensor's correct value returns to zero.

A/D and D/A Conversion

Multiplexer switch failures are, in general, not distinguishable from signal processing or sensor failures. These switch failures are therefore detectable by the sensor monitoring algorithms. The A/D converter can fail by producing bias errors, random or single bit errors, or completely failing to convert—that is, stopping. Any of these failures must be detectable by the monitors.

The monitoring schemes to detect conversion failures are:

Test Words

The converter is asked to encode an array of fixed and variable signals, the correct results being known by the monitoring algorithm. The source of the variable words are the end-around signals used to verify the integrity of the D/A and discrete encoding at the system output. All output signals are fed back to the A/D converter. If the A/D converter has any type of failure mentioned above, all end-around comparisons and all fixed test word comparison checks will fail. (The fixed words are actually positive and negative ramps.)

Dual Comparisons

If the A/D conversion failures were of a transient type, occurring intermittently and only on input words rather than test words, the failure would be detected by the dual sensor comparison checks. Such failures would be diagnosed as sensor failures, and the related function would be shut down.

Program Flow Monitors

If the A/D conversion stopped, then the computer could be held up until it has determined that the A/D conversions are completed. If the computation delay is sufficient to prevent completion of the processor-assigned tasks, then the program flow or so-called ticket check monitors will detect this disruption in normal timing and task execution, and will command a computer shutdown.

End-around monitoring checks every signal output against the digital number which commanded the conversion. This check, therefore, verifies the integrity of the D/A and A/D conversion process as well as all signal paths through the D/A converter, sample hold amplifiers, and input A/D multiplexer. The end-around routines compare the reconverted D/A signals against their stored references, using thresholds derived from the signal processing tolerances (typically less than 1.0%). A failure of the common D/A element will cause a failure of only the end-around signals. A failure of an individual output sample-hold amplifier will cause only that specific signal to fail the end-around test. Such an isolatable failure to a specific output function allows reversion to modes and functions that have not failed.

Processor/Internal Bus Structure/Memory

Since the same circuit elements in the processor and bus structure are used for both flight-critical and nonflight-critical functions, any failure here must result in a computer shutdown. However, there are some areas of program memory that are associated specifically with noncritical functions, and it is possible to detect failures in these sections of memory and shut down only the functions that use the failed memory bits.

The techniques used to detect failures in the digital computer section of the electronics are:

Processor BITE Program

This program executes the entire processor repertoire each iteration cycle of the real-time flight program. It exercises the processor's instructions, using the full range of positive and negative numbers, traversing all directions of the various branching instructions, and employs the entire internal bus structure and associated registers at their maximum (worst-case) data manipulation rates. Any detected failure will result in transfer of execution to the computer fail program which stores the failure type in the maintenance memory, disengages the system, and puts the processor in a wait state.

Heartbeat Detector (External Hardware Monitor)

An external monitor is needed for failures which produce massive error rates. This monitor works by sensing a condition of good health that can be emitted only by a perfectly functioning computer. That condition of good health is referred to as the computer's "heartbeat," which is literally a dynamic beat at the major iteration rate of the real-time flight program. Before the computer starts its next 50-ms iteration of computations, it examines the status of all monitoring algorithm results. If all tests passed, then a software command is issued to output a computed signal. It is a dynamic, multistate signal which appears as a rhythmic pulse or heartbeat. As will be discussed subsequently, all critical computations are performed twice out of separate areas of memory. Hence, there will be two heartbeat outputs (occurring only microseconds apart).

External to the digital computer are two heartbeat detectors which verify that the dynamic heartbeats are occurring with the prescribed waveform and at the prescribed rate. Loss of verification by either heartbeat detector will cause a computer shutdown. In the typical computer failure, the program never reaches the heartbeat-generating algorithm so that the last value of output signal remains as a constant signal (rather than a changing pattern). The heartbeat detectors will diagnose this as a fault and shut down the system. The redundant heartbeat detectors are themselves tested by the computer during power-up initialization.

