(NASA-TM-72860) DIGITAL FLY-BY-WIRE FLIGHT CONTROL VALIDATION EXPERIENCE (NASA) 241 p HC A11/MF A01 CSCL 01C

N79-14109

Unclas
G3/08 40986

NASA Technical Memorandum 72860

DIGITAL FLY-BY-WIRE FLIGHT CONTROL VALIDATION EXPERIENCE

Kenneth J. Szalai, Calvin R. Jarvis, Gary E. Krier, Vincent A. Megna, Larry D. Brock, and Robert N. O'Donnell

December 1978

National Aeronautics and
Space Administration

Page intentionally left blank

NASA Technical Memorandum 72860

DIGITAL FLY-BY-WIRE FLIGHT CONTROL VALIDATION EXPERIENCE

Kenneth J. Szalai, Calvin R. Jarvis, Gary E. Krier
Dryden Flight Research Center

and

Vincent A. Megna, Larry D. Brock, and Robert N. O'Donnell
The Charles Stark Draper Laboratory, Inc.

National Aeronautics and
Space Administration

TABLE OF CONTENTS

SECTION 1

INTRODUCTION

The development of the procedures and policies for the validation and certification of aircraft with an advanced electronic flight-control system will be one of the most important tasks that must be accomplished before the considerable advantages of these systems can be realized. It will be necessary to develop the validation methods as early as possible so that the designers of these advanced control systems will fully understand the reliability requirements and the means that will be available to demonstrate that reliability. The regulatory authorities will also need to be aware of what is being developed and anticipate the data and testing they will require to demonstrate compliance with the regulations. The authorities will be very reluctant to certify a new system for which there is no precedent without ample assurance that flight safety can be assured. On the other hand, airframe companies and the airline customers will be reluctant to commit to the use of an advanced system, in spite of large potential advantages, unless flight safety can be assured and there are no large risks and unreasonable costs in obtaining certification. In order to give both the users and the regulatory authorities the necessary confidence, it is necessary for the validation methods to be developed along with the development of the systems themselves.

The following report gives an account of the experience of the NASA Dryden Flight Research Center (DFRC) and The Charles Stark Draper Laboratory (CSDL) in validating a digital fly-by-wire (DFBW) system. The experience was gained from the development and testing of a DFBW system that is being extensively flight tested in an F-8 aircraft. This report is written in the hope that the experience gained will contribute to the confidence that must be established in these systems before they can be put into commercial service.

This report was written in support of the Federal Aviation Administration (FAA) Advanced Integrated Flight Systems (AIFS)

Technology Program. This program was established by the FAA in anticipation of the impact of advanced systems technology on airworthiness standards and certification procedures. The objectives of the AIFS Technology Program are to:

(1) Evaluate and assess advancing technology for impact on FAA.

(2) Support the development of airworthiness standards and certification procedures.

(3) Disseminate technical information within FAA.

Critical issues relating to airworthiness considerations being addressed by the FAA are:

(1) Systems failure modes and failure effects.

(2) Hardware and software reliability, including verification and validation.

(3) Lightning, electromagnetic, and other transient effects.

(4) Aircraft flight characteristics and performance.

(5) Structural aspects of active controls.

The AIFS program consists in large part of the monitoring of relevant activities of interest at NASA, Department of Defense, and industry. The NASA/DFRC Advanced Flight Control Program on the F-8 aircraft is one such activity that has been selected to support the AIFS program.

This report first gives a general discussion of the directions in which advanced avionics systems are moving (Section 2). This discussion includes the great expansion in critical functions proposed for these systems and the advances in technology that make these functions possible. A brief outline is then given of some of the other advanced systems being developed to provide a context for understanding the F-8 DFBW system. Section 3 contains a brief history of the NASA Advanced Flight Control Program. The system requirements and the qualification requirements for the system are given in Section 4. This section also includes a description of the system in order that the qualification process might be better understood. Section 5 gives the system flight-qualification experience which is the primary purpose of the report. The qualification experience includes the validation of the design, hardware testing, software verification, the total system validation using both ground testing in an "iron-bird" system and flight tests. The report concludes with general comments on validation methods, including a review of validation techniques and tools, comments on the special considerations for software, and comments on the formation of a total validation program and what goals and requirements it must meet.

SECTION 2

DIRECTIONS IN ADVANCED AVIONICS SYSTEMS

This section will give a brief discussion of one view of the new emerging technology of advanced avionics systems. The primary purpose of this overview is to define the context into which the NASA F-8 Digital Fly-By-Wire (DFBW) program fits. It is hoped this discussion will contribute something to an increased mutual understanding of just what kind of equipment and systems are beginning to evolve and to help define some of the concepts and terms.

It is helpful to define the context of the F-8 program to arrive at a realistic assessment of where this experience would apply and where it would not. It is not desired to create the impression everything that was done applies to all fly-by-wire systems. On the other hand much of this experience will be applicable to many aspects of the systems that will be introduced into commercial aviation.

This overview will briefly describe the developing advanced avionics systems from two different viewpoints: one is the new and expanding functions that these new systems are expected to perform; the other is the change in the technology used in the equipment that performs these functions. These viewpoints will then be combined to give some typical system configurations. In all of these discussions particular emphasis will be placed on the implications for validation and certification. Before beginning these discussions, brief definitions are given for some of the terms used.

2.1 Definitions and Terminology

One of the first steps toward understanding a new technology and entering into fruitful dialogue is to define the terminology used; see Table 2-1. In many cases, there may not be complete agreement within the aviation industry on the meaning of some of these terms. This

Table 2-1. Definitions.

Term	Definition
Acceptance	The determination by the user (customer) that the product meets his requirements.
Active Control System	A system which actively commands the movement of control surfaces on the basis of sensor inputs to provide some function or characteristic not available in the aircraft passively.
Assembler	A program which translates mnemonic assembly language instructions into the binary instructions used by the processor, assigns values to named addresses, and performs other functions as an aid to the programmer in writing a software program.
Assembly Language	A programming language which uses the set of processor executable instructions in mnemonic format to write the software program.
Autopilot	Equipment which automatically performs functions that were normally performed by a pilot, such as maintaining heading or altitude.
Central System	A system where the majority of functions is performed with a central computer system.
Certification	The determination by a regulatory authority that a product meets the regulations for that product.
Command Augmentation System (CAS)	An active control system that augments the pilot's control inputs with sensor inputs to provide him direct control of aircraft motion rather than control surface position.
Compiler	A program which translates a higher order language into the language of a particular computer and performs the assembler functions.
Computer	A system containing a processor, variable storage memory, program storage memory, input and output interface circuits, and support circuits including control, timing, power supply, etc. The computer can perform a large variety of functions by the sequential execution of a set of basic operations in the processor. The commands for the set of operations is called the software program and is stored in the program memory. (The hardware necessary to convert input signals to the proper digital form and also the hardware necessary to convert the output signals to the proper form is usually included within the definition of a computer.)

Table 2-1. Definitions. (Cont.)

Term	Definition
Control Configured Vehicle (CCV)	An aircraft whose basic aerodynamic and/or structural design includes the use of an active control system.
Control Law	A set of equations which define control surface position as a function of sensed inputs.
Coverage	The probability that, given a fault, the system continues to perform the required function.
Distributed System	A system where the functions are distributed to a number of different computers.
Elastic Mode Suppression (EMS)	Active control to increase the damping of lightly damped structural bending modes excited by gusts.
Electrical Command System	A system in which electrical signals provide the primary control commands but a mechanical backup system is retained.
Fail-Operational	A system which is able to continue to provide critical functions after one failure.
Fail-Passive	A system that does not cause an unsafe condition after a failure. There is no disruption in the aircraft performance. The function just ceases to be performed.
Fail-Safe	A system that does not cause an unsafe condition after a failure. Immediate pilot corrective action may be necessary.
Fail-Soft	A system that does not cause an unsafe condition after a failure. Pilot corrective action may be required within for example 6 seconds.
Failure	A condition which can give rise to a fault, usually considered permanent.
Fault	An anomaly in the performance of a system.
Fault Tolerant	A system which is able to continue to provide critical functions after the occurrence of a fault.
Federated System	A system where different functions are performed by different computers and all are connected together into a total system.

Table 2-1. Definitions. (Cont.)

Term	Definition
Flutter Suppression	Active control to suppress aeroelastic flutter modes.
Fly-By-Wire (limited definition)	The use of electrical signals to connect the pilot's control devices with the control surfaces.
Fly-By-Wire (broader definition)	The use of electrical control connections with no mechanical backup linkages and providing the pilot direct control of aircraft motion rather than control surface position.
Gust-Load Alleviation (GLA)	Active control to reduce loads due to gust.
Higher Order Language	A language which enables a programmer to use simple English phrases in writing a software program. It is not dependent on the particular computer and is more universal than assembly language.
Integrated System	A system in which the design has been integrated to allow the optimum synergistic arrangement for the performance of functions and the use of resources.
Maneuver-Load Control (MLC)	Active redistribution of the increased loads due to maneuvers in order to reduce structural loads.
Microprocessor	A processor contained within one large-scale integrated (LSI) circuit or a small set of LSI circuits.
Processor	Electronic circuits capable of sequentially performing a set of basic arithmetic, logical, and control operations. A processor is the central element of a computer.
Qualification	A formal process whereby a system or aircraft is defined to be ready for flight operations.
Redundancy Management	The process of managing redundant elements in order to identify a failure and then reconfiguring the system to remove the effects of the failed element and continue operation with unfailed elements.
Relaxed Static Stability (RSS)	The use of active control to allow the static stability of the basic unaugmented airframe to be relaxed. The aircraft with the active system operating will have the normal stability margins.

Table 2-1. Definitions. (Cont.)

Term	Definition
Ride-Control System (RCS)	Active control to improve the quality of the ride for the crew and passengers.
Software Program	The set of processor-executed instructions stored in the memory of a computer which cause the computer to perform the desired functions.
Stability Augmentation System (SAS)	An active control system which augments the natural stability of an aircraft.
Transient Fault	A temporary anomaly in the performance of a system.
Validation	The determination that a resulting product meets the objectives that led to the specification for the product. This determination usually includes operation in a real environment.
Verification	The determination that a design meets the specification. Verification is usually a part of the validation process. A simulated environment is often used.

list will be at least give the understanding, held by those associated with this program, which it is hoped, will make this report more readable and will contribute to more understanding within the industry.

2.2 Expansion of Functions

One of the more important things that is happening in advanced avionics systems is a significant expansion in the number and complexity of the functions performed, and in the importance of these functions to flight safety.(1-5) Originally noncritical auxiliary and convenience tasks, tnese functions are now becoming flight critical, meaning ultimately that a complete electronic failure would cause the loss of the aircraft. This expansion is both broadening the scope of the systems validation effort and increasing the importance of the validation.

Most avionics system functions fall within the broad categories of flight control, navigation, communication, and aircraft systems control and monitoring. This report is concerned almost exclusively with the flight-control functions, which can be further roughly divided into those related to augmenting or providing aircraft stability, controlling the path of the aircraft, reducing or controlling the structural loads, and providing the basic control surface linkages. The following subsections briefly describe the functions performed in each of these categories, and include some of the related certifications implications.

2.2.1 Active Stability Control Functions

One of the most basic functions of flight-control equipment is to provide or supplement the basic stability of the aircraft. This function was performed by very early control equipment almost from the beginning of powered flight. An example is the airplane stabilizer of the early 1900s.(6) The equipment performing these functions employs some form of sensor, which measures aircraft motion. This sensor data is processed and the results are applied to actuators, which automatically move the aircraft control surfaces. The result is a modification of the natural aerodynamic characteristics of the vehicle.

The nature of the modification of these characteristics covers wide ranges of complexity and importance. One of the most basic modifications is the use of vertical-gyro output to control the ailerons and hold the wings level. Another modification in this category, which is in very common use today, is a yaw damper to reduce Dutch roll oscillations.

At the present time, advances in technology and confidence in flight-control equipment have developed sufficiently to make possible a more significant role for this equipment. Automatic equipment has been proposed with enough reliability that the basic design of the aircraft can be configured with the assumption that active control is used. These vehicles are called control configured vehicles (CCVs). One active control function in this class is to supplement or provide longitudinal stability. This mode has been called relaxed static stability (RSS); when active control equipment is assumed, the aerodynamic designer can be relieved from providing inherent stability in the airframe itself.

One advantage of this feature is that trim drag can be reduced. To provide stability, most aircraft require that the center of gravity be in front of the center of pressure of the wing (see Figure 2-1). Thus, a negative lift is required on the horizontal stabilizer to maintain balance. There is a contribution to drag both from the negative lift of the tail and the additional lift required from the wing to compensate for the tail. This drag can be reduced by moving the center of gravity nearer to the center of pressure and then providing the required stability with an active control system. Reduced stability also has the advantage of increasing maneuverability, which is particularly valuable in military aircraft. The first production aircraft to employ these functions was the Air Force F-16. There are many other examples of specific active stability control functions, which will not be discussed in detail. All these functions have the same general characteristic: a system made up of a sensor, computer, and control surface actuator, and used to modify the basic aerodynamic characteristics of the vehicle.

The validation and certification requirements for these newer more advanced systems can be much more demanding than those currently in use. It will be necessary for the validation to reflect the degree to which the function being performed is critical to the safety of the aircraft. It is reasonable to expect that the regulatory authorities will demand that it be shown that flight safety will be at least no worse than it is now. If the aircraft is actually unstable or is very

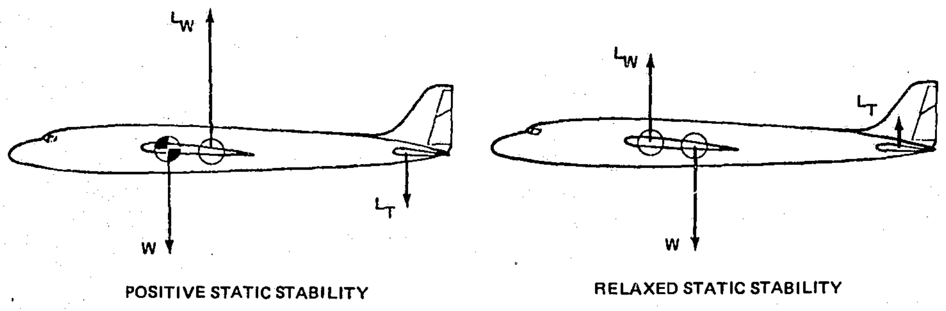


Figure 2-1. Aircraft forces with positive and relaxed static stability.

difficult to fly when the system is not working, then the system would have to demonstrate a reliability that is comparable to the current reliability of a basic airframe with conventional controls. This expanded role for the flight-control system will also expand the scope of the certification process. Flight-control equipment has normally been handled as auxiliary equipment with little interrelationship with the certification of the airframe itself. In many future cases, it will be likely that the aircraft characteristics necessary to meet the regulations will be present only if an electronic system is installed and operating. Thus, the flight-control system will be involved from the beginning in the certification process, and other engineering capabilities besides system specialist will be at least indirectly involved with electronic systems. For example, aerodynamic and stability specialists will have to become familiar with the operation of the electronic active control equipment.

2.2.2 Load-Control Functions

Another class of new active control functions similar to stability control functions is the class of load-control functions. This class includes maneuver load control (MLC), gust load alleviation (GLA), elastic mode control, and flutter suppression. The MLC and GLA modes are normally used with multiple aerodynamic controls along the span of the wing. When a significant maneuver is commanded or a large load is sensed due to a gust, the automatic system redistributes the lift along the span by reducing the lift of the tip and increasing lift at the root in order to reduce wing-bending moments (see Figure 2-2).

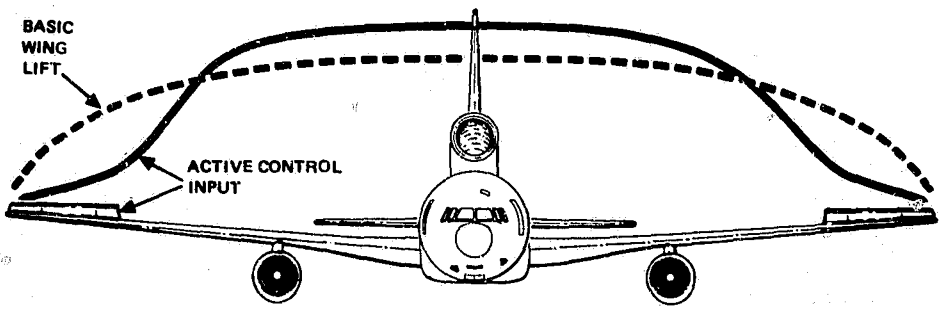


Figure 2-2. Load redistribution with active controls.

Flutter suppression uses a sensor such as an accelerometer near the part of the structure where flutter needs to be controlled. This sensor data is then processed and applied to a control surface that will provide damping for the critical structural modes. A function very similar to GLA is a ride-control system (RCS) except that in the RCS the object is not the reduction of structural loads but an increase in the quality of the ride for the passengers and crew.

The certification situation for these load-control functions will be similar to that for stability functions. As with any equipment, the system must show that, in itself, it will contribute no significant hazard. The further certification requirements will be a matter of the degree of criticality of the function being performed. Load relief may be used only as an auxiliary function to provide, for example, longer structural life. In this case, in order to meet structural requirements, the existence of the active control is not assumed, and thus no more than the basic no-hazard certification requirements are required. However, if the active control is assumed for the aircraft to meet basic structural requirements in all or some regimes of flight, then the equipment will have to demonstrate a higher level of reliability. The level of reliability that must be demonstrated will be proportional to the probability that the function will be flight critical. For example, if a wing is designed structurally such that a flutter-suppression system is required even at cruise speeds, the probability that the function is flight critical is essentially 1. The system would thus have to be shown to be very reliable. However, it will be necessary to show that the use of active control hardware does not increase the total probability of catastrophic failure of the aircraft. The reliability requirement easily fits the 'extremely improbable" category. On the other hand, if the wing is designed to have no flutter mode up through maximum operating velocity (V_{MO}), but does need an active system between V_{MO} and dive velocity (V_D), the functional reliablity requirement would not have to be quite as high. To maintain the same level of reliability, the probability that the aircraft simultaneously exceeded V_{MO} and the system failed would have to be shown to be extremely improbable. If the control system is not continuously monitored, the system reliability used in these failure calculations will have to include all possible failures since the last time the equipment was tested. However, if the system is monitored and a failure is detected, the operational envelope of the aircraft can be reduced, for example, by slowing it to greatly reduce the probability that V_{MO} is exceeded. In this case, it will be necessary to consider system failure only during the time the aircraft is operating above V_{MO}. For this short time, system reliability will be very high.

2.2.3 Automatic Flight-Path Control

Another major category of flight-control functions is that which comprises the guidance and control commands necessary to cause the aircraft to follow or maintain a desired flight path. These functions are sometimes called the "outer loop" functions; the stability augmentation functions are the "inner loop". These outer-loop functions also cover a wide range—from simple heading hold and altitude hold, to the complete control of the aircraft in four dimensions from takeoff to landing.

The basic functions of heading hold and altitude hold have been used for a long time. Their primary purpose is to relieve the pilot from the tedious job of constantly maintaining heading or altitude during long periods of cruise flight. The certification requirements for these basic functions are well known and present no new problems. These functions are not necessary for safe flight. It is thus only necessary to show that there is sufficient warning of a system failure for the crew to disable the system and switch to an alternate system or take over control of the aircraft manually. Any "hard over" failure that would put the aircraft in danger would have to be extremely improbable.

Automatic flight control has been expanded more recently in include other flight regimes besides cruise. The most important example is automatic landing, which is intended primarily to allow landings when the visibility is too low for manual landings. The automatic system must control the aircraft through the most critical phase of flight where the immediate detection of any failure is important and where the possibility of taking over operation manually is extremely difficult or impossible. An unsafe landing due to a system failure must be extremely improbable. Because a single electronic system does not have sufficient reliability to meet the requirements, a fail-operational system is used that can continue to operate after any single failure or combination of failures not shown to be extremely improbable. The validation and certification process to assure that the system meets these requirements can thus become very involved.

2.2.4 Electronic Control Linkage

An additional function being proposed for the electronic flight-control equipment is the linkage between the cockpit controls and the aerodynamic control surfaces. This electronic linkage would replace the present mechanical linkage. This function is what is meant by fly-by-wire to many people. However, fly-by-wire is often used as a more general term to include several of the other active flight-control functions that have been discussed. A NASA/Air Force understanding defines fly-by-wire as a full-time flight-control system with no mechanical reversion with active control which allows the pilot to command aircraft motion directly.

The removal of the mechanical control cables, bell cranks, push rods, etc. has some obvious advantages. One of the primary advantages is the reduction in weight. There is also the potential for reducing maintenance problems due to stretching, bending, and wear in the linkages. The design of the rest of the aircraft can be simplified by removing the constraints imposed by the control cables. The simplifications may be particularly helpful in the cockpit area where the pilot controls may be reduced to improve visibility and usefulness of the instrument panel. Fly-by-wire can also simplify the implementation of some of the more complex active control modes such as maneuver load control, gust load alleviation, and direct lift controls. These modes require the coordinated movement of several control surfaces.

The need for mechanical linkage may have been reduced by the lack of any effective direct link to the actual control surface on many of the latest aircraft. The control linkage moves only the control valves on the hydraulic actuators. A complete hydraulic failure will thus also give a complete control failure.

The primary hindrance to the adoption of electronic control is the difficulty of designing an electronic system with adequate reliability, and then providing the validation of that reliability to the degree necessary to obtain certification and the acceptance by the industry. The basic requirement of extreme improbability would still have to be met. The most stringent requirements for flight control to date have been for autoland. The requirements for a full-time fly-by-wire are much more stringent in two respects. First, the exposure time to a flight-critical failure is greatly increased from less than 1 minute to the time of the entire flight. Also, the consequences of a fault would be much more severe. A complete failure of the automatic system during an autoland will not necessarily be catastrophic. However, the complete failure of the flight-control linkage will almost certainly be catastrophic. There can be no passive failures. The challenge of design, validation, and certification will be correspondingly greater.

2.2.5 Summary of Reliability Requirements for Expanding Active Control Functions

The increasing functional reliability required by expanded active control is summarized in Figure 2-3. In this figure, the major factors contributing to the overall reliability are shown for several representative functions. The numbers shown are to illustrate the concepts. They are intended to be as realistic as possible, but they are not the result of analysis for any particular system.

The probability of a catastrophic result from the failure of a function is assumed to be made up of three major elements:

(1) The probability that a failure is catastrophic.

(2) The exposure time.

(3) The failure rate of the hardware.