Memory Checksums

The flight program continuously performs checksums on the entire program memory. Note that memory checksums inherently check the monitoring algorithms which are, physically, no more than fixed programs stored in read-only memories (ROM's). Since only the fixed memory can be tested with the checksum technique, we must find a way to verify that the random access memory (RAM) or data memory is free of failures. All flight-critical data stored in RAM's are stored in and read from dual RAM's. These dual RAM's are combined with a redundant computation technique described next.

Redundant Computation

If we scrutinize the monitoring strategies presented thus far, two potential loopholes loom as a threat to the 100% monitoring objective. First, there is the question of the intermittent failure. This may be stated as "How do we know that a processor or bus structure failure did not occur for a transient duration of say 20 μs after the processor built-in-test equipment (BITE) had satisfactorily executed and passed?" Second, there is the question of temporary computation results which must be stored in the RAM. The organization of the computation to use the dual RAM's also leads to a solution to the transient or intermittent processor failure problem. That solution is to perform all critical computations twice, executing these computations from two entirely different sections of RAM. Dual computation actually started with the task of inputting, scaling, and monitoring the redundant sensor data as illustrated previously by Fig. 3. A complete set of guidance, control, and monitoring computations are performed in channel A, with appropriate channel A elevator, aileron, and rudder output commands stored in RAM A awaiting final monitoring checks before they can be converted to actual servosignals.

When the channel A computations are complete, another set of these computations is repeated for channel B. The B computations continue, using entirely different sections of memory into which and from which the data and results are being transferred. If a random transient fault had occurred during the channel A computations, then the only way the dual computation process could produce the same result in both channels would be for the random failure to recur within a time window synchronized with the computer's microclock, and remain for the identical duration of the original failure transient (with a time tolerance of about 10-20 ns). Even if this highly unlikley phenomenon could occur, the probability of producing the same errors as those which occurred during the first fault is extremely remote because the channel B computations are using an entirely different set of numbers in its addressing than those used by channel A. Even if a failure eluded the dual-channel strategy, there are other protective monitors designed to screen out the harmful consequences of such an unlikely event. They are the ticket check and performance assessment algorithms.

Ticket Check or Program Flow Constraint

The term ticket check is derived from the analogy of a traveler traversing a complex route in which ticket agents punch the traveler's ticket to mark the completion of each leg of the journey. The analogy refers to the flow of a highly constrained program, branching and returning through a maze of very specific fixed and computed addresses. The programming technique used is to set specific bits in "ticket punch" words to verify that the proper subroutine has been performed. This must be done before the address for the next task can be obtained.

A very large percentage of computer operations involve the manipulation of numbers in which at least part of the bits represent an address. Errors in such numbers disrupt normal program flow, and most such disruptions result in a total collapse of program flow. In such a collapse, insufficient order or control remains to even observe the failures that will be found by ticket-check routines. Failure to enter or read correctly from the B registers (indexing functions) or failure of the push/pull operations of the stack are examples of faults that will cause a total collapse of flow. The purpose of the ticket-check routines is, therefore, only to detect the rare and perhaps transient failure condition where only a minor alteration of normal flow occurred. Included in this category of failures is the intermittent type which can prevent completion of the computations within the allocated time.

The 50-ms ticket-check routine is performed for the prior pass at the start of all 50-ms functions. The ticket-check words should have been incremented so that they are now all ones, or in octal notation $(377777)_8$. If they are not all ones, the computer failure routine is called. If the word check passes, all bits of the 50-ms ticket-check words are set to 0 and the ENABLE for the hardware heartbeat is set. The program then moves to the 50-ms routines, but first it performs a supplementary ticket-check on the 50-ms ticket-check routine.

Each 50-ms routine is assigned a bit position in 50-ms ticket-check words. On initialization, all unassigned bits are set to 1. When routine number (i) is successfully completed, that completion is marked by the addition of $[2]^i$ to the appropriate check word. If the program loops through a subroutine one additional time as a result of a transient failure, the ticket-check word will show a wrong result. (If the bit were set rather than added, the check-word result would have been correct.) The 100-ms and other slower loops likewise process their own ticket checks.

Performance Assessment (PA)

Performance assessment algorithms, independent of the basic system's state estimation and guidance equations, are used during autoland. The PA concept is used as an additional safety assurance to satisfy any nonquantifiable doubts. System analyses show that the safety criteria can be met without the PA algorithms. They do, however, provide inherent protection against some primary flight-control linkage failures or excessive wind-shear disturbances. Reference 1 provides a brief summary of how the PA algorithms detect static and dynamic flight-path errors and activate protective warnings and disengagement controls.

Servo Systems

All electromechanical servos are model-monitored, but the elevator, aileron, and rudder parallel servos are dual with an inherent fail-passive velocity summing mechanization. These surface servos are comparison-monitored and, by adding a model monitor, these servos can be fault-isolated to the failed channel. The achievement of safety criteria has been based on the unique mechanical characteristics of the duplex servomechanization, but this subject is beyond the scope of this paper.

Summary of Monitor Interactions

A feature of the monitoring system is that each monitor acts to back up the other monitors. Thus, even if a failure in the processor eluded the BITE program, the dual computation, ticket check, heartbeat detectors, and performance assessment monitors remain to detect any resulting anomaly. The monitors should therefore be viewed as successive screens placed in the path of errors resulting from failures. As described in Ref. 1, a failure does not produce one error, but it produces an error rate. The monitoring screens, therefore, encounter a barrage of errors for a given equipment malfunction (Fig. 4). The probability of nondetection is the probability that not one of these multiple errors will be detected by the successive monitoring screens. If one of these errors penetrates all of the fault detection screens, it can reach the output controls and introduce potentially hazardous control commands. There are no known computer failure models that will allow a failure to penetrate the successive monitoring barriers and produce a hazardous condition. Indeed, such an event is somewhat contradictory. If a dangerous output were produced, the performance assessment computation would detect the results of the erroneous output.

Safety Analysis and the Probability of Nondetection of Flight-Critical Faults

Safety Analysis

In the area of flight-critical functions, the monitors must cause a corrective response for 100% of failures that can cause safety hazards (100% is defined by the statistical probability criteria illustrated in Fig. 5). This figure is an adaptation of the hazard analyses techniques which have been used to certificate flight-critical autoland systems.[7] Although Fig. 5 shows that the fatality risks contributed by pilot response to shutdowns or failures are part of the overall hazard analysis, this aspect of the problem is not the concern of this paper. The monitoring design goal is that the probability of reaching P_{PE1} or P_{PE2} of Fig. 5 shall be less than 10^{-9}.

As seen in the figure, monitoring integrity is primarily concerned with two classes of failures—failure of the shutdown or "disengage" mechanism with a probability defined as P_{DF}, and failure of the monitors to detect erroneous operation defined by the probability of nondetection, P_{ND}. The technology used to ensure a low value of P_{DF} is essentially

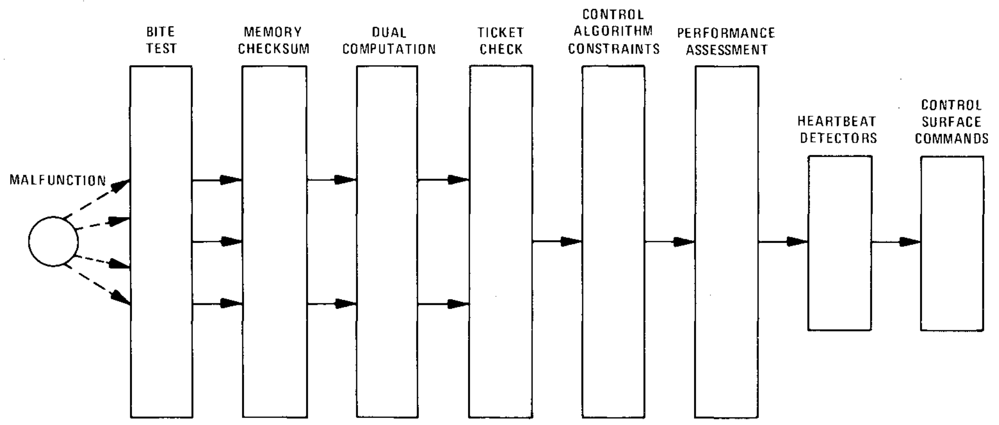

Fig. 4 Interaction of failure monitors (fault consequences must penetrate a sequence of detection barriers).