The first element is the probability that a functional failure will cause loss of the aircraft or a major loss of life. For example, it is assumed that it is improbable that the failure of a cruise autopilot function will be critical. The probability that a failure

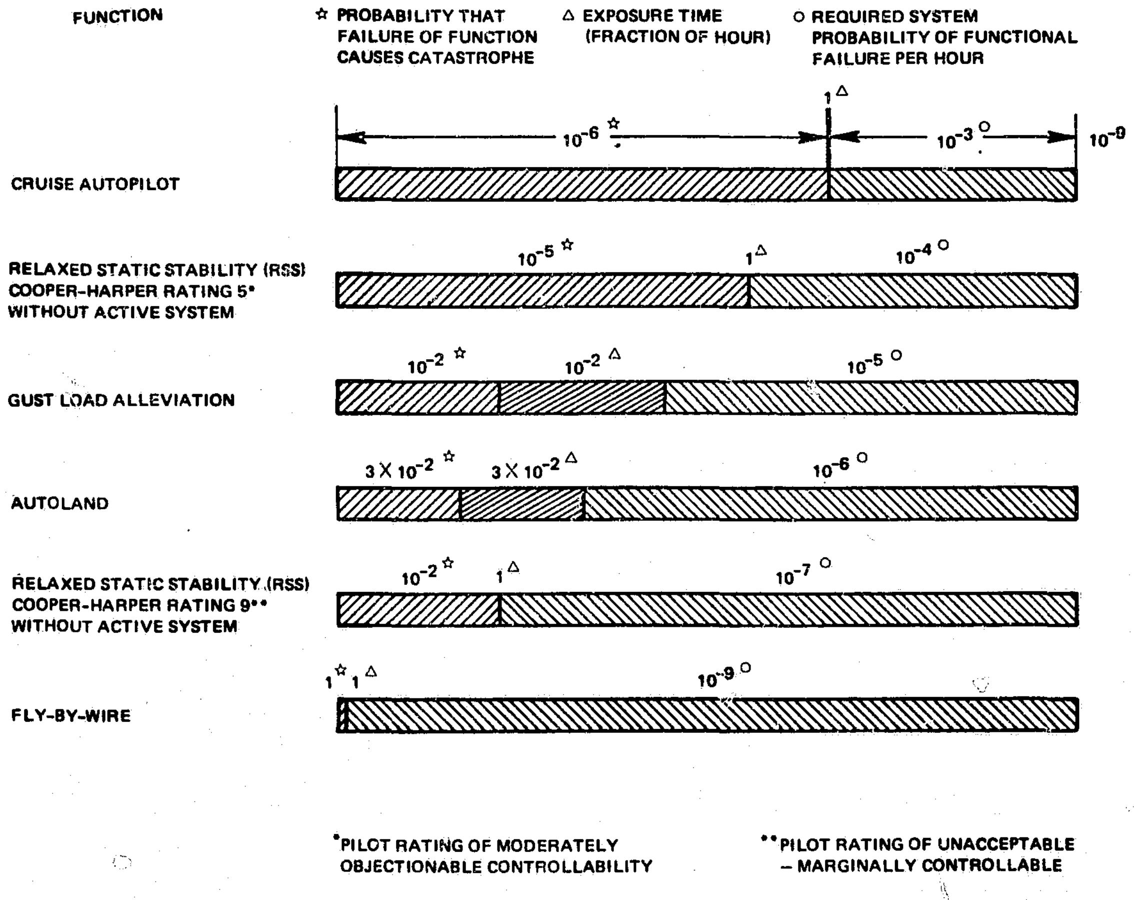


Figure 2-3. Reliability requirements for typical flight control.

of the autoland function will be catastrophic is much higher, but still not certain. To distinguish between equipment failure and functional failure, it is assumed that a fail-operational system is used for the autoland function so that there is no effect from the first equipment failure. A functional failure would only result from a second hardware failure. However, a large percentage of these second hardware failures will not have tragic results during autoland. A majority of the failures would be passive, some causing minor damage. It is assumed that at most 1 in approximately 30 functional autoland failures will be catastrophic. It is, however, assumed that any failure of the fly-by-wire function will be likely to cause the loss of the aircraft. There can be no passive failure when the lost function is the ability to control the aircraft.

The second element contributing to a catastrophic result is the exposure time—the percentage of time that the function is used and that a failure will have any effect. It is assumed that a cruise autopilot is used almost all the time, so that the exposure factor for this function is 1. The exposure factor for fly-by-wire would also be 1. On the other hand, an autoland function is critical for less than 1 minute. For a gust-load-alleviation system, the exposure time would be the probability that a gust is encountered, which causes loads greater than the allowed design loads with the active control system inoperative. The exposure time has to include all of the time since the last time the operation of the function was tested. For failure modes that are continuously monitored, the exposure time would be only the time of the gust itself. At the other extreme, if the function is not tested at all, the exposure factor becomes essentially 1. Any equipment failure before the time of the gust would not be known, and would cause that function not to be available when needed. It is assumed in this example that the function is monitored almost continuously so that the exposure time is the probability of a gust exceeding the limit.

The last element is the required probability of a system failure causing loss of a function. In a single-channel system, a functional failure would be the same as a hardware failure. In a multiple-channel redundant system, a functional failure would be the failure of all channels, any common mode failure, or any other failure that inhibits the operation of the remaining channels.

Figure 2-3 clearly illustrates how the expansion in flight-control functions requires progressively more reliable systems. The effort required to validate this higher reliability is also increasing in a corresponding way. Section 2.3 discusses the technology that makes these reliable systems possible.

2.3 Technology Developments: Digital Technology

The significant change in the technology used to implement the flight-control functions has been due to the fantastic development in electronic technology, especially digital technology.(7) Everyone who has debated the question of when they should buy a calculator cannot help but be impressed by both the increase in functional capability and the decrease in price. It is only natural that these developments would also have a significant impact on flight-control systems.

It is not the intention of this report to give a review of the current developments in electronic technology or to attempt to predict future developments. However, a few basic trends will be mentioned to increase the awareness of the significance of the implications on flight-control equipment. Some of these implications will be discussed with particular emphasis on validation and certification impact.

2.3.1 Trends in Digital Technology

The most significant trend in digital technology is the continual increase in the number of equivalent components that can be placed on a single semiconductor chip. The costs per chip remains relatively constant, which means that the cost per function is dropping significantly. These trends are shown in Figure 2-4.(7) Complexity has been doubling every year for the past several years. Some of the factors allowing this increased density are beginning to reach their theoretical limits, which is likely to slow the growth rate by half. However, a doubling of capability every 2 years gives some fantastic possibilities for the future. Chips are now being developed with 25,000 to 30,000 transistors. These new chips are giving rise to the new term—VLSI (very-large-scale integration).

The most important component made from this technology is the microprocessor. Predicted trends for the computing speed and the cost of a complete microcomputer made from a microprocessor chip is shown in Figures 2-5 and 2-6. Their rapid pace of development will be significant for aircraft systems in at least two ways. On the low end, the microprocessors that currently exist are getting less expensive. This trend will encourage the use of a much larger number of microprocessors in a typical system. Processors are likely to be embedded within sensors, actuators, displays, etc., which will increase their effectiveness. On the high end, microprocessors are getting more capable and will be able to take over the central flight-control computer functions. High-end microprocessors do not exist yet, but they are expected soon and will be able to

(1) Perform more than a half-million instructions per second.

(2) Address more than a million 8-bit bytes of memory.

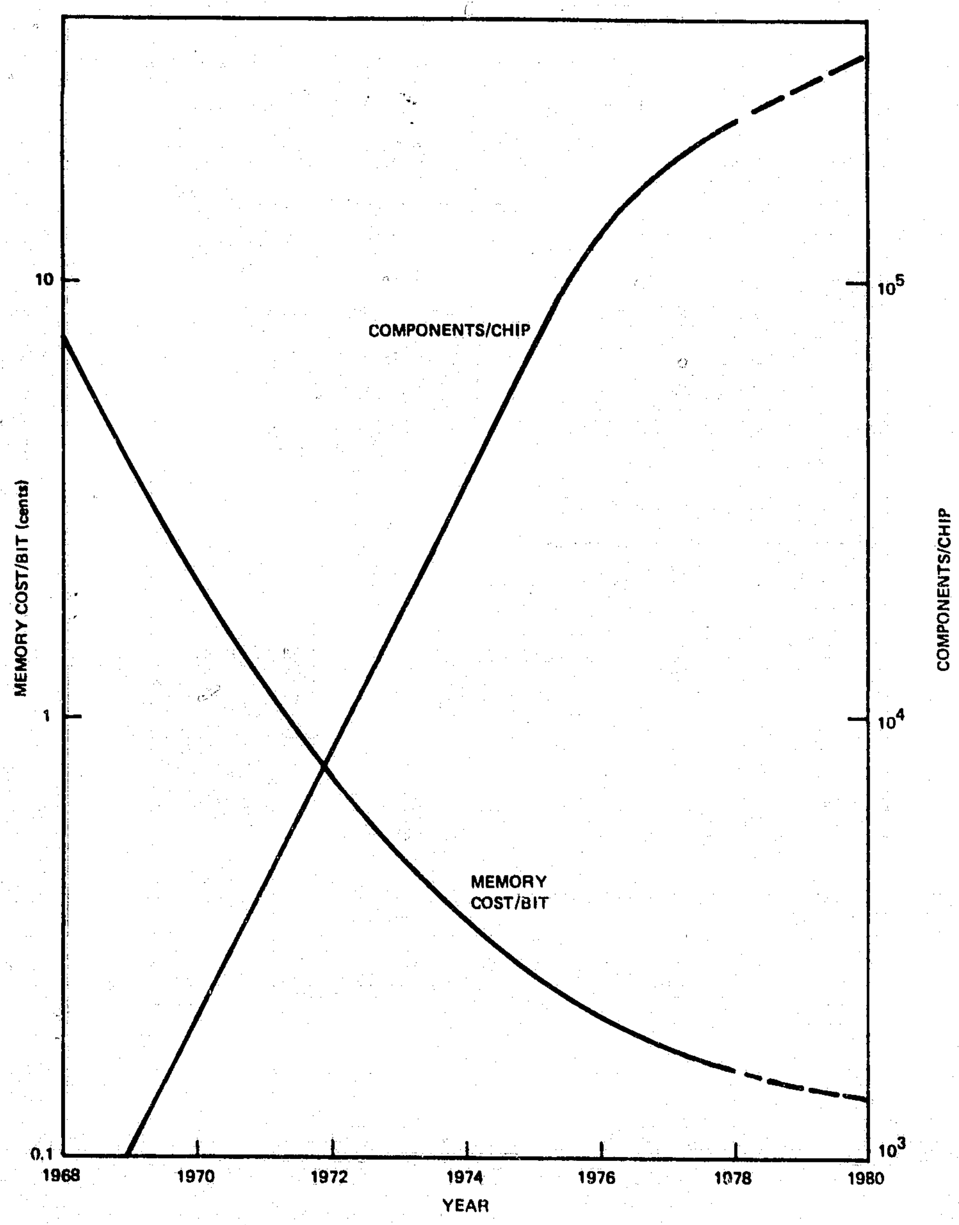


Figure 2-4. Trends in electronics—component densities.

(3) Operate on 16-bit data.

(4) Have a large instruction set, including multiply and divide.

(5) Have on-chip program memory of up to 32,000 bits.

The advances in technology that are making the cost per function significantly less does not necessarily mean that total systems costs are going down. It is most likely that the costs for total flight-control electronics will stay constant or even rise. The big change will be in the capabilities of the systems. The real significance of the technical advances is that the high performance and very high fault tolerance needed for the advanced active control functions can be obtained for reasonable costs. The compatibility of these two trends

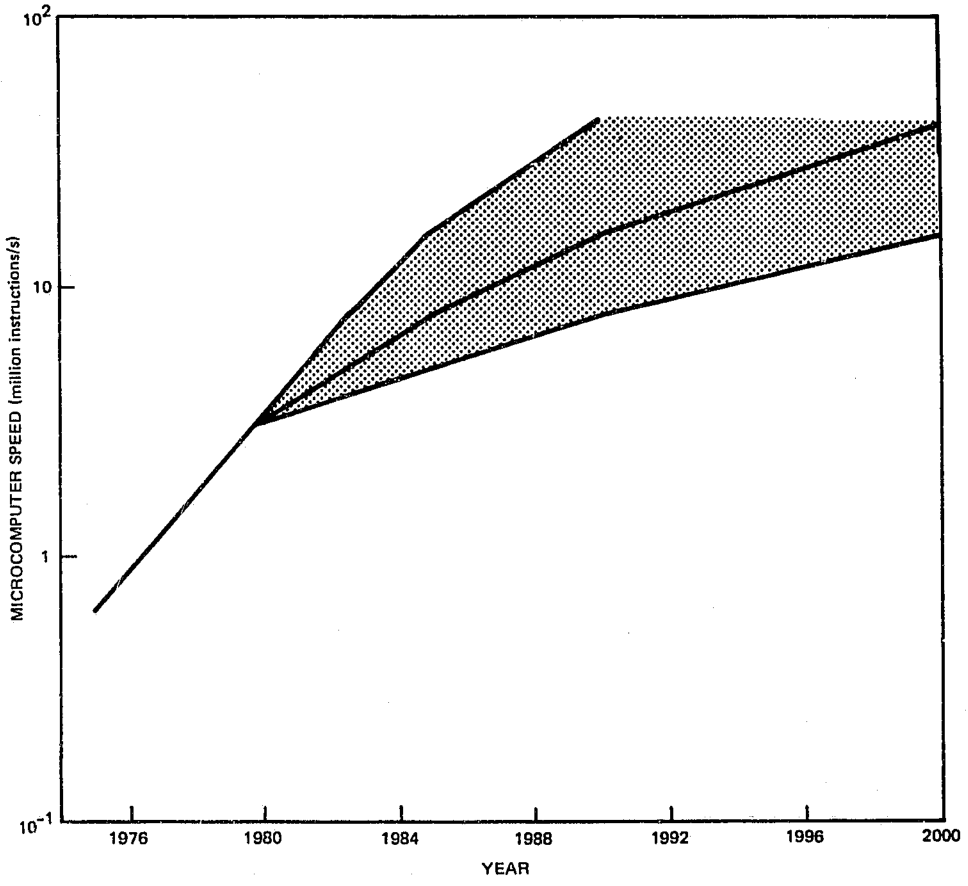


Figure 2-5. Forecasts of microcomputer speed.

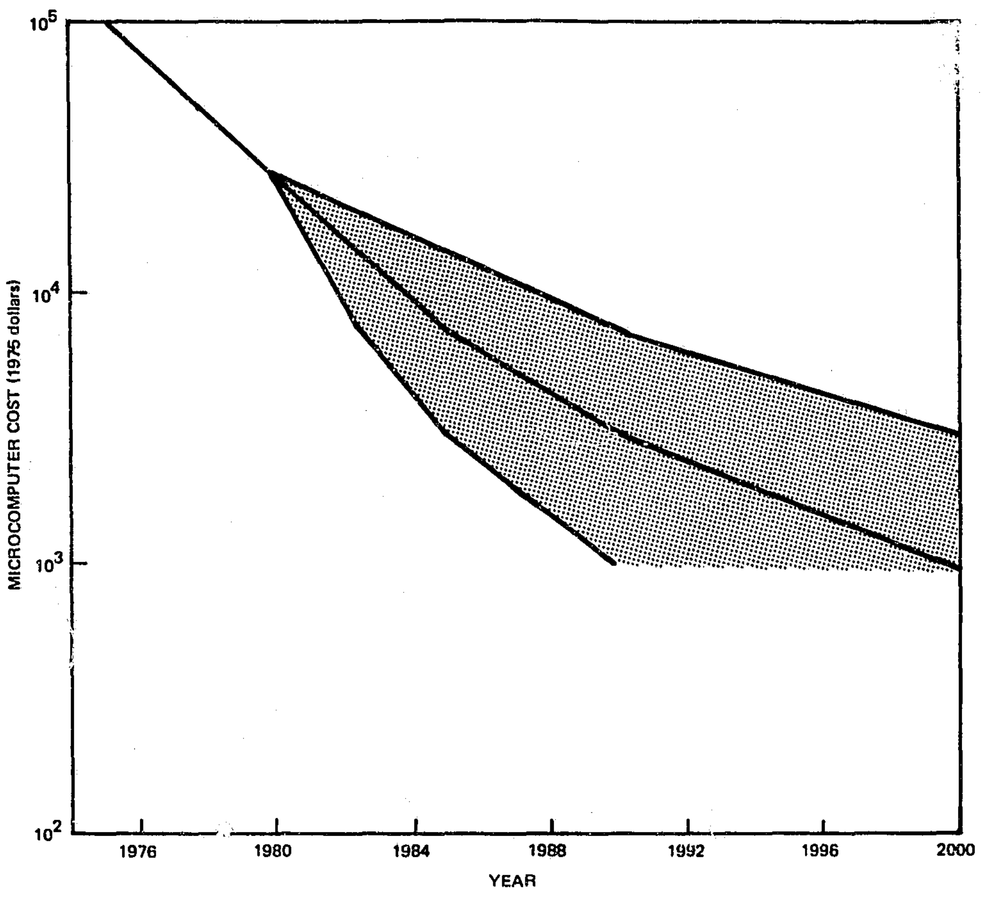


Figure 2-6. Forecasts of microcomputer cost.

toward lower cost and greater complexity are well illustrated by Figures 2-7 and 2-8. In Figure 2-7, the forecast of cost per packaged logic gate is given. In Figure 2-8 (taken from Reference 8), the increased complexity of more advanced flight-control equipment is shown when analog technology is used. The significant thing to note in this figure is that the tremendous increase in complexity is due to the redundancy requirements and not the control functions themselves. This reference assumes that the electronics needed to perform the advanced control functions would be only about three times as complex as a single conventional channel. However, the total complexity would be 25 to 60 times a single channel to achieve the dual fail-operative capability that would be required for flight-critical active control

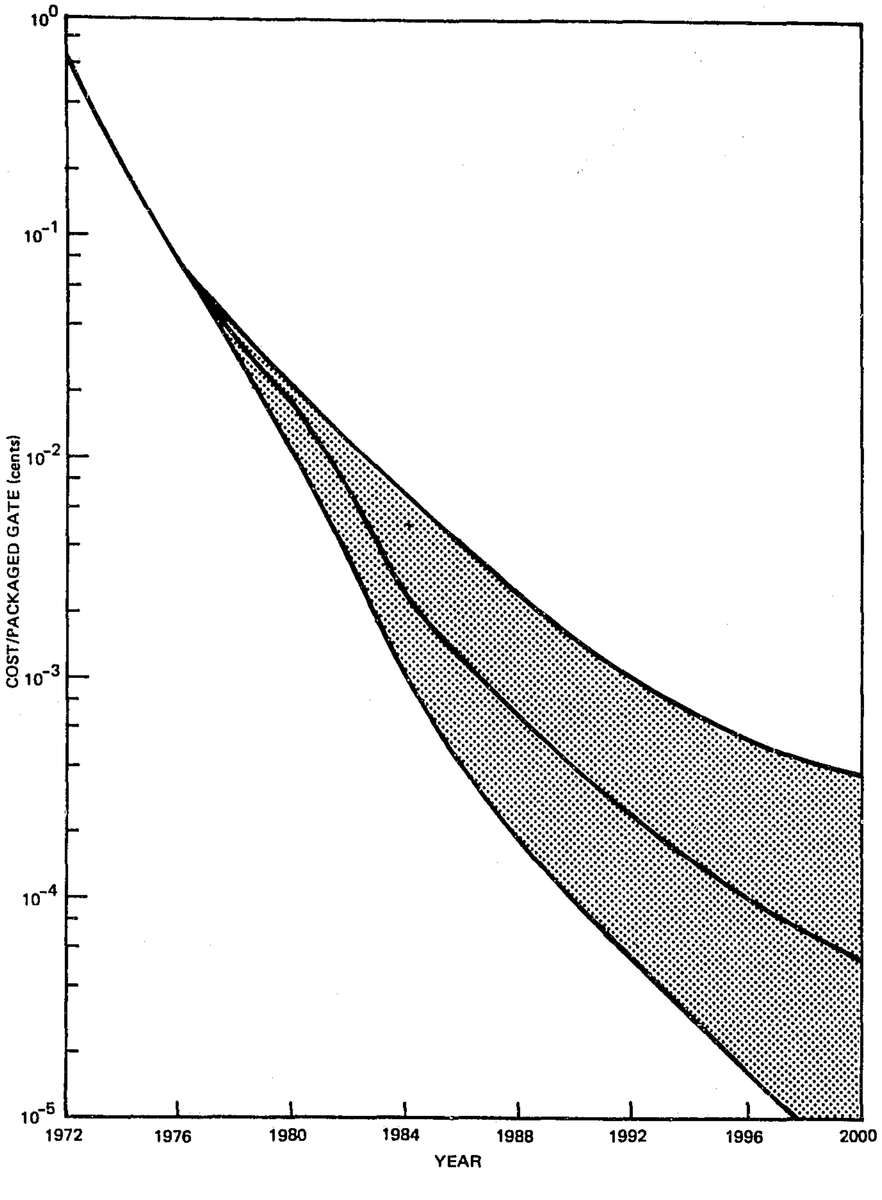


Figure 2-7. Forecast of cost per packaged gate.

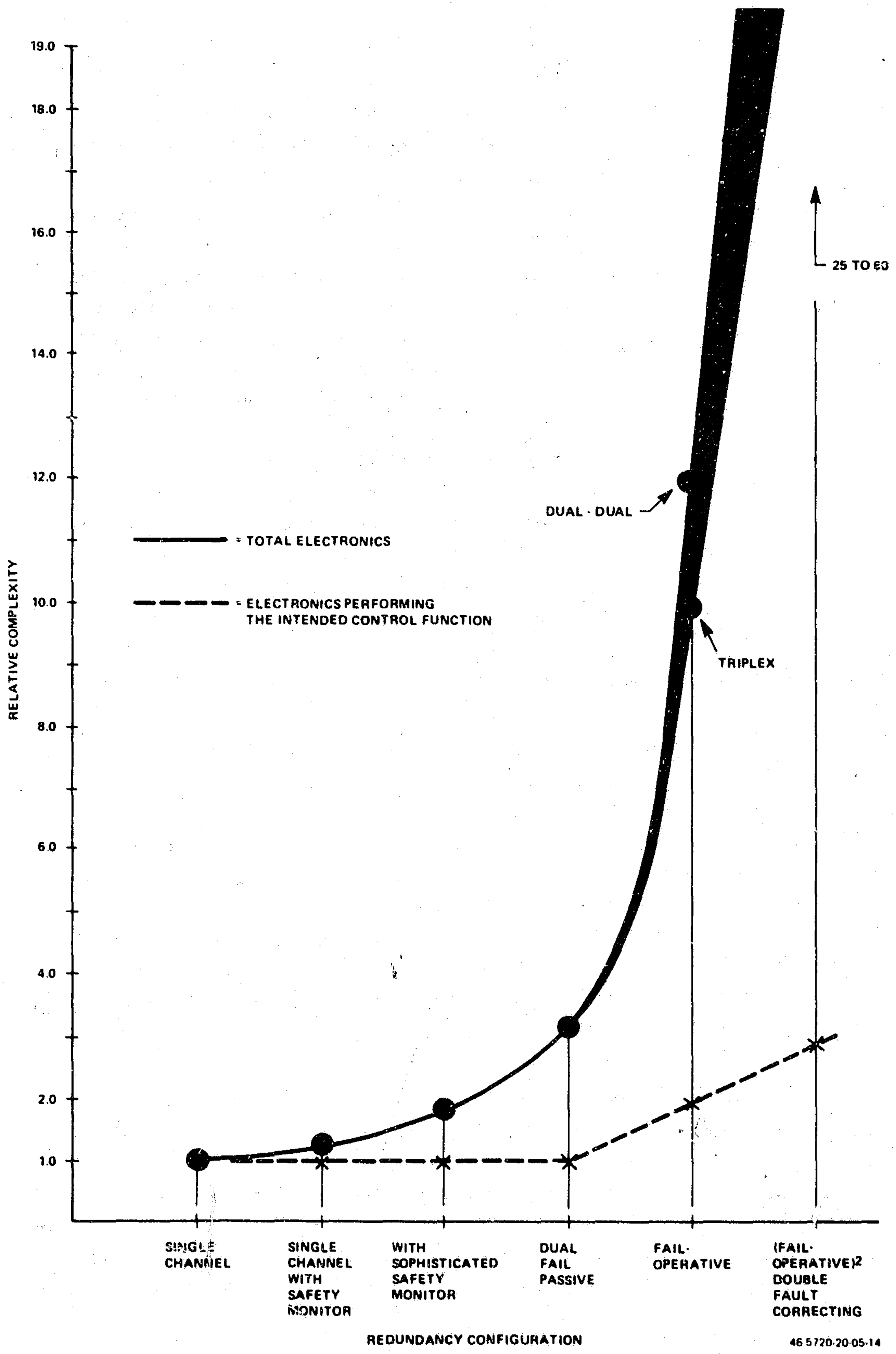


Figure 2-8. Forecast of increased complexity for future flight-control systems.

functions. It can thus be seen that this tremendous increase in complexity would not be feasible without the simplifications and reduced cost provided by digital technology.

The costs for this great increase in complexity are not completely compensated for by the reduced cost of the hardware itself. The engineering cost to develop these complex systems will be significant. The hardware cost for a digital system can become low because of the high production rates of identical digital chips. However, the thing that makes these identical chips perform the complex functions is software programs. The high cost of designing a complex system has not gone away, it has just moved from hardware to software. To illustrate this shift, the cost of a memory can be compared with the cost of the program in that memory with a medium-complexity program of 32,000 instructions. Figure 2-4 predicts that memory costs are expected to approach 0.1 cent per bit, giving a hardware cost of $500 for 32,000 16-bit instructions. Figure 2-9 gives the rising cost of programming. The cost for coding alone for a 32,000-instruction program would be at least $250,000. If a total of 500 units is produced, which is not unusually small for commercial avionic equipment, the amortized cost of the program equals the hardware costs.