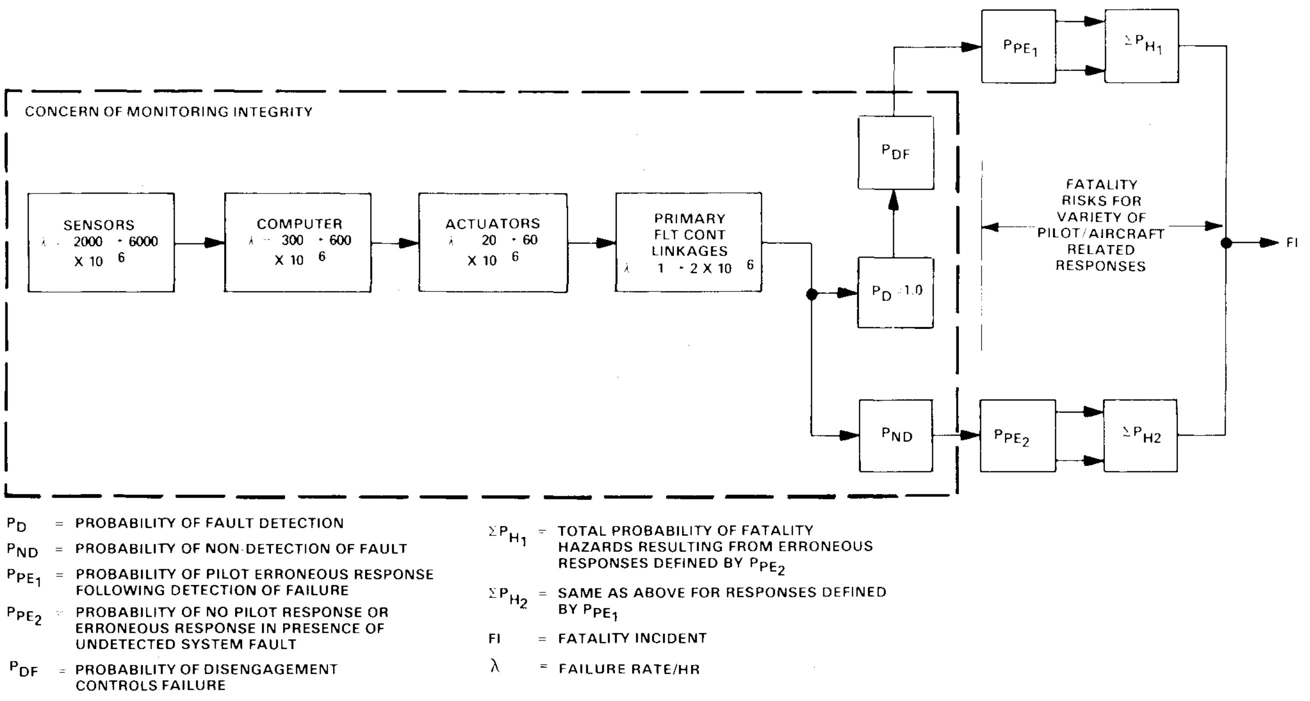
CONCERN OF MONITORING INTEGRITY
SENSORS λ 2000 - 6000 X 10 6
COMPUTER λ 300 - 600 X 10 6
ACTUATORS λ 20 - 60 X 10 6
PRIMARY FLT CONT LINKAGES λ 1 - 2 X 10 6
PDF
PD 1.0
PND
PPE1
ΣPH1
PPE2
ΣPH2
FATALITY RISKS FOR VARIETY OF PILOT/AIRCRAFT RELATED RESPONSES
FI
PD = PROBABILITY OF FAULT DETECTION
PND = PROBABILITY OF NON-DETECTION OF FAULT
PPE1 = PROBABILITY OF PILOT ERRONEOUS RESPONSE FOLLOWING DETECTION OF FAILURE
PPE2 = PROBABILITY OF NO PILOT RESPONSE OR ERRONEOUS RESPONSE IN PRESENCE OF UNDETECTED SYSTEM FAULT
PDF = PROBABILITY OF DISENGAGEMENT CONTROLS FAILURE
ΣPH1 = TOTAL PROBABILITY OF FATALITY HAZARDS RESULTING FROM ERRONEOUS RESPONSES DEFINED BY PPE2
ΣPH2 = SAME AS ABOVE FOR RESPONSES DEFINED BY PPE1
FI = FATALITY INCIDENT
λ = FAILURE RATE/HR

Fig. 5 Conceptual relationship of monitoring integrity to system hazard analysis.

the same as that used in contemporary analog autoland systems. Redundant hardware implementations control the engagement of actuator clutch solenoids, and the integrity of these circuits is verified in pre-engagement tests.

From the range of failure rates illustrated in Fig. 5, it is apparent that the sensors are the primary contributors to the system failure probability. These sensors include attitude references, heading references, localizer and glide slope receivers, radio altimeters, and body-axis acceleration sensors, all of which are dual. If we take the worst-case failure rates shown and allow a 10^{-9} hazard (H) criterion for a 30-s exposure:

$$H=(6662)10^{-6}(30/3600)P_{ND}=10^{-9}$$

$$=55.52\times10^{-6}P_{ND}$$

or

$$P_{ND}=1.8\times10^{-5}$$

If we used the optimistic failure rates, the probability of nondetection can rise to 5×10^{-5} in order to meet the 10^{-9} criterion. This does not mean that every monitor must have a nondetection probability as low as 1.8 to 5.2×10^{-5}. Those elements having the highest λ (failure rates) must have the lowest P_{ND}. A methodology for establishing P_{ND} criteria can therefore be defined, using parameters derived from failure mode and effects analyses.

The hazard contributed by the flight guidance system is

$$H=\Sigma\lambda tP_{ND}+[\Sigma\lambda t]P_{DF}(t) \tag{6}$$

where Σλ is the total failure rate of all system elements (per hour), P_{ND} the effective probability of nondetection, and $P_{DF}(t)$ the probability of failure of disengagement controls (time dependent). The first part of this expression must be derived from a summary of failure effects:

$$\Sigma\lambda tP_{ND}=\lambda_1tP_{ND_1}+\lambda_2tP_{ND_2}+\ldots=\sum_{i=1}^{n}(\lambda_it)P_{ND_i} \tag{7}$$

This form of the expression allows us to define an orderly methodology for computing hazards due to nondetection or disengagement control failures. Let us restrict ourselves to the flight-guidance computer (electronics). The computer elements are divided into functional sections, having failure rates $\lambda_1,\lambda_2,\ldots\lambda_j$. Each of these is subdivided into subelements which may be known to have nondetectable failures by monitor 1, but are detectable by monitor 2. For example, the failure of an I/O instruction is nondetectable by the processor BITE, but is detectable by the end-around and other monitors. Thus, if the CPU and associated bus control are defined by λ_I, then those elements not detected by the processor test would appear in the subcategory λ_{Ii}. Failures associated with subcategories are applied individually to the monitoring screens. Let us therefore examine the comprehensiveness of the various monitors in order to define the parameters needed by Eq. (7).

Monitor Comprehensiveness

BITE Comprehensiveness

The effectiveness of test programs and their validation has been treated extensively in the literature on fault-tolerant computing. It has been claimed that 100% comprehensiveness can be achieved for known failure mechanisms (Ref. 8, for example), although questions always arise regarding subtle transient failures as opposed to "hard" failures.

It is a fundamental principle of the DC-9-80 DFGS's self-monitoring concept that the processor BITE test cannot be relied upon for the required 100% detection of computation failures (primarily because of the transient failure effects). The BITE comprehensiveness was evaluated by using a complete logical simulation of the SDP-118 processor (the designation of the DFGS's CPU). Every failure state of each device could be represented by the simulation. Approximately 1100 failed-circuit elements were simulated, and BITE was executed for each failed condition. Also, every individual bit of the microprogram control store was failed and the BITE executed against each failure. There are more than 26,000 bits in the control store, of which more than 10,000 are "Don't Care" bits because they are not used by the microprogram. More than a million runs were performed to evaluate the effect of certain control store failures that are not immediately detected by the BITE program. Some of these failures are state sensitive, and will produce a BITE-detectable failure only a small percentage of the time. However, a detailed analysis of the failure effects shows that other system monitors will detect the failure effects that elude BITE. The failure mode and effects analyses (FMEA) for the processor employed a methodology which classified effects induced by the systematic insertion of thousands of failures. Six generic classes were definable for failures not detected by the BITE. They are:

A—redesigned CPU BITE program can detect fault
B—newly confirmed "Don't Care" case
C—state-sensitive computation errors resulting in asymmetric A and B channel effects
D—state-sensitive addressing errors resulting in program flow disruption
E—detected by BITE only under certain states
F—state-sensitive uncertain errors—requires specific test against flight program.

An example of how this methodology was applied to control store bit failures is shown in Table 2 which uses the mnemonic notation of the SDP-118 processor instructions and the abbreviations for various machine registers. The table illustrates the method of analysis, using several specific cases that were encountered. This table shows that all of the failures listed were detectable by the system design. Reference 1 discusses some other failure effects that were not detectable by BITE, and describes how other monitors specifically detected those effects. Some class F cases have been found by the FMEA. A selected repertoire of such cases has been entered into a flight computer, and the detectability will be verified in a closed-loop validation facility (six-degree-of-freedom simulator plus all system hardware interfaces). This final test is being performed as part of the certification activity. Note that more than 100 control store bits must fail a generic class F test in order to fail a 10^{-9} hazard criterion for a 3-min autoland exposure. If class F cases existed and proved to elude the dual computation and ticket-check monitors, the performance assessment monitor would detect their effects.

Memory Checksum

The memory checksum will directly detect faults in memory, although a large percentage of such faults will cause effects that are detectable by the ticket check and heartbeat detector since most instructions involve manipulation of addresses. The checksum can be defeated by multiple failures that occur during the checksum timing interval. The probability of such multiple failures defines the probability of nondetection, P_{ND}, for memory checksums.

Reference 1 showed that if λ_m is the failure rate of m words of memory with each word length $=n$ bits, then the memory

Table 2 Computer FMEA techniques control store failure analysis (cases not detected by CPU BITE)

Control store location	Correct destination	Failed destination	Failure effect on computation	Generic class
72	151	155	Instruction & description: <u>BJP</u>—IF $(B)\neq 0$, $(B)-1\rightarrow B, +EA\rightarrow P$ Instruction executes, but also increments SP by 2	D
672	662	666	Instruction & description: <u>ADDUB</u>—$(AU)+(B)\rightarrow B$ Instruction executes using an extra cycle without creating an error—no failure	B
547	777	775	Instruction & description: <u>LRTAL</u>—left rotate (AL) K bits Executes instruction, but also sets compare designator, effecting the incorrect interpretation of a following jump instruction.	A
407	777	377	Instruction & description: <u>CLRILR</u>—(ILR) bit $=\theta$ for (K)bit $=1$ Instruction executes, but also destroys (AL) by: $(AL)+M(SP)\rightarrow AL$	C or A
420	231	233	Instruction & description: <u>ADDLSP</u> $(AL)+M(SP-K)\rightarrow AL$ If no bus contention, executes correctly. For more than one element requesting access at same time, $(AL)+M(BR\&INS+B)\rightarrow AL$ will result from incorrect value at MAD	B, E or C

Abbreviations:	EA	= effective address	B	= index register
	AL	= lower accumulator	BR	= base register
	AV	= upper accumulator	M( )	= memory defined by ( )
	SP	= stack pointer	MAD	= memory address register

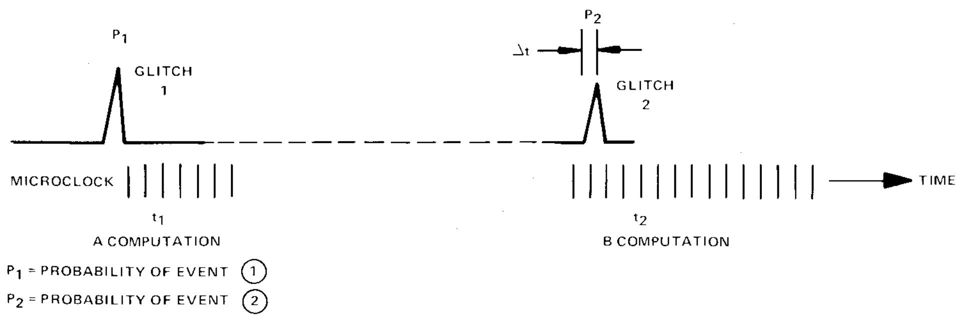

Fig. 6 Definition of glitch repetition requirement needed to elude one pass of dual-computation monitor.