In summary, it can be said that the expansion of flight-control functions and the explosion of digital technology are going hand-in-hand to cause significant changes in flight-control systems. On the one hand, the expansion of functions is requiring more complex electronic systems. On the other hand, the development of more complex and less expensive equipment is encouraging the use of this new equipment to perform functions that had not previously been considered possible. Section 2.3.2 outlines some of the ways digital technology is likely to impact flight-control systems, particularly from the viewpoint of validation and certification.

2.3.2 Impact of Digital Technology

The introduction of digital technology can impact flight-control systems in two ways: the hardware used to perform a traditional function can change, and new technology can encourage the introduction of new functions. In the second case, the impact on the validation and certification process is more related to the new function being performed than to the technology being used to perform the function. The new technology

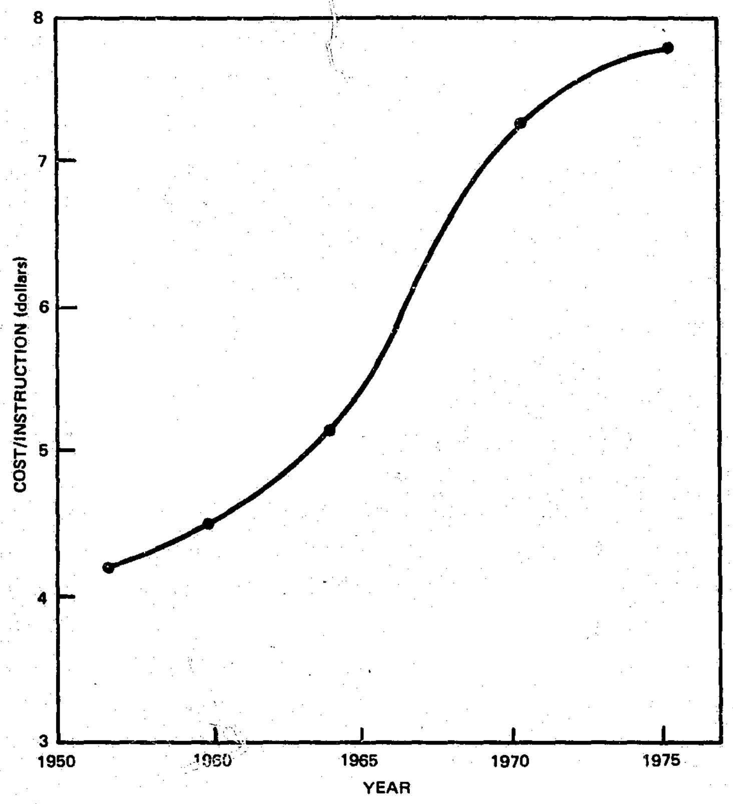


Figure 2-9. Average programming cost per instruction.

has only a second-order effect in that it makes the new function practical. Some of the implications of these new functions have already been discussed. In the first case, a change to digital technology to perform a traditional function may also have little impact on the certification process unless the change in technology changes the basic organization of the system.

2.3.3 Digital Technology Within the Existing Organizational Structure

If a particular functional box such as an autopilot channel is mechanized using digital technology instead of analog, there may be little change in the certification process. The basic validation tasks are:

(1) To assure that, given the necessary inputs, the outputs from the unit have the required performance to execute the function.

(2) To analyze the failure modes of the unit to assure that the probability of failure for that function meets the requirements.

It is likely that the technology used will have little influence on the performance testing; the failure analysis will be more influenced by the technology. There will be different types of failure modes to be considered. There has been concern that lightning and other electromagnetic interference will have different effects on digital technology than they did on analog. There has also been concern for software validation. However, for a single functional unit such as an autopilot channel, the software would likely be implemented in a fixed memory and would be analogous to the circuit design of an analog system. In fact, current analog autopilots contain much dedicated digital logic to implement much of the mode-switching and failure-monitoring functions. Microprocessors are now so inexpensive that if a design engineer were asked to implement the same functions that exist in the current generation of autopilots, he would probably use microprocessor chips with little thought that anyone would care. Although digital technology must be well understood, particularly for fault analysis, it may have no more impact on the basic certification process than other changes in technology, for example going from ac to dc autopilots. Thus, little change is expected in certification if only the technology inside a box is changed with little change in the function or organization of the boxes, and only noncritical systems are considered.

It could thus be argued that digital technology itself might have little effect on the certification process. If a new function is performed, the certification of that new function may not be influenced by what technology performs the functions. On the other hand, if a unit continues to perform a traditional function with only the change in the technology inside the box, again the certification may be little influenced by the technology. However, there is another potential effect of digital technology that could have significant influence on the entire certification process. This effect is the influence digital technology might have on the way avionics systems are organized.

2.3.4 The Development of a New Digital Structure: The Contrast Between Traditional Structure and Digital Structure

As digital technology is introduced, the natural tendency is to continue to maintain the same basic structure for the avionics system.

The traditional avionics system structure is based on a collection of functional units. There has been a natural and automatic assumption that a function is equivalent to a box. Thus, the design, manufacturing, maintenance, and certification of a function is equated with performing these functions on the box that executes that function. The total functional requirements for the aircraft are thus performed by a collection of boxes each providing the required functions. This is the natural consequence of electromechanical or analog technology. However, this is not necessarily the optimum organization when digital processors are used. The ideal objective is to design the entire avionics system to perform all required functions in the most effective way and also to meet all other requirements for reliability, maintainability, etc.

The possibility for a different organization results from the basic character of a digital processor. The heart of a processor is a group of circuits, which sequentially execute a set of general arithmetic and logical instructions. There is a memory to store the set of instructions (software program) necessary to perform the function. There must be interface circuits to bring in the required information and convert it to the digital form necessary for use by the particular processor. There must also be circuits to convert the results of the digital operations to the form necessary to implement the function. There is also memory to store the input, output, and intermediate data until it is used by the sequential logic circuits. This basic structure of arithmetic logic, memory, input interface, and output interface is presently independent of the function to be performed. This basic structure in block diagram form is shown in Figure 2-10.

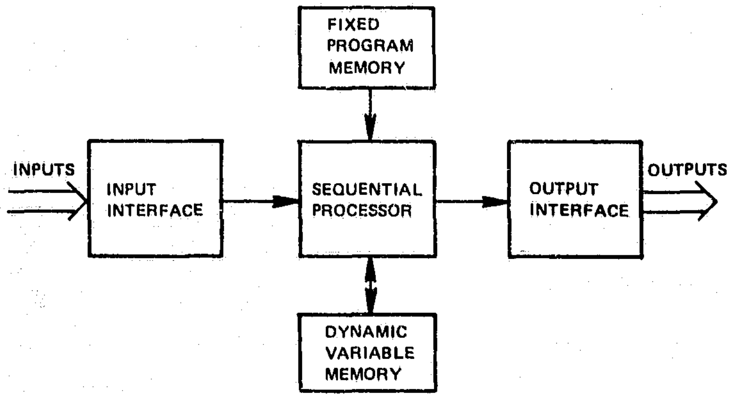


Figure 2-10. Basic structure of a digital system.

A basic difference between systems that use a digital processor and those that do not can be explained as follows. In a system without a processor, circuits are dedicated full time to the performance of a particular function. In systems with a processor, the same general-purpose logic circuits sequentially perform a set of basic operations, which together perform many functions.

2.3.5 Consequences of the Digital Structure

One of the consequences of this basic difference in structure is that in many cases it may be relatively easy to implement more than one function in one processor. Some of the implications of multifunction digital systems can be illustrated by an example of a system where the heading, attitude, air data, and basic autopilot functions are combined. This system is shown in simple block diagram form in Figure 2-11. The choice of which functions to combine is completely arbitrary and is made only to illustrate the idea. However, a system like this could be easily assembled with today's technology. Although the implementation of this system can be rather straightforward, there are aspects that would make it difficult to handle within traditional procedures. Some of the characteristics of this system will be described to illustrate the potential difficulties.

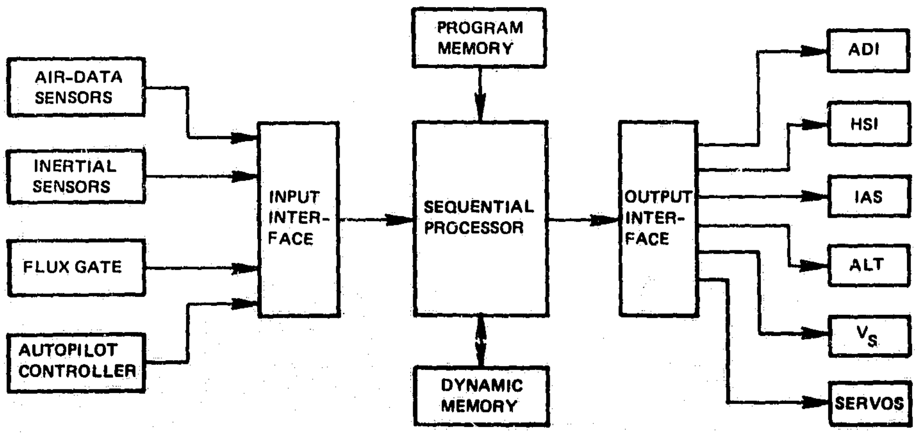


Figure 2-11. Basic diagram of a multifunction avionics system.

The outputs of the sensor units will be direct measurements of the individual sensors and will not be in a form that can be used directly or in the conventional form covered by the applicable regulations. For example, the output of the air-data sensors would be dynamic pressure, static pressure, and temperature. Air speed, altitude, or Mach number would be produced by a program stored in memory and executed by the processor. The values of altitude and air speed would be stored as digital words in the dynamic memory, converted periodically to the appropriate form, and sent to the display device.

In a similar way, the inertial sensors will produce raw data, not the conventional operational parameters. It is likely in a new system that the inertial sensors will be hard mounted or "strapdown" to the aircraft structure. Three single-degree-of-freedom gyros and three accelerometers would be used, and their outputs combined by the processor with the data from the flux gate to produce attitude and heading.

The autopilot function would be performed almost entirely within the processor. Autopilot commands would be received from a control panel. The reference air data, attitude, and heading would already be in the computer. The computed commands would be converted to the proper form and sent to the control actuators.

To obtain the most efficient design, it may be desirable to mix data from different sensors. For example, vertical-speed output may be a combination of inertial data and air data to give a more instantaneous vertical speed.

If all of these elements are combined into one line-replaceable unit (LRU), it may be possible just to combine all of the requirements for all the different functions and apply it to this new unit. However, this LRU may be too large or other factors are likely to make it undesirable to put all of these dissimilar parts in one box. It is possible to assemble all necessary LRUs, consider them a system, and apply the requirements to the whole system. However, this process will become almost impossible if the different LRUs are built by different manufacturers. The customer may not want to obtain inertial sensors from the same people from whom he obtains air-data sensors. In this case, it will be necessary to develop criteria and standards that can be applied to the parts such that when the parts are put together the whole system meets the requirement.

It can thus be seen that digital technology makes new organization structures possible, which can have a significant effect on how individual units are specified and certified. It is not yet clear what the most effective organizations will be. Some of the factors are discussed in the following subsection.

2.3.6 Organization of the Total System Using Digital Technology

One important question is how many different functions should be implemented in one processor. The extreme would be to perform all functions in one computer. This solution could be the most efficient from the hardware point of view, but would have many disadvantages from the reliability and maintainability point of view. This type of system seems to be the one envisioned by some in the commercial airline avionics industry when "integrated" digital avionics are proposed. They seem to envision a huge box with at least a thousand-pin connector with a mean time to failure almost as long as one flight. This box would have to be removed almost every flight, would be very difficult to troubleshoot, and would have to be tested for every function it performed before it was replaced in an aircraft. This image does not make integrated digital systems very attractive to the commercial community.

In spite of obvious disadvantages of extreme centralization, the equally obvious advantages of combining some functions into the same box is also clear. In fact, very practical systems are now being proposed for new aircraft using multifunction units. These proposed systems use several computers and group functions within computers using such criteria as flight critical vs. nonflight critical, dispatch vs. nondispatch, and the natural interrelationships between functions. Certification requirements also influence the choice of what to combine.

An alternative organizational structure that might be even more efficient would take advantage of the natural organizational structure of a processor and create the most effective LRU size by having separate processor modules, memory modules, input interface modules, output interface modules, sensor modules, etc. A standard digital-data-interchange system would be established, and the organization of the system would allow multiple processors to be used to provide the necessary capacity, failure protection, and dispatch reliability. The total set of functional requirements would thus be met by the total set of basic modules in the most efficient way.

This kind of avionic system organization will, however, require new thinking and new techniques to provide the validation necessary to assure that all the different regulatory requirements are met for all the different functions. With this organization, no one unit will be primarily responsible for performing a particular function and most units will be involved in more than one function. For example, the inputs for a flight-control function may come in through two or more interface units depending on the type of signal. One interface unit may be involved with digital signals, while another would be involved with analog signals. Also, the same interface units will probably bring in other signals used by other functions besides flight control. For example, an analog-data-acquisition unit can accommodate additional signals very efficiently by multiplexing these signals into the same signal conditioners and analog-to-digital converters. One or more different units may be used to provide the necessary outputs for this flight-control function. The algorithms that produce the functions would be performed by a central complex of processing units, which would also be included in almost every other function.

This type of organization would also allow the most effective utilization of sensor signals. For example, in a conventionally organized system an inertial-sensor unit would be required to produce roll, pitch, heading, and possibly ground speed. The ground speed would be used in a wind-shear-detection function. With this conventional organization, the inertial sensors would have to be of sufficiently high quality to meet the requirements for each of these parameters. In a more integrated modular-system structure, the basic measurements of the individual sensors would be interfaced directly into the processing complex. Within the processors, all available sensor information would be combined in the most efficient way to provide the basic information on the attitude, velocity, and position of the aircraft. This arrangement makes it possible to use radio navigation or air data to improve and calibrate the inertial sensors. It is thus possible to meet the ultimate accuracy requirements for a particular parameter and use less expensive basic sensors.

The operation of the system can become even more complex when potential failure modes are considered. Any flight-critical function is likely to be fail operational; that is, it will continue to function after any one failure. This requirement means that for every unit involved in a fail-operational function, there must be an alternate to provide the same operation. In most cases there will be duplicate units. However, in other cases the same ultimate function may be provided by different types of units. For example, a combination of inertial and air data might replace radio data for short-term navigation.

2.3.7 Implications of Digital Technology on the Certification Process

The characteristics described thus far—those that are made potentially very desirable by digital technology—do have obvious implications on the certification process. One basic problem is to educate everyone involved in the design, building, operation, maintenance, and certification about the new concepts involved and the implications of these new concepts. Much of the current body of experience and procedures is built around a functional unit performing a basic flight function. Most Technical Standard Orders (TSOs) envision a unit performing a particular function. These TSOs give the testing required on the unit to validate its performance in supplying the desired flight function. It will be more difficult to define the test when these functions are performed by a collection of units. The difficulty will be increased when these units are involved in many other functions that must meet other TSOs, and particularly when the different units are made by different manufacturers.

The potential new approaches to avionics system organization may require new validation and certification procedures or at least a new way of thinking. The processor may be involved in almost every function. However, instead of being tested in combination with many other units to see if it properly performs the flight-operational function, it would be tested to assure that it performs properly its basic operations of add, subtract, multiply, and divide along with the associated logical functions. At some time by analysis or system testing, it will be necessary to show that when each unit performs its basic function properly and when these units are combined, the total system will meet all requirements. It may be necessary to develop new techniques to validate the individual units of a system in order for the certification requirements not to place unnecessary constraints on the design of the system. It will be necessary to make the validation process clear early in the development of new aircraft so that these requirements can be fully understood by the system designers and can be reasonably accounted for in the initial design.

2.4 Evolving System Organization

2.4.1 Basic Organizational Considerations

Functions that are performed by flight-control systems and the technology that is available to perform these functions have been discussed. The question to be considered now is: How can this technology be organized into a system that performs the desired functions?

2.4.1.1 Reliability Considerations

One of the most important factors influencing the configuration of the system is the required reliability for the function. All early and many present applications of electronic flight-control equipment perform auxiliary or supplemental functions. These functions are not flight critical; a failure can usually be easily recognized and the function taken over by the pilot. It is only important that they not fail "hard-over" or in some other way create a hazardous condition. In many cases, these functions are only a convenience, are not essential, and thus are not required to dispatch the aircraft. In this situation, the design organization is usually straightforward. A single channel will be adequate for most applications. When a function is not too complex and can be accomplished in one major electronic LRU with associated sensors and actuators, a probability of failure per flight hour of approximately 10^{-3} to 10^{-4} can usually be achieved. This reliability is sufficient in these cases.

When a high functional reliability is required, a more complex system organization is needed. In the extreme case where a catastrophic event results from a complete system failure, the probability of such a failure must be extremely improbable, i.e., on the order of 10^{-9}. If it were possible to build a simple system with this level of reliability, the organization would not be affected by the increased reliability requirements. However, reliabilities of this magnitude are not presently practical in electronic equipment. The computer for the Apollo space vehicles was probably one of the most reliable single complex digital processors ever built. It has a demonstrated mean time between failure (MTBF) of 70,000 hours. The cost of achieving this reliability was very high. However, a system using this computer still would not meet the reliablity requirements for a flight-critical function in a commercial airplane. It is thus necessary to achieve the required reliability in some other way.

2.4.1.2 Redundant Channels

The only practical way of achieving the desired operational reliability is to use redundant channels. As discussed earlier, one channel of electronic flight-control equipment can be expected to have a probability of functional failure of 10^{-3} to 10^{-4} per flight hour. Thus, a minimum of two channels would be necessary if a probability of failure of 10^{-6} were required and at least three channels for 10^{-9}. More than the minimum are needed if comparisons are used to monitor for failures. The important question now is how to arrange these channels to assure that:

(1) A failed channel has no effect on the output.

(2) The failed channel can be identified.

(3) The properly operating channels are used.

(4) There is no common mode of failure.

(5) No single event that is not itself extremely improbable can cause all channels to fail.

The most critical problem is to assure that a failed channel does not affect the output. There are two major approaches to this problem: voting and self-monitoring.

2.4.1.3 Voting Systems

To obtain a fail-operational capability with the probability of functional failure down to 10^{-6} per hour using the voting method, three channels are used. A circuit that selects the middle value of three outputs is the usual method used to assure that a channel with a failed output is not used. The bad channel can be identified by comparing it with the result of this vote. The bad channel can then be removed and the two remaining channels continue to operate in a fail-passive mode by comparing with each other. If they disagree, they can both be disabled.

2.4.1.4 Self-Monitoring Systems

The other major approach is self-monitoring within the channel, which assures that failures can be detected and the channel can shut itself off. There are several ways this self-monitoring is accomplished. Special circuits called built-in test equipment (BITE) are used. These circuits can test the tolerance of power-supply levels and many other similar types of failures within the channel. It is difficult, however, to achieve more than 90 percent coverage of all possible faults. Another method of analog-system self-monitoring is comparison with an essentially identical model channel. This method is used when full coverage of all possible errors is necessary. This technique is basically very similar to voting between two channels. However, the redundant monitoring circuits are usually within the same LRU, and all circuits may not be duplicated. For example, in an autopilot, the circuits involved in the automatic landing mode would be duplicated, while the circuits only involved in cruise modes might not be duplicated.

When digital technology is used, self-test can be implemented more effectively and with less additional hardware. As was discussed earlier, systems employing digital processors are basically operated sequentially. That is, the same circuits are used to rapidly perform basic operations. The sum total of many basic operations produces the required functions. Since the same circuits are used repetitively, the proper operation of the circuits can be confirmed by performing test operations. For example, a test software program can be performed, which executes every instruction in the processor and compares the result with the correct answers. Using techniques like this, the probability of an undetected failure in the processor can be made extremely small. Many of the support circuits are also multiplexed. For example, analog signals may be brought in through multiplexed signal conditioners and analog-to-digital converters. A test signal can also be brought in, which will test the operation of all common circuits. Program memory can be effectively tested by simply using a software test program that adds all permanent memory locations and compares the resulting sum with a predetermined check-sum. If any bit is gained or lost the sum will not check. To be sure the computer does not just quit or become hung in an infinite loop, a "watchdog" timer can be used. This timer has to be reset by the software program each time it completes its critical computation loop. If the timer is not reset, it will flag the computer as failed.

2.4.2 Typical System Configurations

Systems have been assembled to represent both the voting and self-monitoring approaches, and various combinations in between. Brief descriptions are given of several different existing and proposed flight-control systems that illustrate several different system organizations. It is not the purpose of this report to give a complete description of

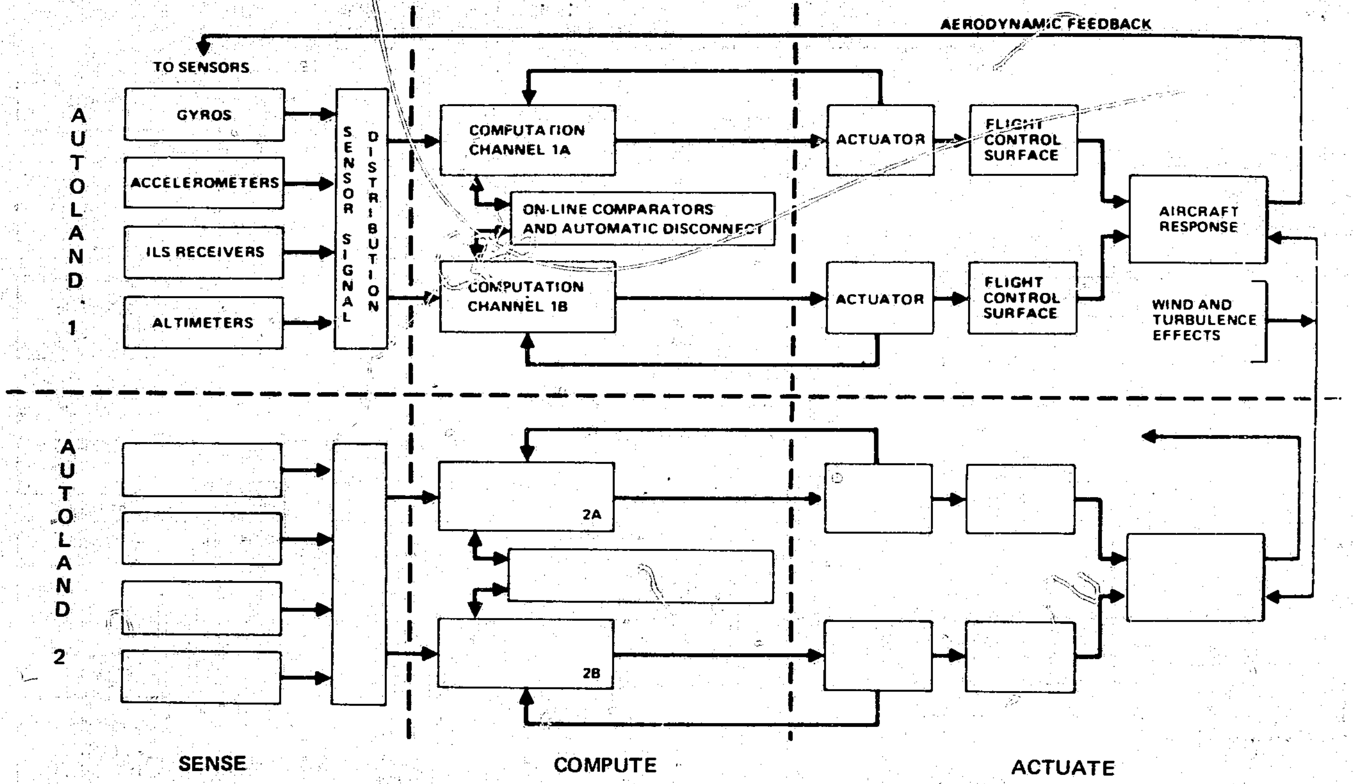


Figure 2-12. Basic configuration of DC-10 AFCS autoland modes.