checksum could be defeated by multiple-bit failures, the probability of the nondetection being

$$P_{ND} \approx \tfrac{1}{4}(m-1)\lambda_B(\Delta t) \quad (8)$$

where

$$\lambda_B = \lambda_m/(mn) \quad (9)$$

and Δt is the time required for checksum.

It can be shown that the worst-case P_{ND} for a 1000-word segment of memory being checksummed for $\Delta t = 1.0$ s and $\lambda_B \approx 0.05 \times 10^{-9}$ is 0.347×10^{-11} and that the probability of such a nondetected failure event is less than 10^{-17}/h. Similarly, Ref. 1 showed that the probability of nondetection of an RAM bit failure is related to the occurrence of a near-simultaneous compensating failure of a bit in the redundant RAM. The worst-case P_{ND} was shown to be 7.7×10^{-10} and the probability of such a nondetected failure was computed to be less than 3×10^{-14}/h.

Detection of "What If" Failure Cases

The two general classes of "what if" failures are the transient fault with recovery and the so-called "glitch." In the transient failure one postulates a condition where the CPU works perfectly during BITE, but fails only during the 96% of the iteration cycle when the CPU BITE is not running. The "glitch" case is similar, but is the result of a noise-induced dynamic failure in which susceptibility is a random phenomenon related to the occurrence of asynchronous events.

Models for all types of transient failure phenomena have been formulated to analyze the probability of their occurrence. Only one analysis will be presented here—the "glitch" case. The other failure phenomena, in general, give lower P_{ND} than the glitch because they require a more elaborate sequence of time-correlated failures during both the A and B computations.

Reference 1 describes how a glitch can occur in a clocked asynchronous event, generating a noise pulse which disrupts a computation. Figure 6 illustrates how such a failure can elude detection by the dual computations. The first occurrence during the A computations destroys the accuracy of some arithmetic or logical operation. It differs from other types of transient failures because it occurs during one microcycle only. To remain undetected, the same wrong result must be produced during the channel B computations. The probability can be estimated as follows:

1) Assume that dual computations involve 30% of the program—15% associated with channel A and 15% with B.

2) Assume the probability of a glitch-type failure is 20% of probability of CPU failures $= 0.2\lambda_p t$ (λ_p = processor fail rate).

3) Probability of glitch-type failure during channel A computation $= P_1 = (0.15)(0.2)\lambda_p t$.

4) Assume glitch failure now recurs with a mean time between failure (MTBF) of 1.0 s ($\lambda_X = 3600$). (Typical glitch failures dependent upon asynchronous event concurrence would yield MTBF's of 5 min to 2 h.)

5) $P_2 = \lambda_X(\Delta t)$ where $\Delta t = 20 \times 10^{-9}$ s, $P_2 = 20 \times 10^{-9} \times 3600 = 72 \times 10^{-6} = P_{ND}$.

6) $P_1 \times P_2 = 0.03\ \lambda_p t(72 \times 10^{-6}) = 10.8 \times 10^{-11}$/h for typical $\lambda_p = 50 \times 10^{-6}$.

The key point made is that transient failures can only defeat the dual computation monitor by producing failure events during channel B operations which are coherent with identical failure events that occurred during channel A operation. The required time coherence between the channel A and B events is that they encompass the identical computer microcycles. This, in general, requires fault coherence in the 20 ns range. There are random asynchronous events occurring between the A and B computations [I/O bus accesses by the direct memory access (DMA)]. Hence, the balancing second failure event must occur at a random, uncorrelated number of microcycles after the first failure. This results in a similarity to the probability of identical failures in two independent computers.