ORIGINAL PAGE IS OF POOR QUALITY

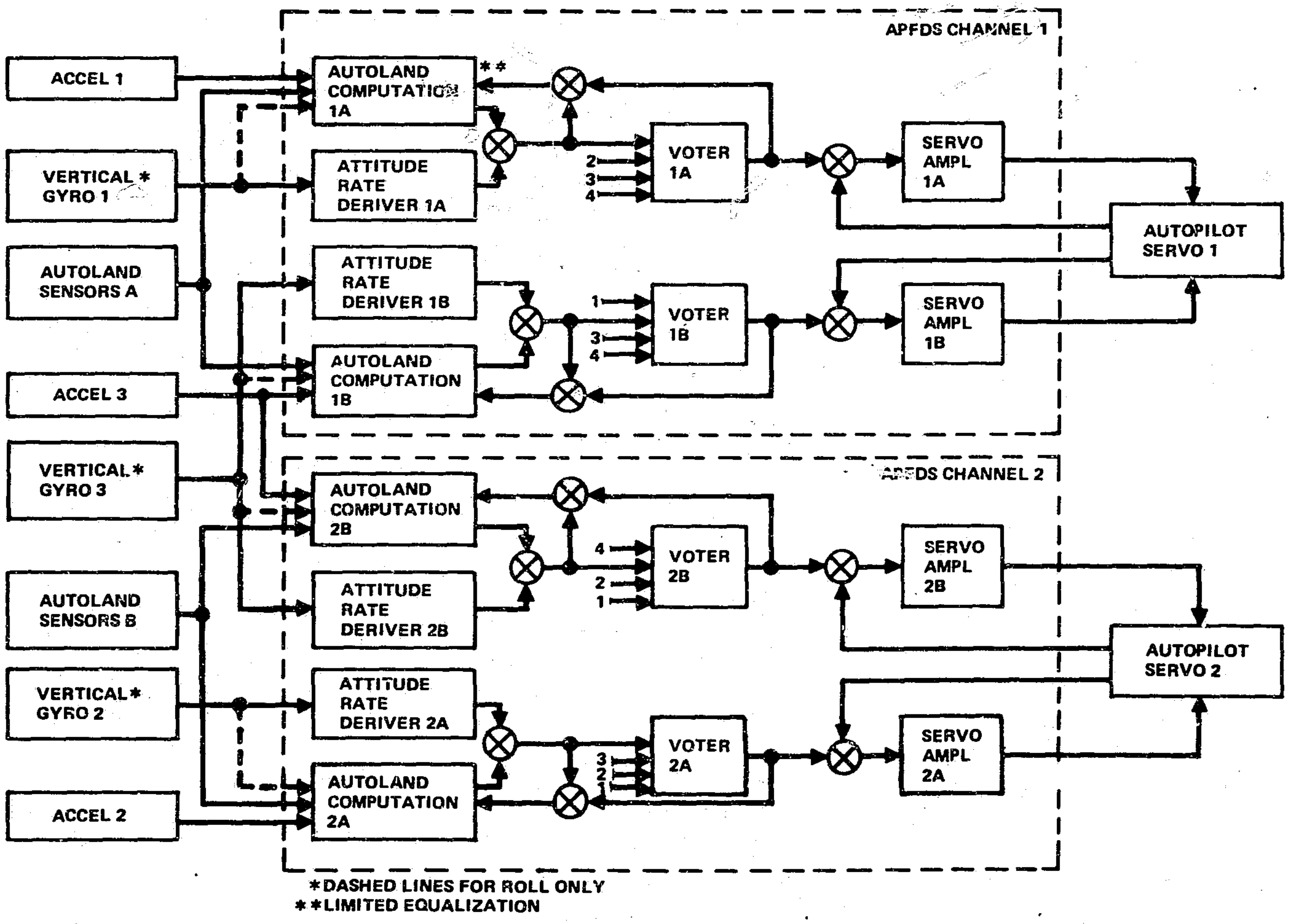


Figure 2-13. Basic configuration of L-1011 AFSC autoland modes.

of a dual-monitored digital flight-control system. A simplified block diagram of the system is shown in Figure 2-14. It is designed to be fail-safe for all failures, and fail-operational for failures in the computer and memory units. The self-tests within the computer and memory are considered to be sufficient to identify their own failures to a high degree of confidence. The self-test monitoring functions are related to those developed for the Swedish Viggen fighter. The Viggen uses a single fail-safe system, which achieves a probability of catastrophic failure of 10^{-6}.

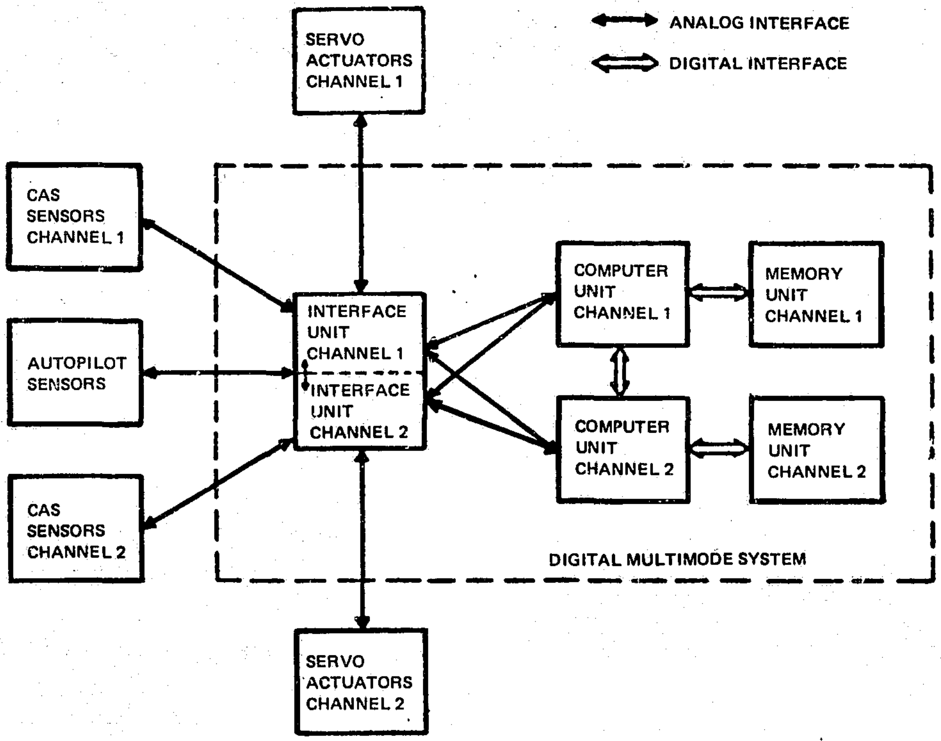


Figure 2-14. System block diagram for the Air Force DIGITAC A-7 system.

The function of the DIGITAC system in the A-7 is to perform stability augmentation and pilot-relief autopilot modes. The mechanical control linkages were not removed, and the stability augmentation is not flight critical. The aircraft can thus be safely flown after a complete system failure.

The flight tests were very successful in demonstrating the capability of digital flight-control system and in establishing a technology base for the development of future systems. Development problems uncovered by this program are contributing to future designs. One example is the problem of interaction between self-test routines. In one instance, a power-supply problem caused one computer to fail. An unforeseen timing situation in the self-test of the cross-computer data link caused the good computer to shut itself off. This problem was corrected. However, its existence shows that these kinds of interactions must be studied very carefully.

2.4.2.2 Triplex Configurations

The other major configuration used in current systems is triplex with voting. Brief descriptions are given of the B-747, Air Force YC-14 Advanced Medium STOL Transport, and NASA F-8 DFBW.

B-747—The fail-operational autoland option of the B-747 flight-control system is a triplex system using analog technology. This system uses three independent channels feeding three secondary actuators. With a single failure, the system continues to operate even if the failed channel is not disconnected.

YC-14 Digital Flight-Control System—The YC-14 system uses a triple-redundant set of electronics and multiple aerodynamic surfaces to achieve fail-operational/fail-safe performance.(12) A schematic of the system is given in Figure 2-15. A unique feature of this system is that the three computers are interconnected with a fiber-optic system. Thus, there is no electrical connection between the three computers. The system provides automatic signal selection, failure detection, failure isolation, failure warning, and failure isolation confirmation during flight-critical operations. The input signal selection guarantees that all computers will use the same numbers and thus produce identical outputs. The output is selected as the midvalue of the three values. The system continues to operate after the first failure by taking the average of the two remaining systems. When the two remaining systems disagree, they are both disabled and the aircraft is flown manually.

The YC-14 is intended to be a full-time system to enhance aircraft performance and to give it excellent flying qualities with low-workload conventional piloting techniques by compensating for the unusual STOL powered lift characteristics. The system is made fail-operational to give a low probability that the flight characteristics will change.

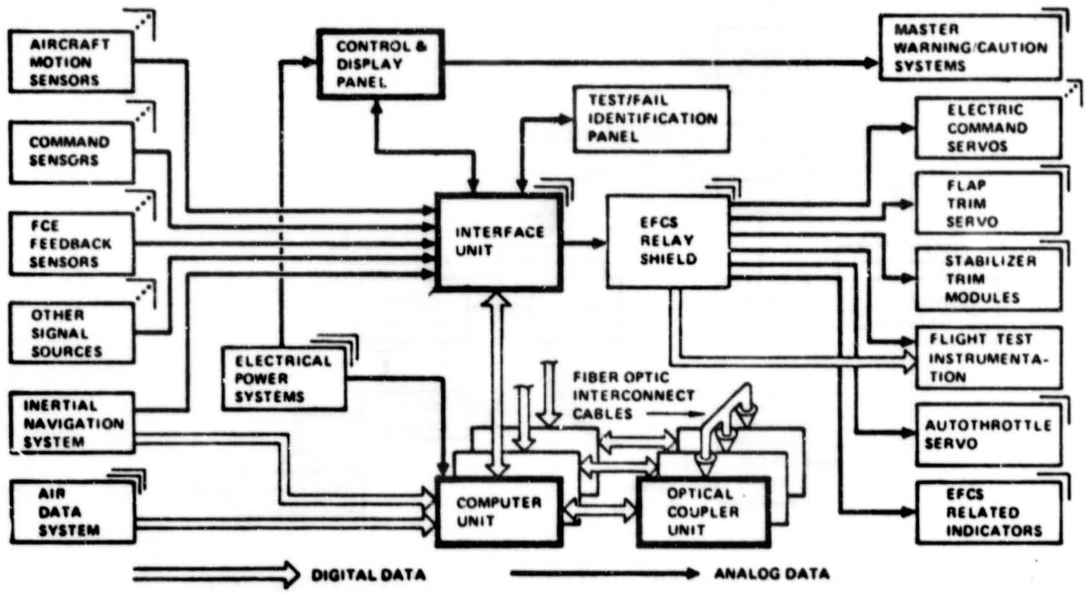


Figure 2-15. YC-14 electrical flight-control system schematic.

However, if there is a second failure, the aircraft can be flown manually with increased pilot workload and some limitations of the operational flight envelope.

NASA F-8 DFBW System—The basic configuration of the F-8 DFBW system is shown in Figure 2-16. This system also has three primary digital channels. A major distinction of the system is that the backup control system is also electronic. The mechanical linkage between the pilot's controls and the aerodynamic surfaces have been removed. The system is thus truly fly-by-wire. Another major distinction is that the primary system is fault tolerant with transient fault recovery capability.

The critical input sensors are triplex, and the data from each of the redundant sensors is supplied to all three computers. Identical signal-selection programs are performed in each computer. This signal selection identifies and removes the effects of failed sensors and produces identical input signals for each of the three computers. These identical inputs are used by the computers to produce three control-surface command outputs. The midvalue of the three commands is selected by three different servo-control-electronics channels. These three channels drive the three sections of triplex force-summed secondary

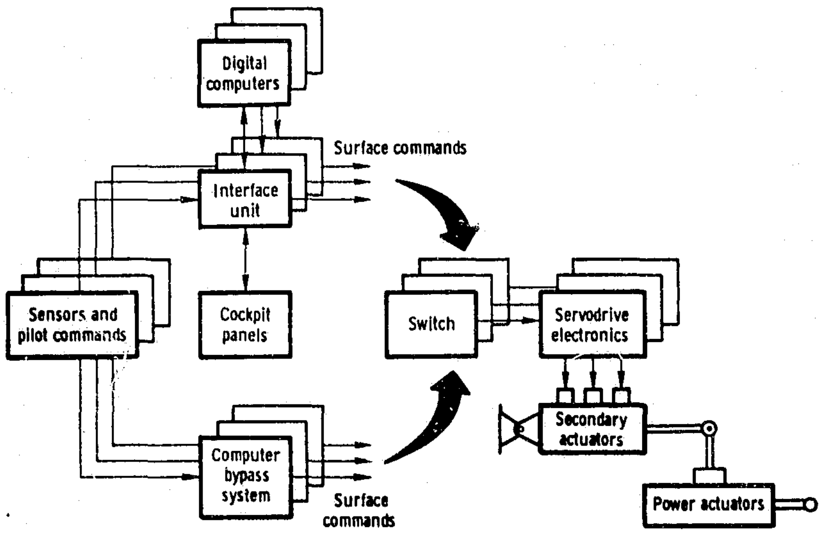


Figure 2-16. F-8 DFBW control system mechanization.

actuators. These secondary actuators replace the function of the original mechanical linkages, and command the primary power actuators. The selection logic in the analog drive channels will identify and eliminate a failed digital channel if its command signals deviate significantly from the other two. The system will continue operating using the two remaining good channels. Many of the faults detected are transient and the system has the capability of restarting the failed channel and returning to full three-channel operation. If the fault is permanent so that only two channels remain and they do not agree, the system reverts to a triplex direct analog coupling between the pilot commands and the servo drives. This computer bypass system does not provide stability augmentation, command augmentation, or autopilot functions. However, this system allows the aircraft to be safely flown manually. The triple digital system backed up by a triple analog system gives a probability of failure of the fly-by-wire function due to identified random failure sources that is lower than 10^{-9}.

2.4.2.3 Emerging Configurations

It can be seen from the previous examples that electronic flight-control systems are progressing from functions that are optional and, if used, are only flight critical for a very short time to those that are flight critical all the time. Future systems employing more flight-critical CCV modes or fly-by-wire require configurations with progressively lower probabilities of complete system failure. Two such systems will be briefly discussed, which illustrate possible future configurations. One is now flying on the Space Shuttle and the other is a concept which indicates future trends.

Shuttle Orbiter Computer System—Flight control of the Shuttle Orbiter is the responsibility of a highly complex integrated digital computer system. A simplified diagram of the system is shown in Figure 2-17. Five identical computers are used, which are connected to each other and to peripheral devices by a system of redundant serial data buses. There are seven groups of buses. The number in each group is shown on the figure. Only three of these groups are involved in the flight-critical functions that could be compared with a system used on a conventional aircraft. The other buses and much of the peripheral equipment is involved with functions such as telemetry, launch operations, and payload systems.

During approach and landing, the primary flight-control function is performed by identical programs in four of the five computers. Each of the redundant-sensor subsystems is connected to a different bus, which is controlled by one of the computers. However, since all computers are connected to all buses, the computers that are not commanding a particular bus can listen to the data being requested. In this way, identical data is available to all computers. Identical flight-control commands can thus be computed by each processor. Each channel of the voting actuators is connected to a different bus. The computer responsible for that bus sends the commands to that actuator channel. Each computer monitors the command data sent out by each of the other computers. In order to prevent data skew of either inputs or outputs, a synchronization program is employed, which uses discrete signals sent back and forth between computers on the intercommunication bus.

In order to guard against the possibility of a complete failure due to some generic fault in the system software, the fifth computer performs a backup function. This computer uses a different program developed by a different organization from the programs in the primary computers. In this way the Shuttle system has been able to place reliance on a completely digital fly-by-wire system with no mechanical or analog backup means to control the vehicle. It thus represents the most flight-critical, pure digital control system in operation to date.

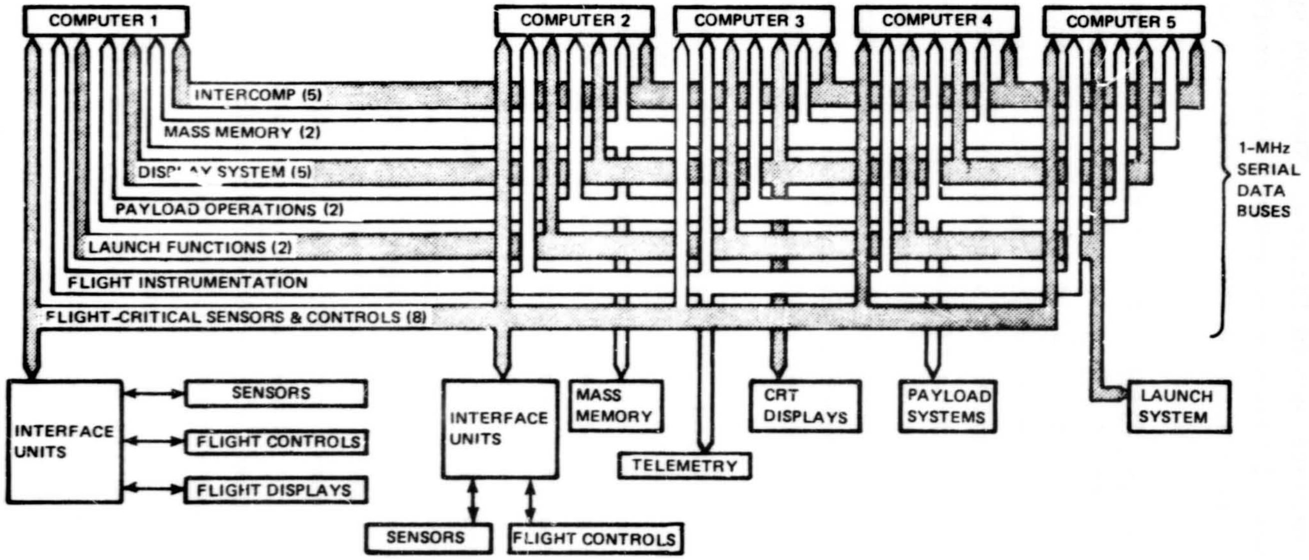


Figure 2-17. Shuttle orbiter computer-system block diagram.

Onboard Survivable Integrated Redundant and Interface System—As the desire for more capable and more reliable flight-control systems increases, and as the technology that make these systems possible becomes available and less expensive, more effective and more complex system structures are likely to emerge. One example of the more advanced systems being conceived and developed is called OSIRIS—an Onboard Survivable Integrated Redundant Information System.(13) This system will be described briefly to give a glimpse of what kind of systems might be coming in the future. Elements of this system include a fault-tolerant multiprocessor,(14) a damage-and-fault-tolerant digital data information network,(15) distributed local processors, and operational software for fault detection, identification, and recovery, for the entire system.

A diagram of the fault-tolerant multiprocessor (FTMP) is shown in Figure 2-18. The FTMP is made up of redundant sets of processors, memory units, and input/output interface units. These units are connected by redundant data bases. All processing is performed by triads of units. Three processors, three memories, and three buses are configured into a processing set, which performs identical operations. Using this technique,

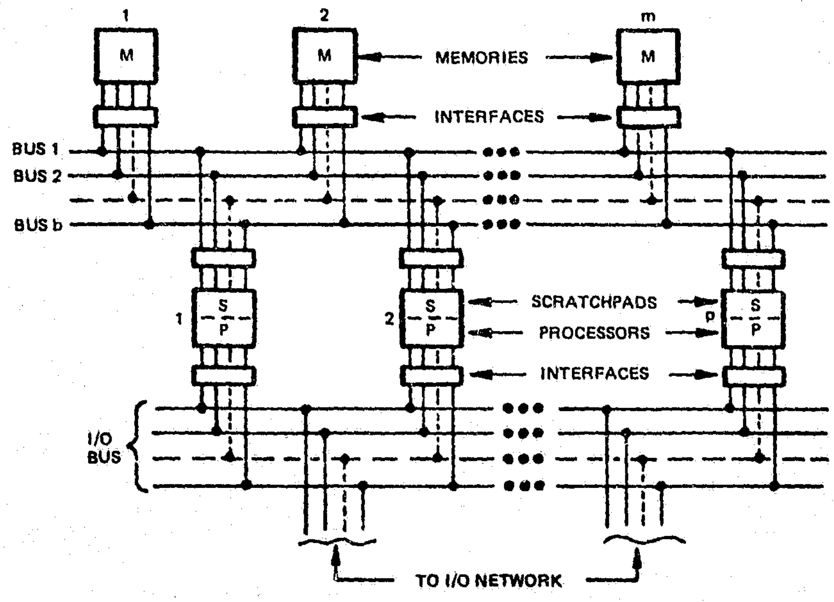


Figure 2-18. Fault-tolerant multiprocessor.

any failure can be immediately detected and the failed unit removed. The total processing requirement is met by combining several sets of these processing triads into a multiprocessor system.

To illustrate the operation of the FTMP, assume that the total processing needs for the central controllers of an avionics system require three processor sets. This example system would be constructed with ten processors. Nine processors would be configured into a three-set multiprocessor. The tenth would be a spare. The processors making up the triads would be continually reconfigured so that any failure of the spare processor would be quickly detected. If one processor failed, it would be replaced by the spare and the system would continue to operate normally. When the second processor failed, many peripheral functions would be lost, but two new spares would be created making it extremely improbable that the system would ever reduce to only one triad. With only two sets operating, the system would not be able to perform long-term support functions, such as maintenance monitoring. However, two sets would provide almost all operational capability needed for a particular flight. With only one set working, much of the operational capability would be lost, but the aircraft could be safely flown and landed. The operational envelope may have to be reduced and pilot workload would be higher. Needless to say, the probability that there would not be enough processors to support one triad is extremely small. This multiprocessing design allows relatively low capability processors to be used, making possible efficient use of spare units and graceful degradation of capability. It is likely that future high-end microprocessors will be able to perform this task.

The FTMP forms the central controller to manage the entire OSIRIS system, but the FTMP is not intended to be a central computer performing all processing tasks in the system. The total system is built up of a hierarchical set of processors with tasks done at as low a level as possible to allow for the maximum practical functional independence. There would be local processors associated with particular sensors, actuators, and displays, and middle level processors associated with different functional groups such as navigation.

All of the elements of the system would be connected by a network of communication links between each of the nodes in the system. A very simple eight-node network is shown in Figure 2-19. This network is so designed that any node can communicate with any other node if any link is lost or if any node fails.

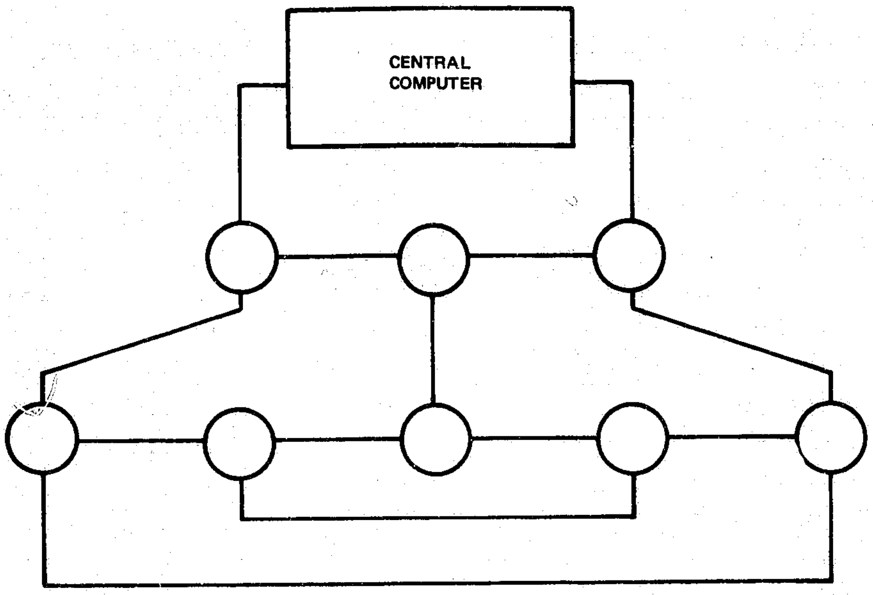


Figure 2-19. Simple eight-node network.

2.4.3 Implications of Different Configurations

Different system configurations will have different implications for the validation and certification process. Some systems will be relatively straightforward and easy to understand so that the magnitude of the analysis task will not be unusually large. Other systems, by the nature of their configurations and design philosophies, will be much more difficult to analyze. This situation presents a challenge and a responsibility to the system designer and the certification authority. The system designer must recognize that validation and certification are real system requirements. The system should be designed to make these steps as easy as possible within the constraints of other competing system requirements. On the other hand, the certifying authority must have the capability and willingness to understand and perform the extra effort necessary to certify a complex system if that complex system can be shown to be more cost effective.

Two of the currently most common configurations—dual-monitored and triplex-voting—could have significantly different validation requirements. For example, a triplex-voting system can be more straightforward. Since fail-operational capability is usually obtained by a majority vote, it may not be critical to identify and analyze the nature of all possible failure modes within each channel. No matter what the failure is, if it has a significant effect on the output, the effect will be detected and eliminated by the voting process, and the failed channel can be identified. It is important in this configuration to assure that there are no common mode failures and that the voting process itself is not the source of a critical failure.

On the other hand, the dual-monitored system may require considerably more analysis of failure modes and effects. It is necessary to assure that all failure modes have been identified and that all with potentially critical effects are either extremely improbable or that the system is capable of detecting the fault and eliminating its effects. The analysis required to gain the necessary assurance that all failure modes have been properly accounted for may be much greater than the equivalent triplex system to achieve the same confidence of fault tolerance. The reward, however, would be one less channel of hardware. On the other hand, the complications necessary to assure full monitoring in a dual-monitored system may be more expensive than the third channel. The design engineer and certifying engineer should understand each other's responsibility enough that the technically most efficient design is obtained without any artificial constraints.

SECTION 3

NASA ADVANCED FLIGHT-CONTROL PROGRAM

The F-8 DFBW flight experiment is a research flight-test program being carried out within NASA to provide the technology required for implementation of Advanced Digital Fly-by-Wire control systems in future aircraft, permitting greater operational capability and increased performance. The program is being carried out using an F-8 test aircraft.

3.1 Brief History of the Program

The program has been divided into two major phases. The first phase, which began in 1971 and concluded in 1973, successfully demonstrated the feasiblity of using DFBW systems for the primary control of aircraft. This was accomplished by flight testing a single-channel DFBW system in the F-8 test aircraft. A surplus Apollo guidance and navigation computer was used for the primary flight-control system, and the basic F-8 mechanical system was completely removed. Forty-two flights were accomplished during this phase by six evaluation pilots, and a total flight time of 58 hours was accumulated. Historically, this was the first recorded flight of an aircraft using as its primary means of flight control a Digital Fly-by-Wire system, with no mechanical backup means.

The second phase of the program began in 1973 and is currently underway. The overall Phase II program objective is to establish a data base which can be used in the design and development of practical DFBW systems for future aircraft. To accomplish this objective, the simplex Phase I system has been replaced with a triplex multichannel DFBW system, which uses fully programmable state-of-the-art digital processors for primary flight control. The first flight with the Phase II system occurred in August 1976, and since that time, over 50 flights have been successfully accomplished. The flight-test program has been successful both in demonstrating a practical DFBW design concept that works, and in developing required operational procedures. This is also the only airplane primary flight-control DFBW system currently in operation that does not employ a means for mechanical reversion in the event of failure.

One of the tasks accomplished during Phase II has been the implementation and flight-test verification of certain basic concepts to be utilized in the Space Shuttle control system, which is to be the first practical application of DFBW technology. Use of the F-8 as a test bed resulted in early availability of redundancy management flight-test experience, as well as the verification of other concepts of particular concern in support of the Space Shuttle Orbiter Development Schedule. It is a tribute to the flexibility of DFBW systems in general to have accomplished a multitude of tasks relatively easily in terms of cost and schedule constraints.

The F-8 DFBW program is managed out of the Electronics Division of the Office of Aeronautics and Space Technology (OAST) at NASA Headquarters. The project office resides at the Dryden Flight Research Center (DFRC), which has functioned as the lead center during the entire program. Other NASA centers have been jointly involved. The Langley Research Center (LARC) has been responsible for development of certain advanced control law concepts for flight-test evaluation of the Phase II system. The Johnson Space Center (JSC) has been jointly responsible for coordinating all shuttle-related flight tests.

3.2 Brief Summary of the Flight-Test Program

A summary of the flight-test program accomplishments to date is presented in Figure 3-1. During the 50 flights accomplished thus far, all control modes have been engaged and evaluated using the various control tasks that are listed. The various data generated during the accomplishment of these flights has contributed greatly to expanding the technology data base for the DFBW controls, and in accomplishing the F-8 DFBW program objectives. Of primary significance is the fact that at no time during ground or flight tests of the flight-qualified system has a total DFCS failure occurred requiring use of the analog bypass system for aircraft control.

The current flight-test schedule for the F-8 DFBW program is shown in Figure 3-2. This includes completion of the remotely augmented vehicle facility development during the current year (1978) and using this capability as a means to evaluate an advanced adaptive control law concept. Also included are low sampling rate investigations. An Analytical Redundancy

- NUMBER OF FLIGHTS 50
- TOTAL FLIGHT TIME . . . . 55 h
- MAXIMUM SPEED . . . . MACH 1.2
- MAXIMUM ALTITUDE . . . . 12,200 m
- MAXIMUM ACCELERATION 6g
- NUMBER OF PILOTS 4
- CONTROL MODES EVALUATED
  - DIRECT
  - STABILITY AUGMENTATION (SAS)
  - COMMAND AUGMENTATION (CAS)
  - AUTOPILOT:
    - MACH HOLD
    - ALTITUDE HOLD
    - HEADING HOLD
    - ATTITUDE HOLD
  - RIDE SMOOTHING
  - MANUEVER DRAG REDUCTION (MDR)
  - REMOTE AUGMENTATION (RAV)
  - SIDE-STICK CONTROLLER
  - ANGLE-OF-ATTACK LIMITER
- EVALUATION TASKS
  - ROUTINE FLIGHT
  - HANDLING QUALITIES INPUTS
  - FORMATION
  - TRACKING
  - MODERATE/SEVERE TURBULENCE
  - SIMULATED SHUTTLE LANDINGS
- FLIGHT-TEST VERIFICATION OF SHUTTLE RM SOFTWARE
- EVALUATION OF PILOT WORKLOAD DURING SHUTTLE LANDING MANUEVER
- ESTABLISHED HARDWARE AND SOFTWARE OPERATIONAL PROCEDURES FOR DFBW SYSTEMS
- FOUR IN-FLIGHT COMPUTER FAILURES – DEMONSTRATED VALIDITY OF FAILURE DETECTION AND RECOVERY ALGORITHMS
- NO TOTAL SYSTEM FAILURE REQUIRING USE OF ANALOG BYPASS SYSTEM.

Figure 3-1. Flight-test summary.

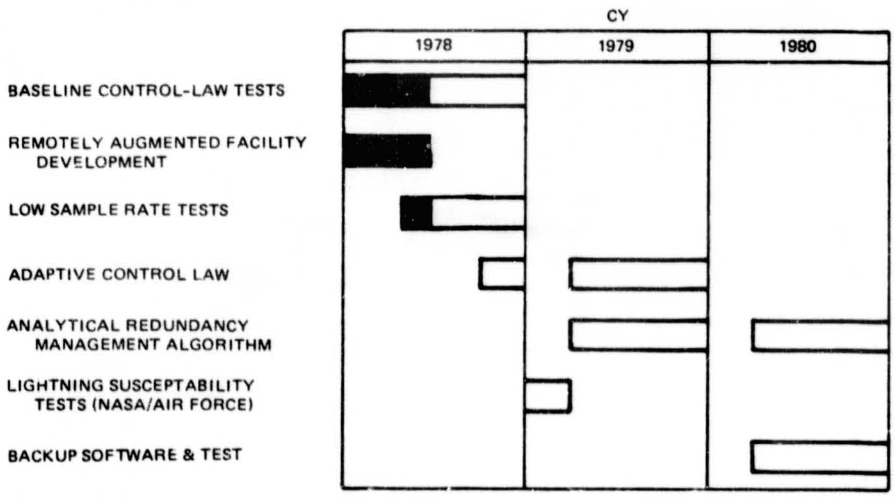


Figure 3-2. Current flight-test schedule.

Management (ARM) algorithm is to be evaluated in flight during 1979, in addition to assessing the susceptibility of the Phase II system to induced lightning transients. The lightning susceptibility tests will be carried out on the ground using special equipment and techniques developed by the Air Force EMI Hazards Division of the Flight Dynamics Laboratory (FDL). Special backup software algorithms are to be developed and tested during the 1980 time period, addressing the generic software failure problem associated with redundant digital systems.

SECTION 4

DIGITAL FLY-BY-WIRE PROGRAM EXPERIENCE

This section first describes the system requirements. These requirements include both those specific to this program and those more generic specifications that would be typical of an advanced primary flight-control system. The requirements that must be met in order to qualify the system for flight are then given. Finally, a description of the system is given. This description includes a brief overview of each of the units in the system, how they are installed in the aircraft, and how they are integrated into an operating system. Particular emphasis is placed on how fault tolerance is achieved to provide a very high level of functional reliability.

4.1 System Requirements

4.1.1 Mission-Specific Specifications

The requirements for the F-8 DFBW system can be divided into two categories: mission-specific and generic. The mission-specific requirements (shown in Figure 4-1) are those determined by operational considerations, installation constraints, program funding and schedule guidelines. The system was specified to be triplex, using government-furnished general-purpose digital computers, because a triplex configuration would present all the problems of multicomputer operation and could be installed within the volume available in the F-8.

Program funding did not permit the procurement of an inertial platform set, as might have been desirable in this program. Therefore, aircraft-quality rate gyros and accelerometers were specified. The sensor and command lines were specified to be dedicated hardwire. Multiplexing was not possible within program funding.

Experience in the Phase I program and a state-of-the-art assessment in actuator stabilization resulted in the specification that this stabilization be done using analog components, outside of the digital computer,

- TRIPLEX COMPUTERS/INTERFACE UNIT
- GOVERNMENT-FURNISHED COMPUTERS
- AIRCRAFT-QUALITY MOTION SENSORS

  TRIPLEX: INNER-LOOP CONTROL
  DUPLEX: AIR DATA, AUTOPILOT
- NO INERTIAL PLATFORM
- NO SENSOR OR COMMAND SIGNAL MULTIPLEXING
- ANALOG STABILIZATION/EQUALIZATION OF ACTUATORS
- ASSEMBLY-LANGUAGE PROGRAMMING
- INDEPENDENT ANALOG FLY-BY-WIRE SYSTEM FOR EMERGENCY BACKUP
- COMPUTER/IFU ON CENTRAL PALLET

Figure 4-1. Mission-specific requirements.

due to the sample rate requirements and computational burden. Current microprocessor technology makes digital control of actuators feasible and attractive. The use of a secondary actuator to drive the existing F-8 power actuators instead of a new integrated actuator was dictated by the burden of requalifying the primary actuation system of the F-8, including flutter clearance.

Assembly-language programming was specified for the F-8 DFBW system. This was due to the fact that a qualified high-order language was not available for the flight computer at the time programming was initiated.

The research nature of the primary DFBW flight-control system required an independent dissimilar backup control system. The primary motivation was to protect against a common-mode software error that would disable the entire primary system.

Finally, the available volume in the F-8 required custom packaging. The computers and interface units were specified to be mounted in a removable pallet assembly. Flight-control sensors were to be installed in easily accessible locations.

4.1.2 Generic Specifications

The generic specifications are those that tend to be independent of the particular application. They represent the fundamental operating characteristics of the system. In the case of the F-8 DFBW system, these requirements were selected to both tax the technology and to represent realistic and achievable specifications for a primary flight-control system.

4.1.2.1 Overall Fault Tolerance

The key system fault-tolerance issues can be stated in the following manner:

(1) No single fault in the primary digital system shall cause degraded inner loop performance.

(2) No second fault in the primary system shall result in a hazardous situation.

(3) No sequence of sensor or display failures shall result in an automatic transfer to the bypass system.

(4) No single fault in the primary digital system shall result in a transfer to the bypass system.

(5) The loss of two computing channels in the primary system shall result in an automatic transfer to the bypass system.

(6) No primary system fault sequence shall prevent manual transfer to the bypass system.

Figure 4-2 shows the fault-tolerance requirements for each major system. Generally, fail-operational requirements were specified for each system. A second like fault in any system has differing consequences, depending on the actual device faulted.

4.1.2.2 Primary-Digital-System Generic Requirements

The requirements for the primary digital system are of particular interest. Figure 4-3 lists the major requirements that were imposed on the primary system. The overall fault-tolerance requirement, as explained previously, was fail-operational, with the second like major channel failure resulting in an automatic transfer to the bypass system.

Single-channel digital operation was not permitted in the F-8 DFBW aircraft because of the experimental nature of the system. Automatic

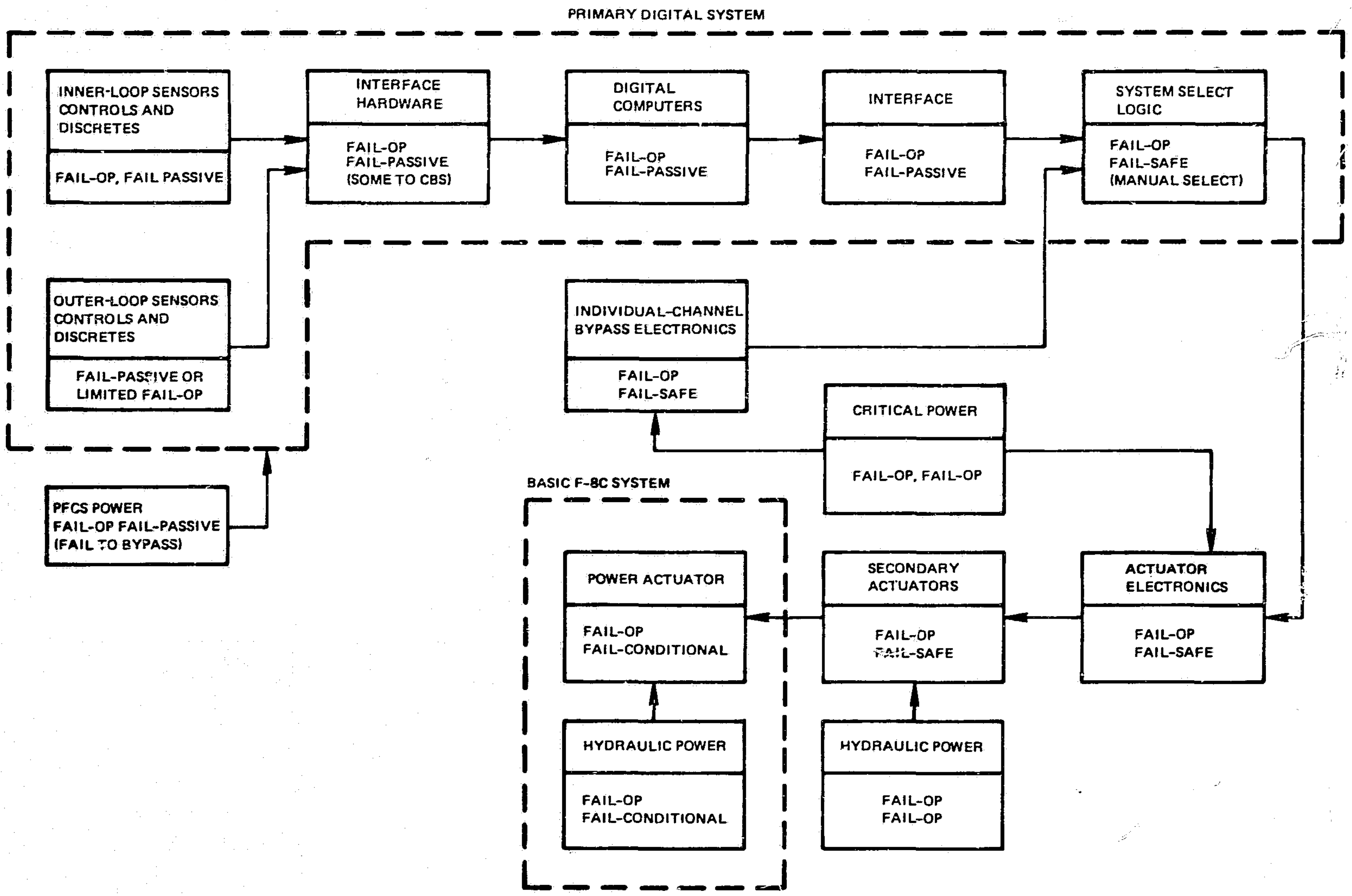


Figure 4-2. Overall system fault-tolerance requirements.

AREA	REQUIREMENT	IMPLICATIONS
FAULT TOLERANCE	● FAIL-OP/FAIL-SAFE ● SECOND FAIL TO BACKUP	● NO SINGLE-CHANNEL OPERATION ● REDUNDANT POWER SOURCES
TURN-ON/OFF	AUTOMATIC INITIALIZATION FROM ARBITRARY TURN-ON/OFF SEQUENCE	NO CREW ACTION PERMITTED FOR START-UP
FAULT DETECTION	● HARD FAILURES TO BE DETECTED WITHIN 200 ms ● HARD FAILURE DECLARATIONS TO BE IRREVERSIBLE	NO PROVISION FOR REINITIALIZATION BY PILOT
RECONFIGURATION	AUTOMATIC	NO CREW ACTION ASSISTANCE PERMITTED IN FAULT ISOLATION
TRANSIENT FAULT RECOVERY	FULLY RESTARTABLE IN ANY CONFIGURATION/MODE	CONTINUED OPERATION FOLLOWING: ● TEMPORARY POWER LOSS ● TRANSIENT HARDWARE/SOFTWARE PROBLEM
IMMUNITY TO FALSE ALARMS	DESIGN TO BE HEAVILY WEIGHTED TO AVOID FALSE ALARMS	NO QUANTITATIVE REQUIREMENT
OUTPUT COMMAND VOTING	ANALOG VOTING OF SURFACE COMMANDS	NO SOFTWARE VOTE OF SURFACE COMMAND
COMPUTER INTERCOMMUNI-CATION	MINUMUM POSSIBLE	REDUCE COMMON-MODE ERROR SOURCES
SYNCHRONIZATION	FRAME OR MINOR CYCLE ONLY	
CONTROL-LAW INTERFACE	MULTICOMPUTER STRUCTURE TO BE TRANSPARENT TO CONTROL LAWS	CONTROL LAWS WRITTEN AS FOR SINGLE COMPUTER
SYSTEM INTEGRITY	● FULL TIME ● FULL CONTROL SURFACE AUTHORITY ● FLIGHT CRITICAL CONTROL ● NO MECHANICAL REVERSION	MAN-RATING REQUIRED PRIOR TO FIRST FLIGHT

Figure 4-3. Generic system design requirements.

initialization from any arbitrary turn on/off sequence was specified so as to exclude the crew from any special action.

The 200-millisecond hard-failure fault-detection time was based on F-8 dynamic response at high dynamic pressure flight conditions. Hard failures were to be irreversible, with no provision for pilot reselection. All reconfiguration logic was to be automatic with no pilot participation permitted in the fault-isolation process.

The system was to be fully restartable in any mode following a transient fault. This meant that a channel or channels would be restored to normal operation following unspecified transient faults. There is always a problem in defining a transient fault. In the F-8 digital system, transient faults were divided into two categories: power loss (or apparent power loss) and all others.

The transient survivability times were defined as:

(1) Single-channel external-source power loss—no time limit.

(2) Multichannel power loss—40 milliseconds.

(3) Detected fault any type—200 milliseconds.

This meant that a single channel was required to be restored to normal operation after being powered down for any indefinite period of time. This requirement is also necessary in order to be able to turn the system on. If power was lost by two or three channels for less than 40 milliseconds, the primary system was to be restored to normal operation and continue to be in control of the aircraft. If this power loss occurred for more than 40 milliseconds, the bypass system was to effect an automatic takeover. It should be noted that the primary system was still required to be restored to normal operation following a long power interruption, even though command had been handed over to the bypass system.

For detected faults, that is, for conditions where execution is apparently continuing, but where an abnormal condition has been detected, including an internal power supply fault, the transient time was specified to be 200 milliseconds. This is based on 10 attempts to restore normal operation at the nominal 20-millisecond minor cycle, and is the maximum time a fault can be tolerated at critical F-8 flight conditions. These requirements are illustrated in Figure 4-4.

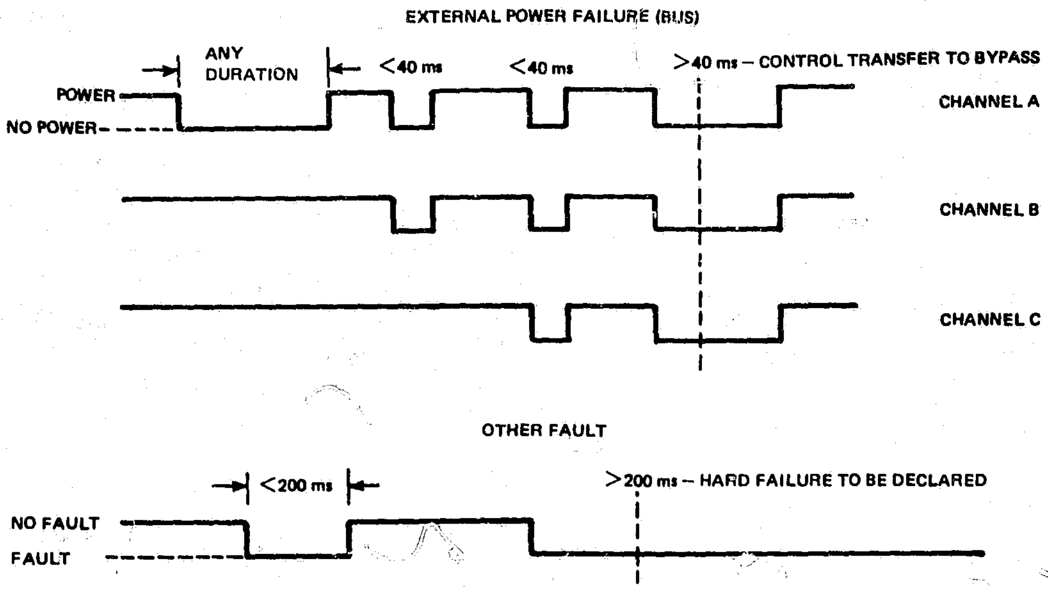

EXTERNAL POWER FAILURE (BUS)
ANY DURATION
<40 ms
<40 ms
>40 ms – CONTROL TRANSFER TO BYPASS
POWER
NO POWER
CHANNEL A
CHANNEL B
CHANNEL C
OTHER FAULT
<200 ms
>200 ms – HARD FAILURE TO BE DECLARED
NO FAULT
FAULT

Figure 4-4. Transient fault survivability requirements.

False-alarm immunity was recognized to be a critical characteristic of the DFBW system. A requirement was imposed that the system was to be weighted in favor of continued or restored operation in all cases possible. It was not known how this requirement could be proven analytically, thus no quantitative specification was given. Actual operating experience would give an insight into this feature of the system.

In the preliminary design it became apparent that special consideration had to be given to the problem of undetected digital system failures occurring in a sequence that would cause hazardous commands to be generated. The solution chosen was to require analog output voting on the digital-system surface commands. This approach was taken to protect against a catastrophic fault sequence in a manner independent of digital system software or failure-detection logic.

In an attempt to reduce the possibility of interchannel fault propagation, it was specified that intercomputer communication was to be kept to an absolute minimum, with design approaches deliberately avoiding complex computer intercommuncations.

Synchronization was specified explicitly to be frame or minor cycle only. Thus the computers, while being tightly synchronized, would not be in exact step. This was specified in order to permit a simpler interface unit design and to permit a "looser" more tolerant system operation.

The control-law or applications software was specified to be independent of the multicomputer structure, with the redundant hardware transparent to application routines.

Finally, overall system characteristics were specified. The digital system was to operate full-time and in fact be the primary (albeit experimental) flight-control system of the airplane. It would be used during the first takeoff and landing. The flight-critical system was to be given full surface authority in all three axes. The mechanical system had already been removed during the first phase of the F-8 DFBW program. It would not be available. These requirements meant that the primary system would have to be fully man-rated and flight qualified prior to the first flight.

4.1.2.3 Application Software Specification

The application software was specified to be a set of control modes typical of those projected for future active control vehicles. These control laws, in the form of a detailed software specification included:

(1) Pitch-Command-Augmentation System

(2) Lateral-Directional Augmentation

(3) Angle-of-Attack Limiter

(4) Ride-Smoothing System

(5) Maneuver Flap System

(6) Standard Autopilot Functions

(7) Control-Stick Steering

Figure 4-5 illustrates the major control-law functions and the control surfaces involved. Additional control laws were to be examined as part of the advanced control research program.