Summary of Probability of Nondetection

Following the methodology of Eq. (7), we can partition the computer failure rates into the following normalized table (assuming the total computer failure rate is λ).

λ_1	CPU and bus structure	$=0.2\lambda$
λ_2	ROM (approximately 26K)	$=0.2\lambda$
λ_3	RAM (4K)	$=0.1\lambda$
λ_4	A/D and D/A and multiplexer (MUX)	$=0.1\lambda$
λ_5	Servoelectronics (9-servo channels)	$=0.1\lambda$
λ_6	Analog and discrete signal processing	$=0.1\lambda$
λ_7	Serial processing and DMA controller	$=0.1\lambda$
λ_8	Power, switching, and excitation circuitry	$=0.1\lambda$

These percentages are not necessarily precise, but will vary from the true values by no more than a few percent. Specifying each element in terms of the total computer failure rate λ allows us to qualitatively appraise the sensitivity of λ to achieve the 10^{-9} hazard objective described previously. The flight guidance computer λ is in the $250\text{-}400 \times 10^{-6}$ range. Since P_{ND} and P_{DF} are in series with the failure probability, sufficiently low values for these parameters remove the equipment failure rate as a factor in the hazard analysis.

Analyses of the failure modes of all electronic elements of the DC-9-80 DFGS have been performed. A partial summary of results was tabulated in Ref. 1, showing that there were no P_{ND}'s poorer than about 10^{-10}. These results were prior to reaching the monitoring barrier imposed by the performance assessment algorithms. The probability of disengagement controls failure is between 10^{-10} and 10^{-11}. This is based on the probability that two failures in the control circuitry will occur, both after the system has been engaged, since these circuits are tested upon engagement.

Conclusions

A combination of processor self-monitoring techniques and conventional redundancy can be used to meet safety criteria for autoland systems. These criteria state that faults and their nondetection cannot occur with a probability greater than 10^{-9}. The methodology needed to verify that safety criteria are met requires a failure modes and effects analysis (FMEA) of the processor and its related bus structure and memories. The ability to achieve the extremely high-monitoring comprehensiveness does not depend upon the thoroughness of the processor built-in-test (BITE) program, but upon creating a hardware and software structure that simulates two separate computers while using only one processor. The analysis verifying monitoring comprehensiveness was performed for the DC-9-80 DFGS, but the technique is not restricted to any specific processor. Since a detailed processor FMEA is required to verify the monitoring effectiveness, the ability to perform the same type of analysis on the newer single-chip microprocessors is doubtful.

References

[1]Osder, S.S., "The DC-9-80 Digital Flight Guidance System's Monitoring Techniques," *Proceedings of the AIAA Guidance and Control Conference*, Aug. 1979, pp. 64-79.

[2]Osder, S.S., Mossman, D.C., and Devlin, B.T., "Flight Test of a Digital Guidance and Control System in a DC-10 Aircraft," *Journal of Aircraft*, Vol. 13, Sept. 1976, pp. 676-686.

[3]Humphrey, R.B., "Phase II YC-15 Digital and Analog SCAS Configuration," Douglas Aircraft Co. Rept. No. MDC-J7207.

[4]Vandenberg, L.J., "Advanced Aerial Refueling Boom/Advanced Aerial Refueling Nozzle System—Subsystem Hazard Analysis," Douglas Aircraft Co. Rept. No. MDC-J7356, Sept. 1976.

[5]Westermeier, T.F., "In-Line Monitoring of Digital Flight Control Computers," McDonnell Aircraft Co., *NAECON 78 Proceedings*, May 1978, p. 62.

[6]Hendrick, R. and Hill, C., "Self-Testing Digital Flight Control Applications," AIAA Paper 75-568, AIAA Digital Avionics Systems Conference, April 1975.

[7]Warren, D.V., "Safety Criteria for Fail-Operational Autoland Systems and their Application," *Integrity in Electronic Flight Control Systems*, AGARDograph No. 224, 1977.

[8]Mersten, G.S., McGough, J.G., and Jeungoh, S., "Achievable Error Detection through Self-Test Software," *NAECON 78 Proceedings*, May 1978, p. 1231.