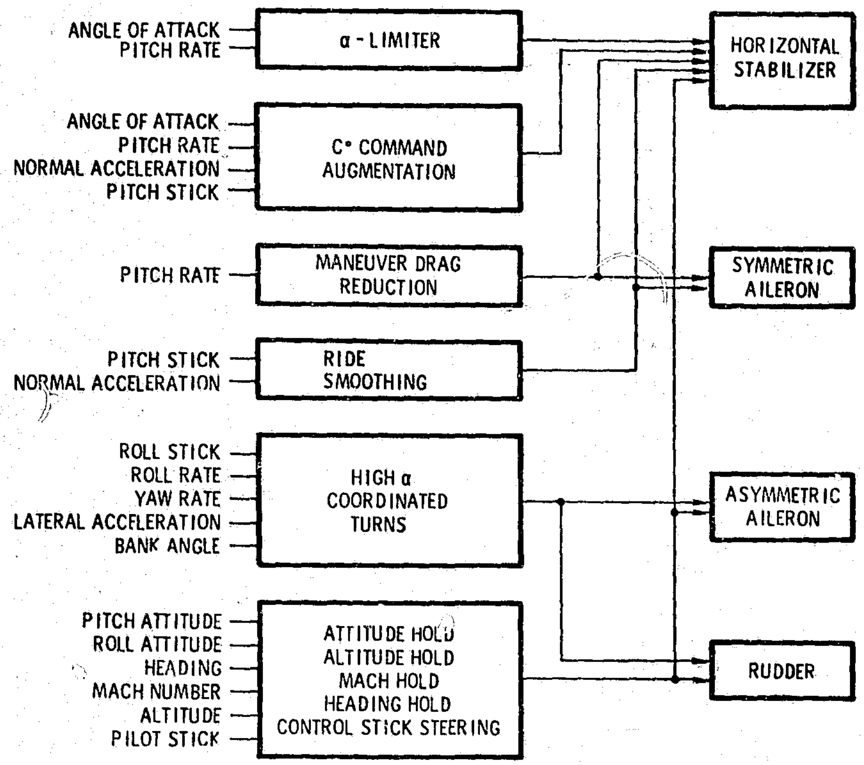


Figure 4-5. F-8 DFBW baseline control system.

4.1.3 Miscellaneous Requirements

An interactive built-in preflight test program was specified that would assure primary system integrity prior to flight. This test would not test software itself, but the system interfaces with the software. Figure 4-6 illustrates those interfaces to be examined in the preflight test, and the source of signal activation. The preflight test program was to be a non-obstrusive test routine, which, as much as possible, would utilize the actual operating software. Sequencing of the test was to be via the Cockpit Computer Input Panel. For hangar testing or postflight analysis, hard-copy test results were to be available. For the ramp preflight test, test results were to be logged, with the software making a Go/No-Go decision on the test data.

Several other requirements were specified because of the experimental nature of the primary digital system. These included:

(1) Real-time displays for ground testing.

(2) Extensive logging of computer and sensor redundancy management operation.

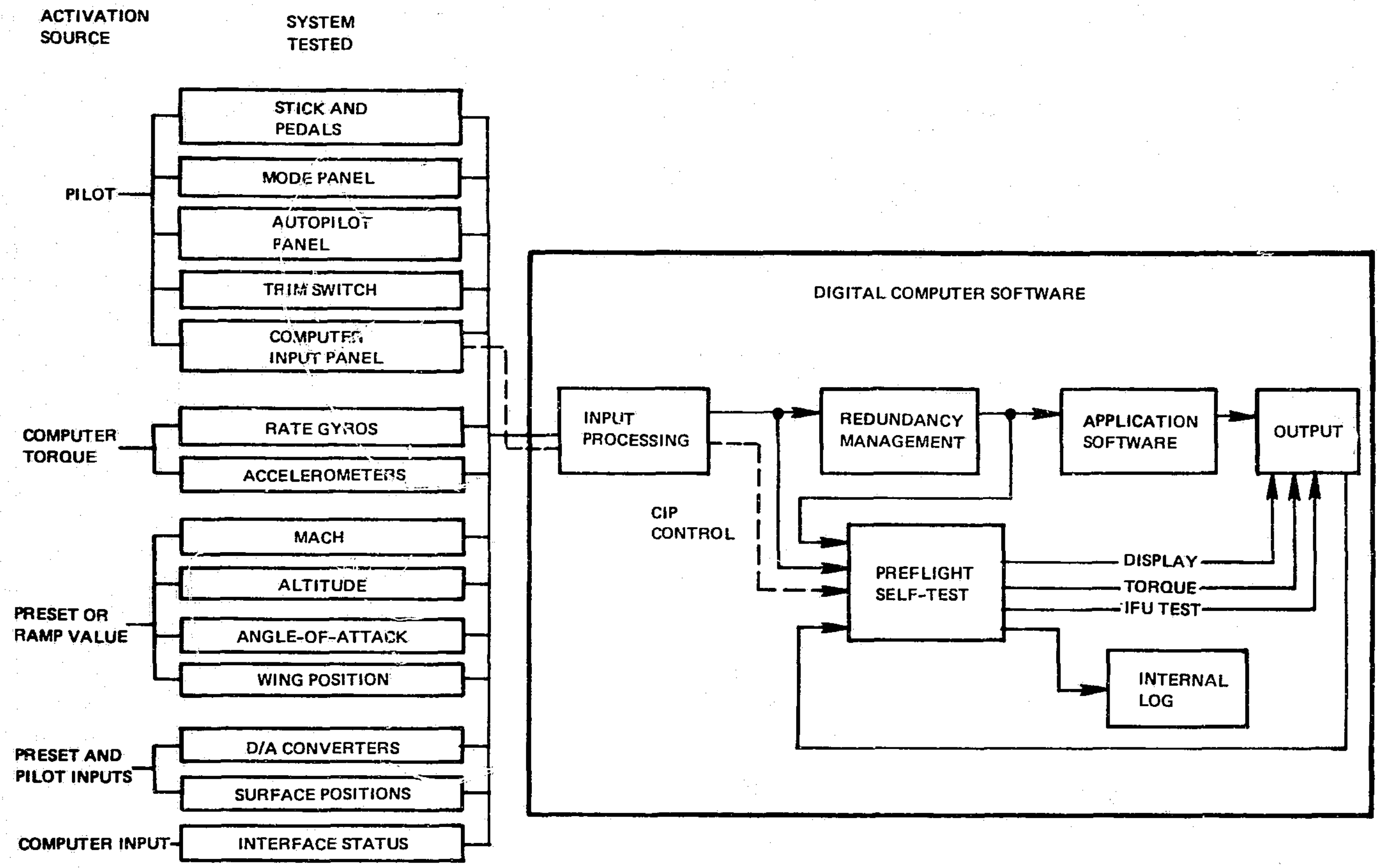


Figure 4-6. Preflight self-test software interfaces.

(3) Logging of alarms representing abnormal performance.

(4) Data link to an onboard tape recorder of 1000 32-bit words per second.

(5) Various pilot-initiated programs for the Computer Input Panel.

4.2 Flight-Qualification Requirements

Flight qualification involved all the steps in the process whereby the DFBW system was determined to be ready for first flight. This process included hardware, software, and system qualification. Much of the hardware qualification, especially that for the nondigital elements, involved practices commonly used and well understood, and hence, is only lightly treated in this report.

Figure 4-7 illustrates the overall flight-qualification process for the F-8 DFBW system. For purposes of illustration, the hardware and software paths are shown completely distinct, although that is somewhat misleading. In fact, where there is not a very close technical and organizational interface between the two paths, success is improbable. While there are some unique aspects of software, it should be pointed out that the development and qualification processes are very similar to those of hardware. That is, there are parallel and analogous steps in the two paths. In this respect, software development is more like hardware development than like something else. The following sections describe the requirements that were imposed as part of the overall qualification process.

4.2.1 Design Requirements

Configuration control was required in the design of both hardware and software. For hardware, standard practice was followed in the design by the use of revision control for drawings and documents, engineering-change report forms, and an overall configuration control plan.

In the case of software, similar requirements were imposed prior to software design. These were composed of the following specific items:

(1) Flow Charting: It is exceedingly difficult to determine whether the software design meets the intent of the specification if the software engineer never explicitly reveals his intent. In software, more so than in hardware, it is

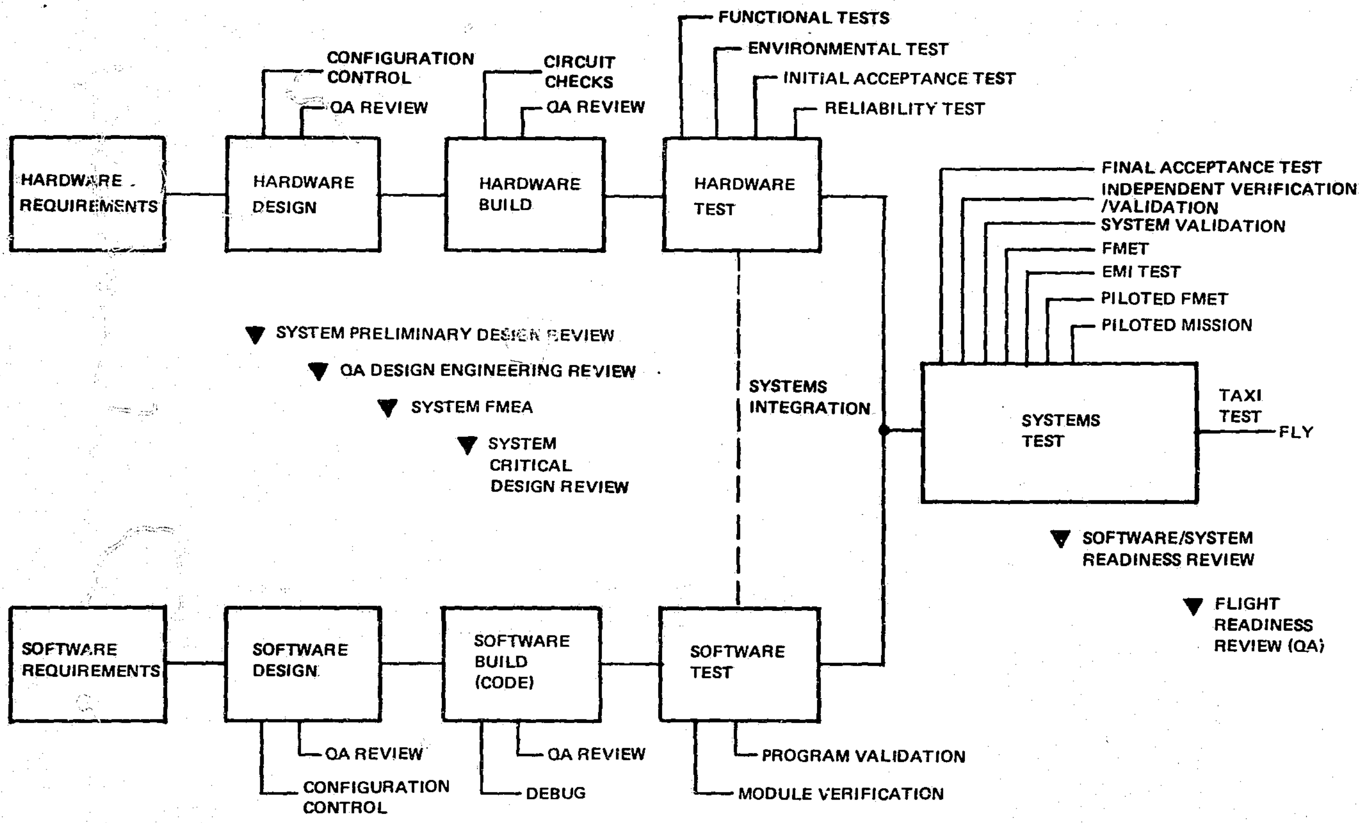


Figure 4-7. Flight qualification process.

necessary to describe what is intended to be done; in the F-8 program, flow charting was required for all but the most obvious functions.

(2) Design-Change Documentation: Prior to coding, design changes were incorporated into the software specification, the ruling document.

(3) Engineering Documentation: A memorandum system was used to document the rationale for algorithm implementation where it was not straightforward. These documents were invaluable in the ensuing months and years when changes were considered.

(4) Modular Structure: This favorite expression is meaningless unless explicitly defined. Formal structured programming principles were not specified, but functional partitioning was required between the operating system (executive), system redundancy management, control laws, downlink, self-test, and I/O. It was desired that changes in these elements could be made independently. Within the control laws, very specific modularity was specified which was mission-unique.

(5) Built-In Test Points: One of the very critical qualities required of software is testability. If test points and test access are not designed into the software, verification and validation are hampered, because special software must then be patched in or the flight code must be altered. Both options are undesirable as they occur late in the software build process and have a high potential for introducing errors into the flight software. In the F-8 program, intermediate computations were required to be accessible. This increased timing and core requirements, but substantially eased the verification job. In addition, several special displays and a data link were built in to provide real-time data recovery.

The quality assurance steps in the design phase are carried out by designers insofar as they seek independent assessments of their design. This was not an explicit requirement, however.

Two system-level reviews were held during the design phase. One was a technical preliminary design review. For software, this involved the review of top-level flow charts. The second review was part of the regular DFRC Quality Assurance (QA) policy, and involved an independent assessment of the design from safety, reliability, and mission-success points of view. This review was conducted by DFRC personnel not directly involved in the program.

4.2.2 Build Requirements

Whereas the hardware build phase represents a relatively straightforward transformation of design drawings to actual circuit layout, software build often involves a more artistic transformation of flow-charted design into actual code. Good programming practice, experience, and an understanding of the problem greatly reduce the artistic element. NASA DFRC did not specify coding strategy, but there were programming rules enforced within the programming team.

Debug, representing elementary checkout of bits and pieces of the code to achieve execution, corresponds closely to circuit checks that are made in hardware as it is built. Quality assurance in the hardware build is comprised of inspection. Software quality assurance is today only an emerging technique in this phase of software development. In the F-8 program, independent inspection by other programmers comprised quality assurance. This was not required formally by NASA, however.

The failure modes and effects analyses were to be carried out during this period of time. This consisted of analyses for the actuators, bypass system and servo electronics, and for the primary digital system. An independent FMEA was not required for the digital computer. Rather, fault detection capabilities within the computer software and hardware were considered in the system FMEA.

The system critical design reviews are generally conducted before significant amounts of the hardware or software were built, (less than 10 percent). Detail-level flow charts and skeleton code are reviewed in the case of the software.

4.2.3 Stand-Alone Tests

Prior to system integration, stand-alone hardware and software testing was required. For hardware, these tests were:

(1) Functional Tests: Testing of entire functions was required such as analog-to-digital input, and digital-to-analog output, signal processing, etc. Simple test software routines were used to drive the hardware.

(2) Environmental Tests: Temperature, altitude, and vibration tests were required on the entire flight hardware pallet. In these tests, it would be desirable to have software which exercised every function in the digital computer and in the interface hardware simultaneously. Such a program would have been very expensive to build and so compromises were necessary. Computer environmental tests were to be performed separately with the manufacturer-supplied functional test program which would thoroughly exercise the computer operation. Flight pallet environmental tests were to be conducted using special test software which would exercise the major system functions.

(3) Initial Acceptance Test: This is a functional test using an operational software package. It was to be performed prior to delivery by the contractor, and after initial systems integration. The test was to demonstrate top-level functional operation and all hardware interconnections. It was to be performed in a stand-alone open-loop manner.

(4) Reliability Test: The requirement was to have 100 continuous failure-free hours of operation within 200 hours of normal utilization.

Stand-alone software testing was specified as consisting of module verification and program validation:

(1) Module Verification: These tests were to show that the individual modules of code executed and met the requirements of the software specifications, such as the sensor read module.

(2) Program Validation: These tests were to show that the collection of modules operated together to perform the intended functions of the Software Specification, such as computer synchronization.

4.2.4 System-Level Tests

The system-level tests were to be accomplished at NASA DFRC and were composed of:

(1) Final Acceptance Test: This was to be similar to the Initial Acceptance Test except it would be accomplished with the system installed in the iron bird.

(2) Independent Verification and Validation: These tests would duplicate most of the module verification and program validation tests previously accomplished, but would be accomplished on an operational system.

(3) System Validation Tests: These tests were for the purpose of establishing that the system met the mission requirements for closed-loop fault-tolerant primary flight control. They were to include dynamic control law tests, and fault-tolerance tests.

(4) Failure Modes and Effects Tests: These tests were to be accomplished to validate the failure modes and effects analyses. They were specified to be based on the manifestations of component faults, not on component faults themselves. Thus, this testing would not include a part-by-part fault test. Component faults within the computer were not to be induced. Thus, a computer was to be considered a single item which could have categories of fault manifestations.

(5) EMI Tests: These modest tests were specified to assure non-interfering operation of the flight control system and other aircraft avionics, and to verify digital system immunity to noise induced on the power lines themselves.

(6) Piloted FMET: These tests were to demonstrate acceptable closed-loop reaction to categories of faults previously examined in the detailed failure modes and effects tests, and to verify fault-tolerance requirements of major subsystems, such as actuators, hydraulics, electrical power, etc.

(7) Piloted Mission: These tests were to demonstrate mission suitability of the DFBW system. It was specified that some of these tests would be accomplished using all flight hardware in the iron bird.

The technical software/system readiness review was identified to take place when all testing had been completed, and the system was technically considered ready for flight. The flight readiness review is a QA function which involves the concurrence of an independent board on the state of flight readiness. The final determination of flight readiness rests with the Flight Operations Directorate.

The similarities between hardware and software development requirements far outweigh the differences. The form of the object under development is, of course, different. The F-8 DFBW flight qualification requirements started in the design, as does any good flight qualification plan. It depended heavily on traceability, documented configurations, independent verification and validation, and independent mission safety assessments. It also depended heavily on pilot assessment in the iron-bird simulation.

4.3 The Phase II F-8 DFBW System(16,17)

The major components of the F-8 DFBW system are (refer to Figure 2-16 for the basic configuration):

(1) Digital Computers (3) (see Figure 4-8).

(2) Interface Unit (IFU) (3 independent sections in one chassis (shown with the computers in Figure 4-9).

(3) Sensor Pallet (3 rate gyros on each axis and 3 accelerometers on each axis for a total of 18 sensors) (see Figure 4-10).

(4) Additional Sensors (2 heading and attitude systems, 2 angle-of-attack sensors, 1 slideslip sensor, and 3 sensors for each pilot control).

(5) Cockpit Control and Display (Encoder/Decoder, Mode and Gain Panel, Annunciator Panel, Digital Autopilot Panel, and Computer Input Panel) (see Figure 4-11).

(6) Computer Bypass and Servo Electronic System (CBS) (3 including command-signal voting and servo amplifiers) (see Figure 4-12).

(7) Secondary Actuators (5 including right and left aileron, right and left elevator, and rudder) (see Figure 4-13).

These major components will be described individually to give an understanding of the basic characteristics and operation of each unit. The physical installation of this equipment in the aircraft will then be briefly outlined, and the overall operation of the system described. Particular emphasis will be placed on the failure-detection and redundancy management techniques.

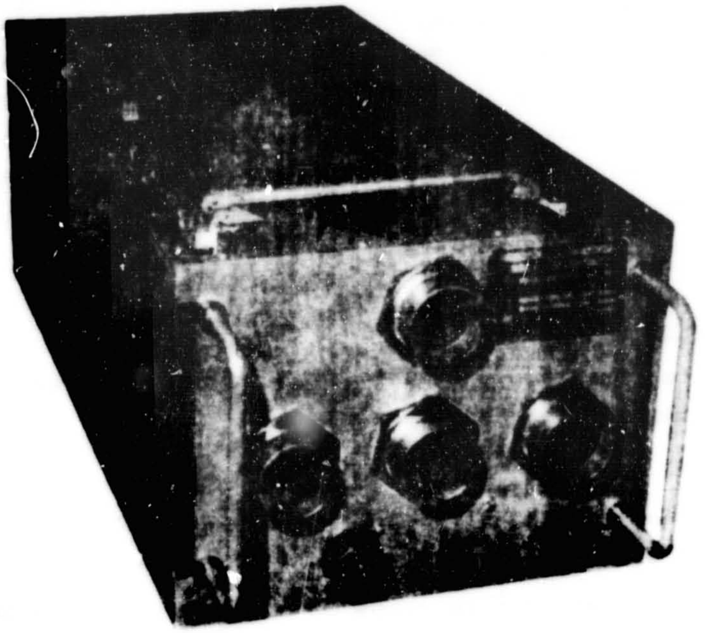


Figure 4-8. Central processor unit.

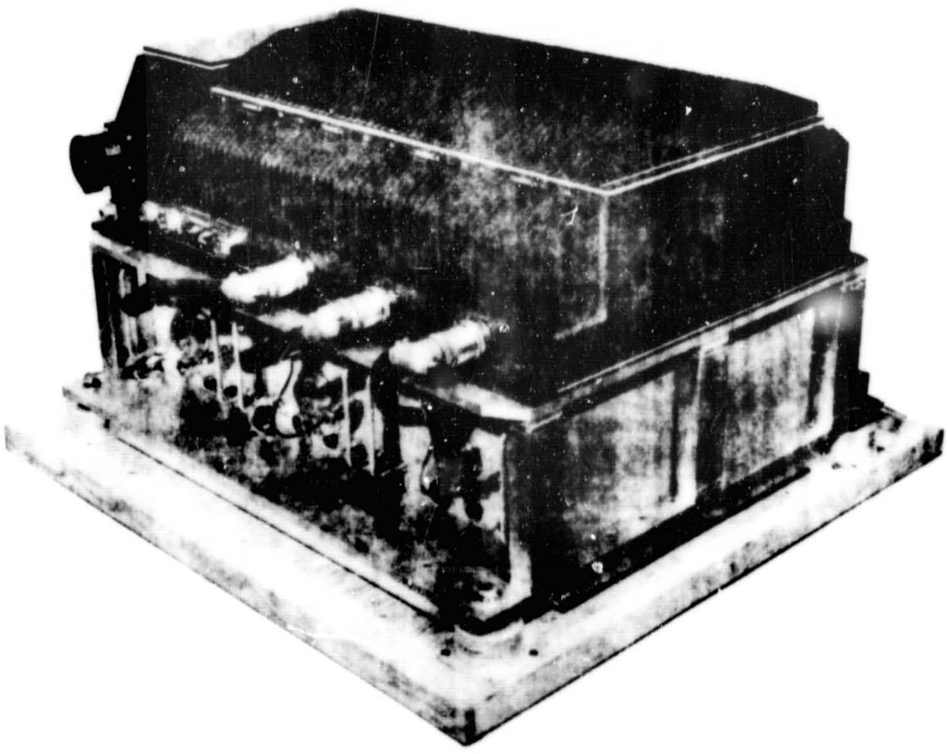


Figure 4-9. Pallet assembly containing the interface unit and three central processors.

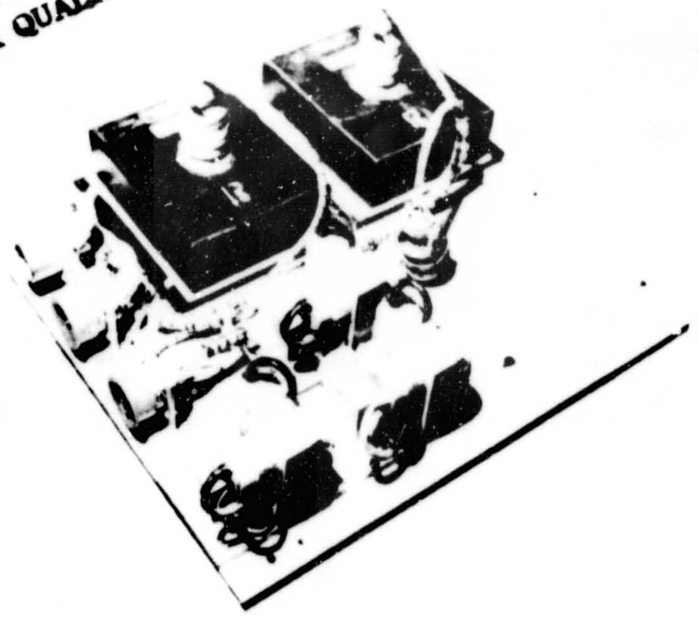


Figure 4-10. Inertial sensor assembly.

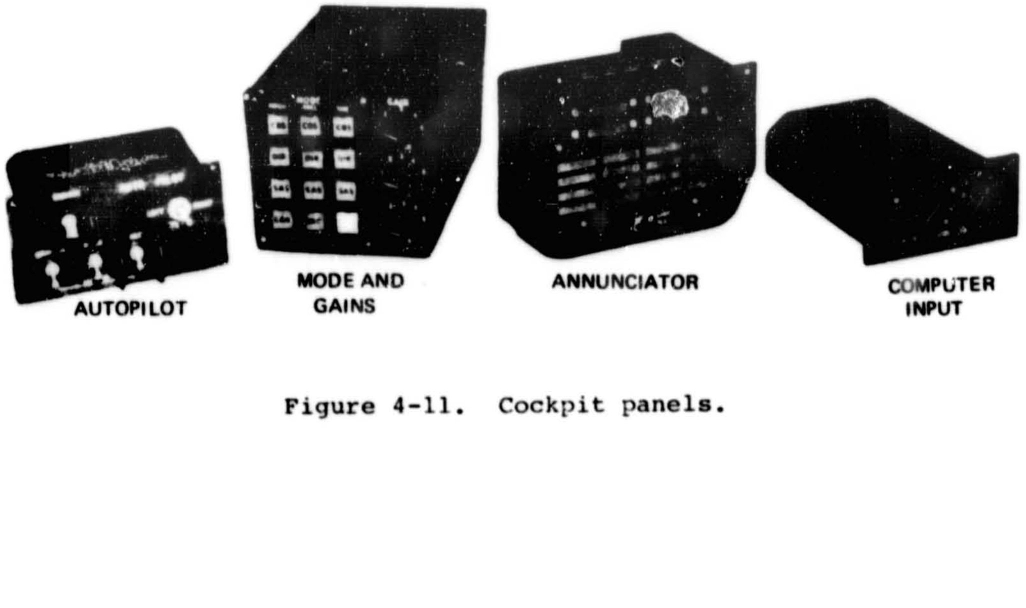


Figure 4-11. Cockpit panels.

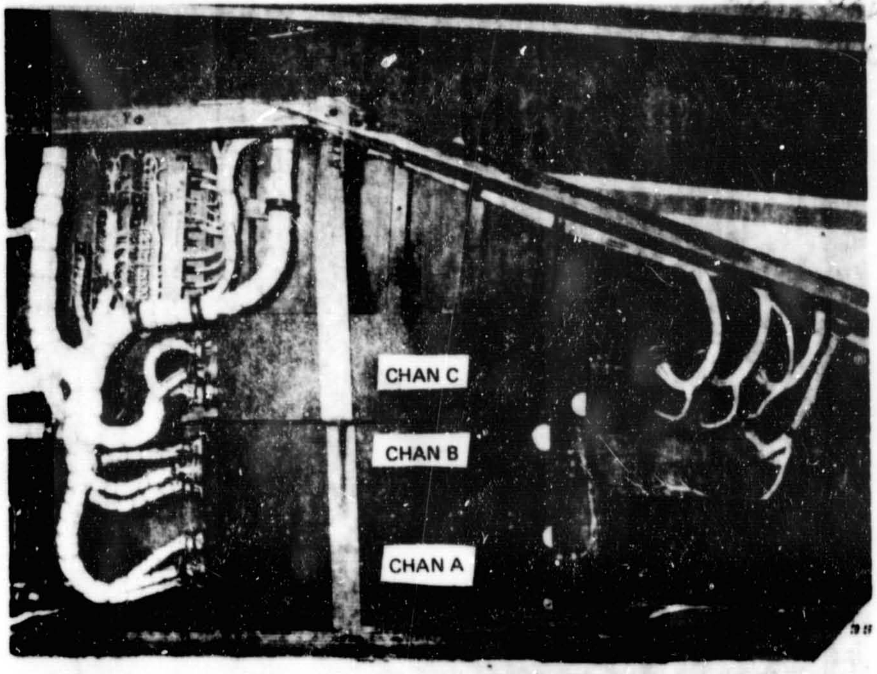


Figure 4-12. Computer bypass and servo electronics system installed in airplane.

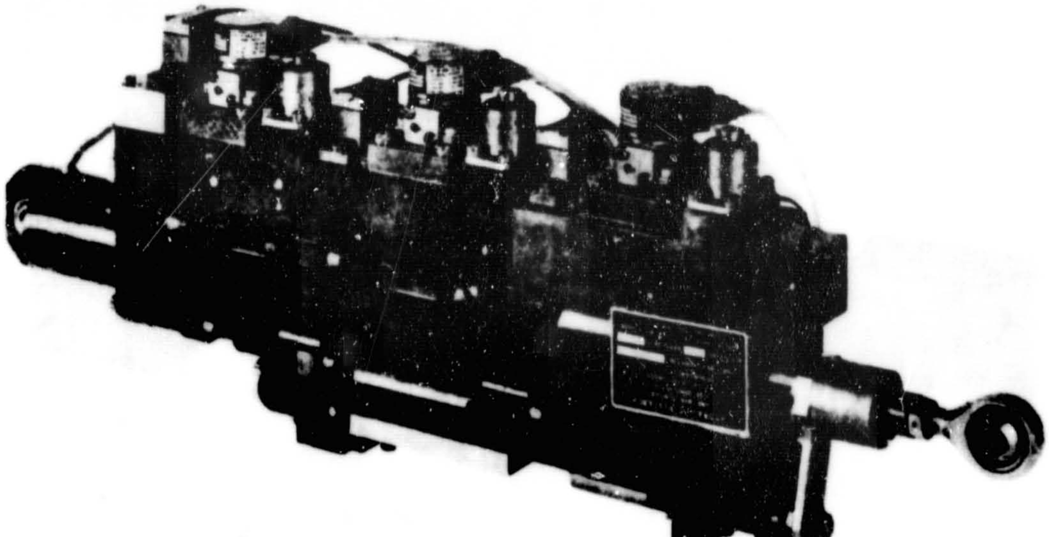


Figure 4-13. Triplex secondary servoactuator assembly.

4.3.1 DFBW F-8 Major Components

4.3.1.1 Computer

The AP-101 computer used in the F-8 DFBW is similar to that used in the Space Shuttle. This computer was developed over the 1972-1973 time period. It is a general-purpose stored-program parallel machine. It works with both 16- and 32-bit words. It has a microprogrammed instruction set with 146 total instructions. These instructions include both fixed-point and floating-point operations. The fixed-point add time is approximately 1.5 microseconds, and multiply is approximately 5.5 microseconds. Floating point add and multiply are approximately 3 and 6 microseconds, respectively. The AP-101 uses a magnetic-core memory with 32,768 36-bit words. The words include two parity bits and two store protect bits. The primary I/O for the computer is one 16-bit parallel channel with a 450,000 halfword per second data rate. Direct-memory access is available through this channel. There are also five external interrupts, four discrete inputs and outputs, and a real-time clock. The AP-101 is basically one full ATR (19.3 × 25.6 × 49.8 cm) and weighs 26 kg. Its primary power is 28 Vdc and it uses 375 watts; it is cooled by individual blowers. Figure 4-14 is an expanded view of the internal construction of the computer. The major characteristics are summarized in Table 4-1.

4.3.1.2 Interface Unit (IFU)

The IFU contains the equipment necessary to process and condition the I/O signals for the three digital flight computers. The IFU was specially designed and built for the F-8 DFBW program. There are actually three electrically independent IFU channels, one for each processor. Because of F-8C installation requirements the three channels are packaged within a single enclosure. Each IFU channel is interfaced with only the one processor, and can be thought of logically as part of the processor.

Each IFU channel is responsible for four major functions:

(1) To provide conditioning of input signals, convert the analog signals to digital form, and provide buffer memory for input data.

(2) To process output signals and perform digital-to-analog conversions.

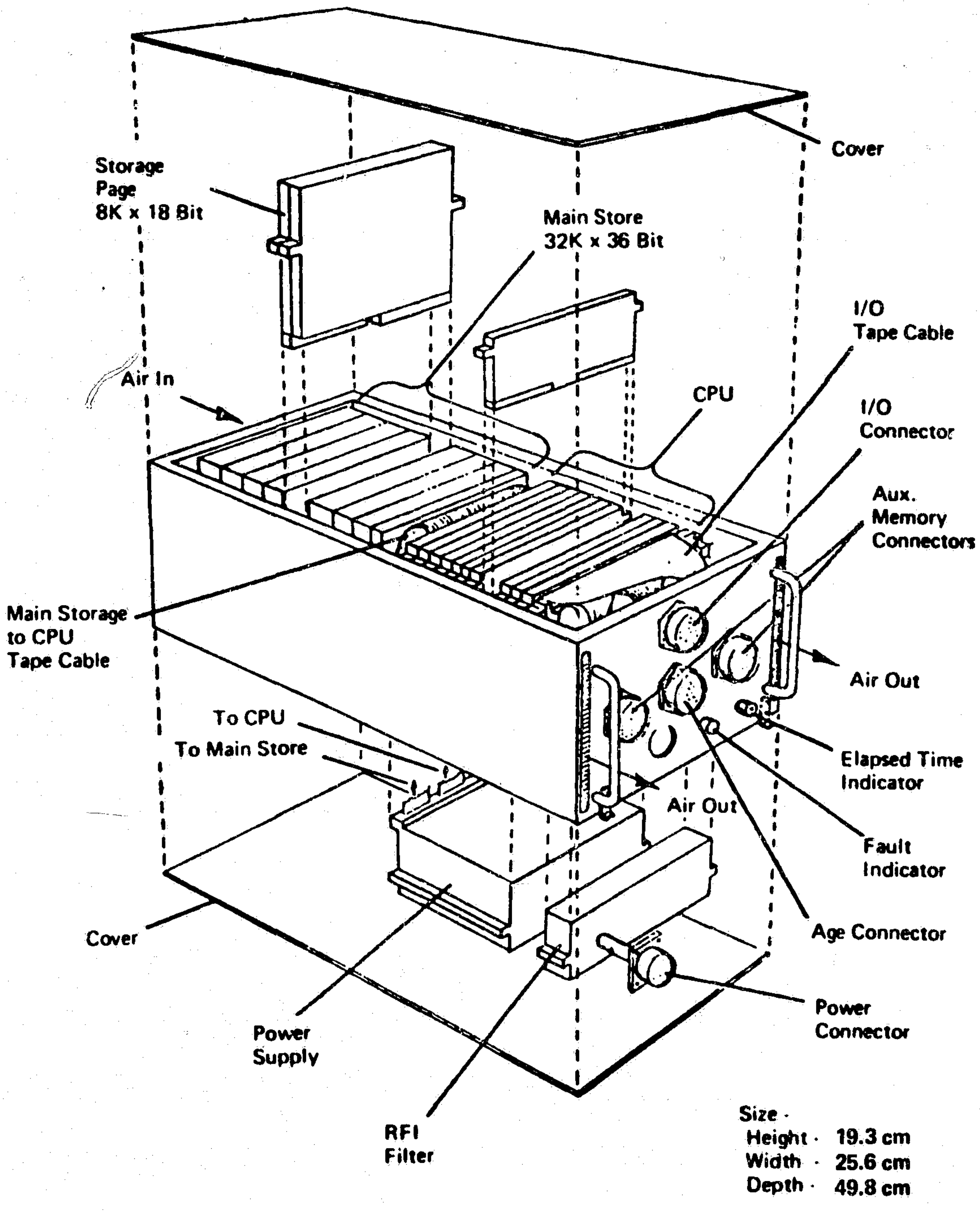


Figure 4-14. Computer internal construction.

Table 4-1. Digital control computer characteristics.

Component	Characteristic
Central Processor Unit	
Number system	Binary, fixed point, two's complement, fractional floating point
Operation	Full parallel
Fixed-point data	16 and 32 bits, including sign
Floating-point data	32 bits (24-bit mantissa) and 64 bits (56-bit mantissa)
Typical execution times, register to register: (μs)	
Fixed point:	
Addition	1.2
Multiplication	5.2
Division	9.0
Floating point:	
Addition	3.2
Multiplication	6.0
Division	10.0
Average instruction rate (per second)	480,000
Main Storage	
Type	Random access, destructive readout, ferrite magnetic core, nonvolatile
Capacity	32,768 words, 36 bits per word
Cycle time (ns)	900 read/write
Addressable unit	16-bit halfword
Features	Parity and store protect on halfword
Input-Output	
Type	Parallel halfword plus parity, multiplex, half duplex
Maximum data rate (per second)	225,000 full words
Discretes	Four input, four output
External Interrupts	Five levels

Table 4-1. Digital control computer characteristics. (Cont.)

Component	Characteristic
Physical Characteristics	
Size	1 ATR (19.3 cm high × 25.6 cm wide × 49.8 cm deep)
Weight (kg)	26
Volume (m^3)	0.025
Power (W)	375
Environment	MIL-E-5400 Class 2X

(3) Provide for interchannel data transfer between computers.

(4) Participate in fail detection and redundancy management.

A functional diagram of the IFU is shown in Figure 4-15.

Control—The IFU is designed to provide as simple an interface as possible for the computer programmer, with a minimum load placed on the processor. Each channel contains a microprogrammed controller that decodes the computer commands and performs the requested input or output function. The AP-101 computer has two forms of I/O that are used in the F-8 DFBW system; Direct Output and Buffer I/O. The Direct Output is used by the computer to command the IFU to perform a particular input or output operation. The Buffered I/O form is used by the IFU to execute the requested operation. The IFU either writes input data into computer memory or reads data out of computer memory for use as output.

Input—The IFU receives three different types of input signals; analog, discrete, and serial digital. The IFU is capable of handling 32 analog signals including dc, ac, and synchro. A list of the analog signals used is given in Table 4-2. There is capability for 80 discretes; Table 4-3 is a list of the input discretes used. There are also serial data inputs used for altitude and the encoder/decoder, which provides the interface with the pilot control panels.

Output—Each channel of the IFU has capability of outputting 10 analog signals with 12-bit resolution and 28 discretes. In the flight configuration, only four analog outputs are used. Three of these outputs go to the horizontal stabilizer, ailerons, and rudder. The fourth

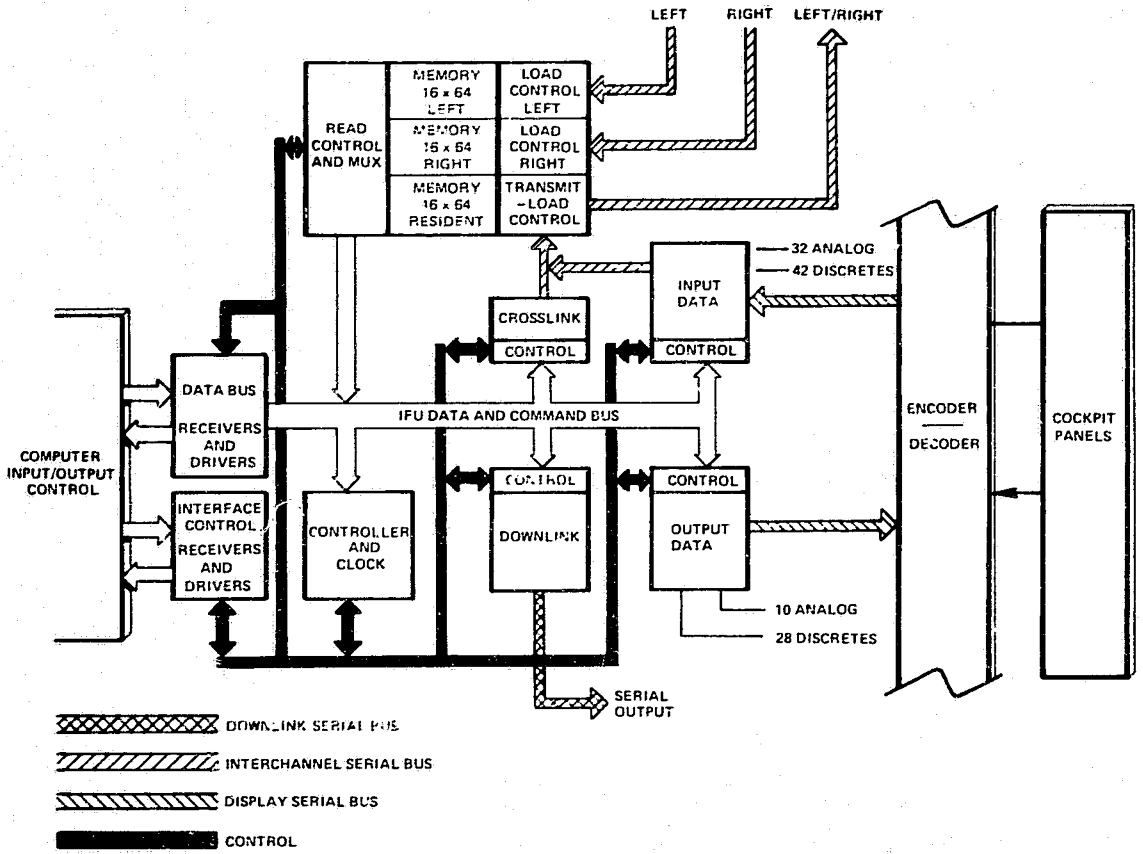


Figure 4-15. IFU simplified functional diagram.

Table 4-2. Input sensors.

Sensor	Redundancy Level	Signal Type
Pitch rate	3	dc
Roll rate	3	dc
Yaw rate	3	dc
Axial acceleration	3	dc
Lateral acceleration	3	dc
Normal acceleration	3	dc
Pitch center-stick position	3	ac LVDT
Roll center-stick position	3	ac LVDT
Pitch side-stick force	3	ac LVDT
Roll side-stick force	3	ac LVDT
Rudder pedal position	3	ac LVDT
Angle of attack	2	dc
Angle of sideslip	1	dc
Horizontal stabilizer actuator position	3	dc Potentiometer
Surface positions (5)	1	dc Potentiometer
Pitch attitude	2	Synchro
Roll attitude	2	Synchro
Heading angle	2	Synchro
Wing position	3	dc Potentiometer
Mach number	2	dc Potentiometer
Altitude	2	Serial digital
Computer temperature	1	dc

is a command for symmetric ailerons, which provides for flaps and can be used for direct lift control. Serial digital data is provided for the encoder/decoder. Also in this experimental system is a serial output to a data recorder.

Buffer Memory—Each IFU channel uses three buffer memories. Each memory has a capacity of 64 16-bit words. One memory is for the data from the resident channel and the other two are for data from the other two channels. These two memories are used to allow data from the other two channels to be available to each computer and also to allow computer-to-computer data transfer. A buffer memory is loaded by the sending channel and unloaded by the receiving channel. The operation of the IFU is described in the following paragraphs.

Table 4-3. Input discretes.

Discrete	Redundancy Level
Mode select	3
Autopilot panel	2
Annunciator reset	2
Trim	2
Side-stick enable	3
CIP digits	1
CIP enter/clear	2
Wing up/down	3
Weight on wheels	1
Gain switches	1
Gear position	3
Autopilot disconnect	1
Computer bypass status	3

Operation of the IFU—An input or output transaction is initiated by command from the computer. There are five commands: Data In, Data Out, Crosslink Out, Crosslink In, and Downlink.

Data In— When the computer commands Data In, the IFU converts the analog and discrete data into the proper form, placing it simultaneously in the buffer memory of that IFU chanel and in the buffer memories reserved for that data in the other two IFU channels. Figure 4-16 is a diagram of these connections. When the IFU has completed conversion of the first analog parameter, it prepares to send the data to the computer from all three buffer memories. However, the IFU channel must wait until the other channels have sent their first encoded analog parameter to their buffer memory. Operation Timeout counter prevents the failure of one channel from preventing the data from being obtained from the operating channels. If the timeout expires before the first parameter is received, the channel goes ahead, sending the data that is available. The data from the three memories is loaded alternately one word at a time so that the redundant-sensor information will be in adjacent locations. The complete analog parameter format is shown in Table 4-4. The DRRAV parameters are associated with the remotely augmented vehicle mode, which will be discussed later. Most of the rest of the parameters are self-explanatory.

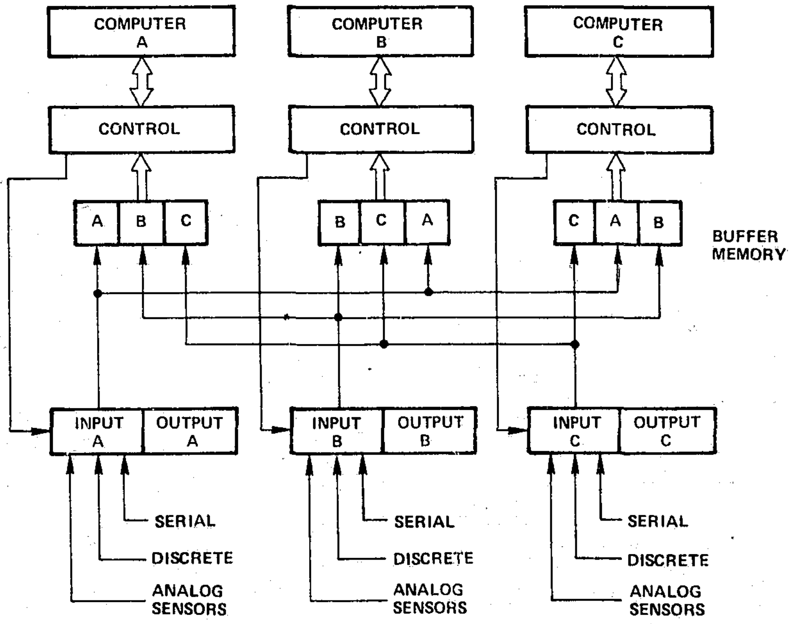


Figure 4-16. Input.

Data Out—When the IFU receives the Data Out command, it reads data out of computer memory and routes the data to the proper place to provide serial digital data output, discrete output, or to signal the digital-to-analog converters to provide the analog outputs. The analog outputs are wrapped around as analog inputs to provide monitoring of the outputs.

Crosslink Out and Crosslink In—The Crosslink commands are used to transfer data between computers. The crosslink process is shown in Figure 4-17.

Downlink—The Downlink command is used to move data from computer memory to a special buffer in the IFU. This data is then sent to an onboard digital recorder. The buffer memory holds 128 17-bit words. Data can be recorded at a rate of 2100 halfwords per second.

4.3.1.3 Sensors

There is a sensor pallet that was assembled and installed in the aircraft specifically for the DFBW system. The DFBW system also used several other aircraft sensors.

Table 4-4. Analog parameter list.

No.	Channel A		Channel B		Channel C	
0	DRRAV	A	DRRAV	B	DRRAV	C
1	DERAV	A	DERAV	B	DERAV	C
2	DFRAV	A	DFRAV	B	DFRAV	C
3	DARAV	A	DARAV	B	DARAV	C
4	Pitch Rate	A	Pitch Rate	B	Pitch Rate	C
5	Roll Rate	A	Roll Rate	B	Roll Rate	C
6	Yaw Rate	A	Yaw Rate	B	Yaw Rate	C
7	Axial Accel	A	Axial Accel	B	Axial Accel	C
8	Latl Accel	A	Latl Accel	B	Latl Accel	C
9	Norm Accel	A	Norm Accel	B	Norm Accel	C
10	P Ctr Stk	A	P Ctr Stk	B	P Ctr Stk	C
11	R Ctr Stk	A	R Ctr Stk	B	R Ctr Stk	C
12	Pitch S Stk	A	Pitch S Stk	B	Pitch S Stk	C
13	Roll S Stk	A	Roll S Stk	B	Roll S Stk	C
14	Yaw Pedal	A	Yaw Pedal	B	Yaw Pedal	C
15	Alpha	A	Alpha	B		
16	Beta	A	—		—	
17	L Elev Posn	A	L Elev Posn	B	L Elev Posn	C
18	R Elev Posn	A	R Elev Posn	B	R Elev Posn	C
19	Pitch Att	A	Pitch Att	B	—	
20	Roll Att	A	Roll Att	B	—	
21	Yaw Att	A	Yaw Att	B	—	
22	Wing Posn	A	Wing Posn	B	Wing Posn	C
23	Mach Coarse	A	Mach Coarse	B	L Ail Cpt	
24	Mach Fine	A	Mach Fine	B	R Ail Cpt	
25	Rudder Cpt		L Elec Cpt		R Elec Cpt	
26	DAC 00 Wrap	A	DAC 00	B	DAC 00	C
27	DAC 01 Wrap	A	DAC 01	B	DAC 01	C
28	DAC 02 Wrap	A	DAC 02	B	DAC 02	C
29	DAC 03 Wrap	A	DAC 03	B	DAC 03	C
30	+28 Vdc Bus	A	+28 Vdc	B	+28 Vdc	C
31	Comp Temp	A	Comp Temp	B	Comp Temp	C

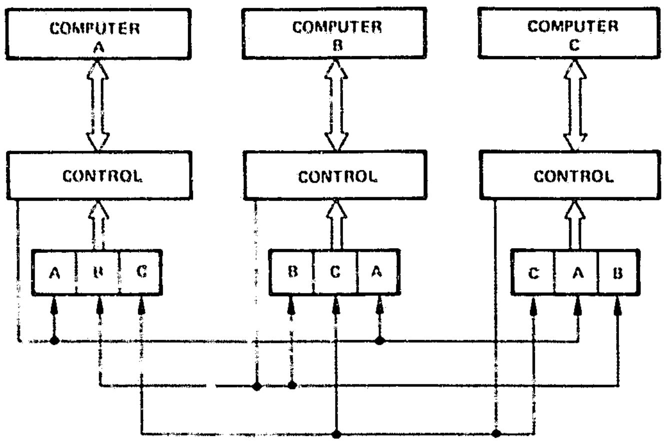


Figure 4-17. Crosslink.

The sensor pallet consists of nine gyros and nine accelerometers. There are three gyro assemblies, with three gyros in each assembly. Each gyro in an assembly is mounted parallel to one major aircraft axis. The roll-axis gyros have capability to 240 degrees per second and the other two axes use 70-degree-per-second gyros. The arrangement for the accelerometers is the same as for the gyros. The normal accelerometers are 10 g and the lateral and longitudinal accelerometers are 2 g.

The operation of the gyros and accelerometers is tested during the preflight test procedure by torquing the sensors in each direction by a command from the computer through the IFU. The gyros are also monitored continuously by a spin-motor run-detection sensor, which gives a discrete input to the IFU when the wheel is turning.

The DFBW system also uses several other sensors, which are distributed about the aircraft. There are triple-LVDT stick-position sensors for both roll and pitch control and also triple-LVDT sensors for the rudder pedals. There is also an experimental side-stick controller. Thus, there are triple-LVDT force sensors for both roll and pitch from the side stick. The system receives inputs from two angle-of-attack sensors and one sideslip sensor. Two heading and attitude reference systems are used in the system. Each provides three synchro signals: one for pitch, one for roll, and one for heading. Mach inputs are obtained as dc signals from two Mach meters. Altitude is obtained as serial digital signals from two altimeters. There are also position sensors on the control surfaces, the horizontal stabilizer actuator, and the wing. The F-8C has a two-position wing in order to reduce the attitude of the fuselage during landing.

4.3.1.4 Pilot Control and Display

The F-8 DFBW has four pilot control and display panels. These units allow adequate visibility and flexibility in an experimental system. Many of the functions would not be necessary or desired on a production system. Each of the panels is interfaced to the IFU and computers by the encoder/decoder unit.

Mode and Gain Panel (see Figure 4-11)

The Mode and Gain Panel provides control of the major modes of the system and allows pilot control of selected parameters. There are separate mode controls for each channel: roll, pitch and yaw. Each switch has triple contacts; these mode switches allow manual switching between the digital and analog systems. With the digital system there is also the choice between the Direct Mode and augmented modes. The Direct Mode gives a direct connection between the pilot controls and the aerodynamic control surfaces.

In the pitch channel, there is simple stability-augmentation (SAS) mode and a more highly augmented command-augmentation (CAS) mode. There are also lateral-direction SAS modes. The mode switches are lighted to indicate which mode is active. These lights indicate the mode both when manually selected or when the mode changes automatically due to system-detected faults.

There are three five-position gain switches on the panel. These switches can be tied through software to any parameter for which adjustment is desired during that particular test flight.

Digital Autopilot Panel (see Figure 4-11)

The Digital Autopilot Panel is very similar to traditional autopilot control panels. It has a magnetically latched engage switch and mode switches for altitude hold, Mach hold and heading. It also has a turn control switch.

Annunciator Panel (see Figure 4-11)

The Annunciator Panel is capable of displaying 20 separate indications; four of the indicators have a switch for reset. These displays include:

(1) Hardware-Detected Failures: Channel A Fail, Channel B Fail, Channel C Fail.

(2) Software-Detected Resettable Failures: Trim, Downmode, Self-Test.

(3) Status and Software Detected Failures: A Temp, B Temp, C Temp, P RAV, R RAV, Y RAV, Flap, Air Data, Alpha, Center Stick, Side Stick, Rudder Pedal.

Computer Input Panel (see Figure 4-11)

The Computer Input Panel allows the pilot to initiate preprogrammed software functions. These include the control of an extensive preflight test program. In flight, control-system options can also be selected. The panel has two thumbwheel switches to select the program. The selected program is displayed on a three-digit display. The program is initiated when the Enter button is pressed.

4.3.1.5 Encoder/Decoder Unit

The Encoder/Decoder Unit provides the interface between the control and display panels and the IFU. Although housed in one chassis, the unit contains independent channels for failure protection.

4.3.1.6 Computer Bypass and Servo Electronic System (CBS)

The CBS consists of three independent units, which provide most of the analog electronics necessary for the operation of the control actuators. The CBS provides three basic functions: it contains the control-loop electronics for the actuators; the selection logic and basic output failure detection for the analog command signals from the digital system; and an analog link from the pilot controls to the servo electronics to serve as a backup in case of complete digital system failure.

An overall diagram of the functions performed in the CBS is shown in Figure 4-18. The CBS receives and conditions the analog servo-position commands from the digital computer system. The CBS also receives the cockpit control position sensors directly. There is a switch to allow selection of either the command signal from the digital system or the direct computer-bypass signal. The signal is then compared with the signals from the other two channels, and the middle value is automatically selected as the one that will be used to command the servos. The output of the midvalue select is then compared with

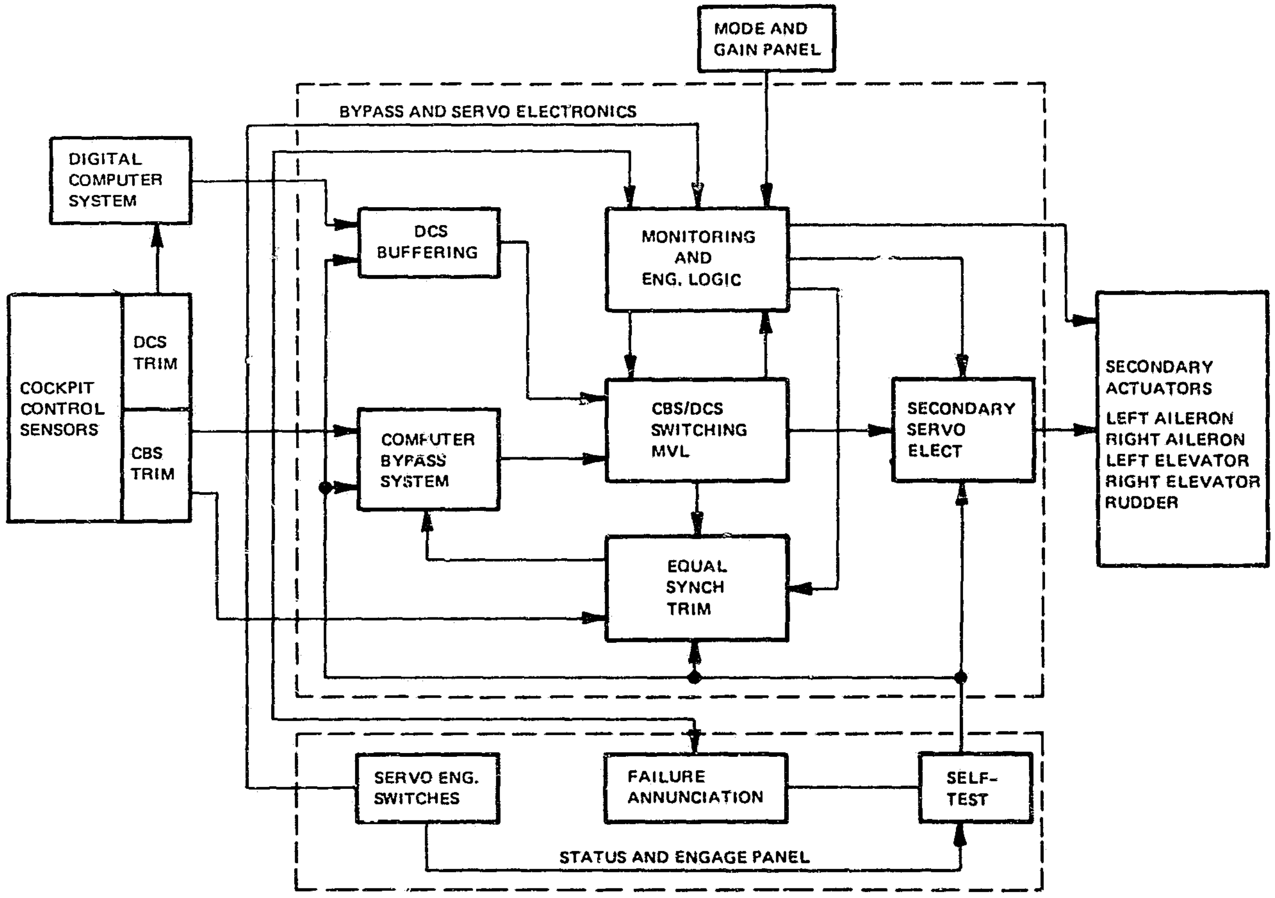


Figure 4-18. Simplified F-8 DFBW computer bypass and servo electronics system block diagram.

the original signal from that channel. A difference greater than the specified limit for longer than 40 milliseconds indicates a failure in that channel, and is used to switch off the failed channel. Figure 4-19 is a diagram of this part of the system for channel A.

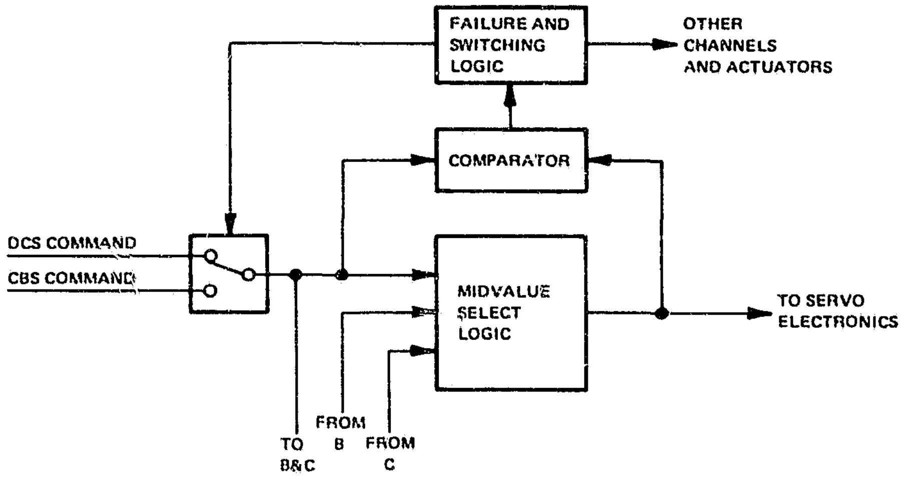


Figure 4-19. BCS switching and selection logic.

The CBS contains servo electronics, shown in Figure 4-20. The servo amplifier receives inputs from the selected midvalue position command, the actuator shaft position feedback, and a delta-pressure-equalizer signal. The delta-pressure-equalizer signal is obtained from the midvalue of the delta pressures from the three chambers of the secondary actuator. This signal minimizes the low-frequency force fight that occurs in the high-pressure gain system. This circuit acts to drive the delta pressure of each value toward the midvalue of all three. This difference between the selected midvalue delta pressure and the delta pressure for that channel is used for fault detection. This fault-detection circuit includes all system elements after the midvalue select logic, and includes the actuator itself.

If a fault is detected, the engage solenoid is disabled, supply pressure is dumped to return, and a bypass path around the piston in the failed channel is opened. Faults are annunciated in the cockpit, and reset capabilitiy is provided to the pilot. A second failure in

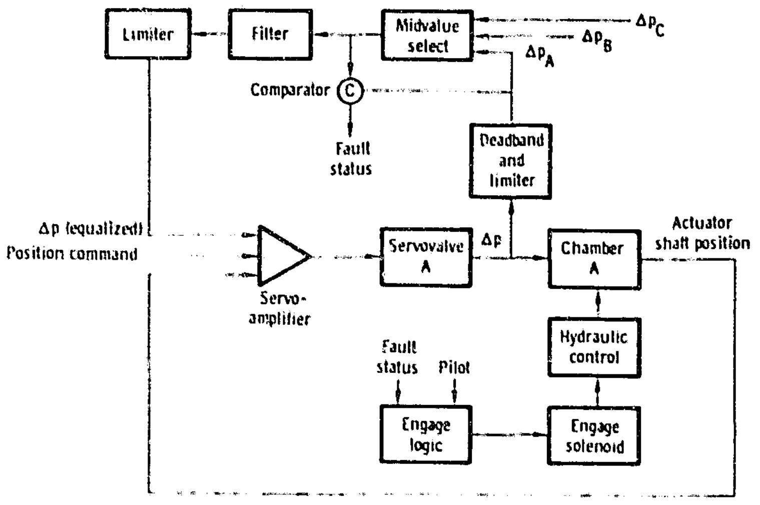


Figure 4-20. Schematic diagram of single channel of secondary actuator and servoelectronics.

the same actuator results in that actuator being turned off. Mechanical centering springs move the disabled actuator to a safe static position.

4.3.1.7 Secondary Actuator

The secondary actuator is triply redundant and is capable of providing a single fail-operational/fail-neutral force-sharing operation for any two similar nonsimultaneous failures. The actuator has three independent electrohydraulic channels. Each channel incorporates the following features and components:

(1) Independent hydraulic fluid supply.

(2) Two-stage electrohydraulic flapper nozzle servo valve to control the actuator motion.

(3) Solenoid valve to port pressurized fluid to the servo valve and to the actuator chambers.

(4) Engage valve, which by being engaged (energizing of the solenoid valve) allows the servo valve to port pressure into the actuator chambers, and by being disengaged (de-energizing of the solenoid valve) puts the actuator into a bypass mode preventing hydraulic lock.

(5) Pressure transducer for ΔP sensing, failure detection, and channel synchronization.

(6) LVDT for position output sensing and feedback-loop closure.

A schematic diagram of the servo actuator is shown in Figure 4-21, and a more detailed schematic of the electrohydraulic servo valve is shown in Figure 4-22. Table 4-5 lists the characteristics of the secondary and primary actuators.

The servo actuator has been designed to assure that all failures are detected and that no single failure will cause hydraulic lock. Any passive failure in the electrohydraulic servo valve is detected by the fact that a null bias is built into the valve's first stage. A current of 10 percent of full value is required to hold the valve at null. If there is any failure in the coil or electrical connections, the null is not held and a delta pressure is generated, which is detected in the servo electronics and causes the channel to be shut off. The system "ON" solenoid is electrically fail-safe. If there is any electrical or coil failure, the valve will shut, disabling that channel to a passive condition. If there is a mechanical failure such as a broken spring, the valve could remain open when it should be closed because of some other failure. However, the actuator will still operate, though with reduced performance, because the other two channels can overcome any irregularity in the faulted channel. Moreover, experience shows that this kind of mechanical failure is extremely rare.

Hydraulic lock is normally prevented by the engage valve. When the engage solenoid shuts off hydraulic pressure or pressure is lost for any other reason, the engage valve is moved by a spring to the bypass condition. If the spring in the engage valve breaks, the channel could be locked by the electrohydraulic servo valve also being at null. This condition is prevented because the second-stage servo valve is also spring biased; if hydraulic pressure is lost, this valve will move hard over and prevent lock.

4.3.2 Installation

The Phase II DFBW system is installed in a Navy F-8C aircraft, which is single engine and single place, and is capable of supersonic flight. It has a two-position variable incidence wing for reducing fuselage attitude during the landing approach. The modifications to the aircraft for this program were all internal; there were no basic structural

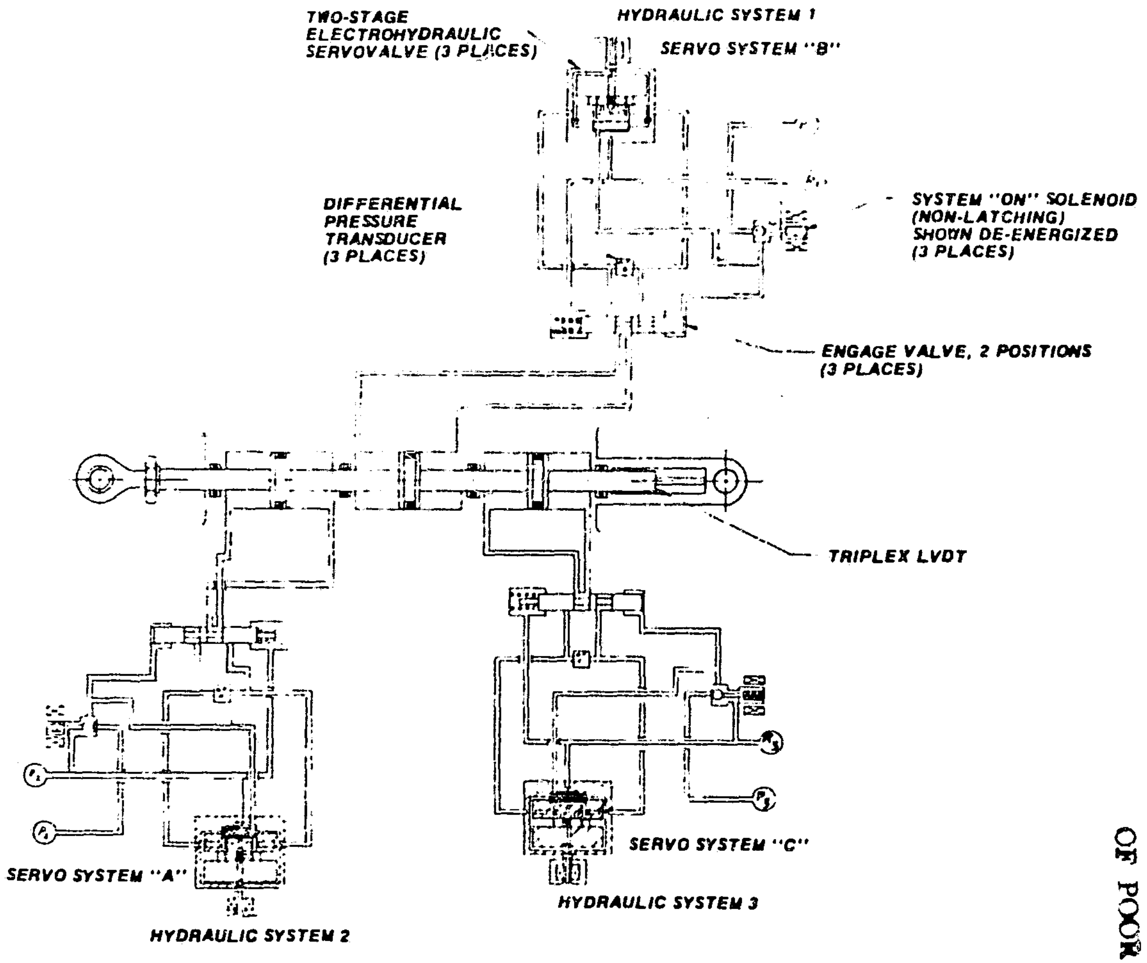

TWO-STAGE ELECTROHYDRAULIC SERVOVALVE (3 PLACES)
HYDRAULIC SYSTEM 1
SERVO SYSTEM "B"
DIFFERENTIAL PRESSURE TRANSDUCER (3 PLACES)
SYSTEM "ON" SOLENOID (NON-LATCHING) SHOWN DE-ENERGIZED (3 PLACES)
ENGAGE VALVE, 2 POSITIONS (3 PLACES)
TRIPLEX LVDT
SERVO SYSTEM "C"
HYDRAULIC SYSTEM 3
SERVO SYSTEM "A"
HYDRAULIC SYSTEM 2

Figure 4-21. Phase II hydraulic schematic, triplex redundant secondary servoactuator.

ORIGINAL PAGE IS
OF POOR QUALITY

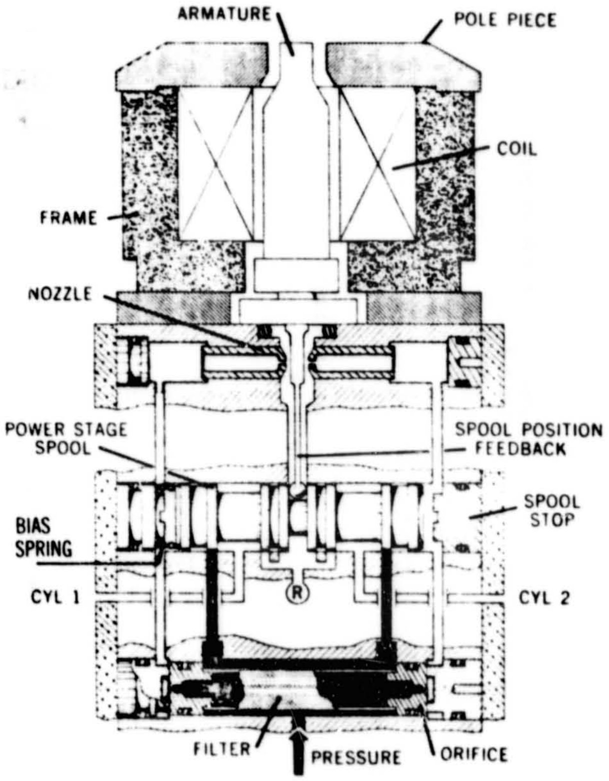


Figure 4-22. Phase II two-stage electrohydraulic servo valve.

Table 4-5. Actuator characteristics.

	Bandwidth (rad/s)	Slew Rate	Hysteresis (percent of full stroke)	Force or Torque Output	Stroke
Secondary actuators	125.0	30 cm/s	0.2	10,000 N	±2.5 cm
Power actuators					
Pitch	12.5	25 deg/s	0.1	3,000 N-m	+6.75°, -26.5°
Roll	30.0	70 deg/s	0.1	1,030 N-m	+45°, -15°
Yaw	25.0	120 deg/s	0.1	356 N-m	±21°

or aerodynamic changes. The major change was the removal of the mechanical control linkages. The only other changes were the addition of the flight control sensors, electronics, and actuators.

The location of the major elements of the system is shown in Figure 4-23. The three computers and the IFU are mounted on a pallet that was shown in Figure 4-9. The pallet is installed just behind the cockpit at the top of the fuselage. The computer bypass and servo drive electronics are mounted in the lower left side of the fuselage behind the cockpit. The locations of the control and display units are shown in Figure 4-24. The encoder/decoder unit is mounted in the nose to minimize the wire run lengths. The gyro and accelerometer sensor pallet is located near the center of gravity in the belly of the aircraft. The angle-of-attack and side-slip sensors are located on a boom in the front of the aircraft. The secondary actuators are located with the primary actuators and essentially connect to the same position as the original mechanical linkage.

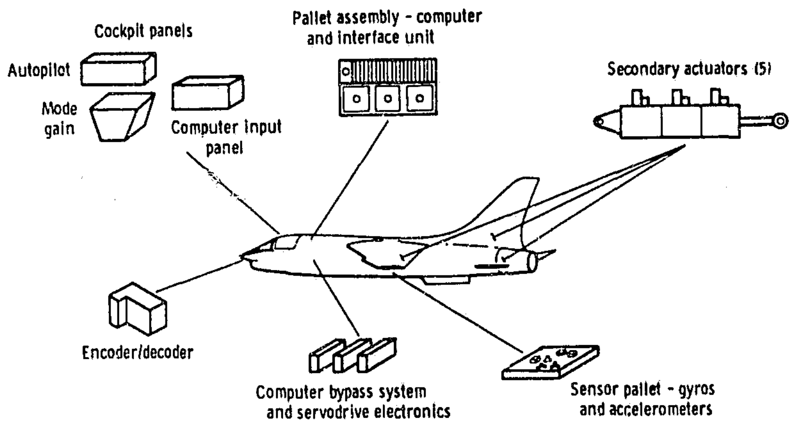

Cockpit panels
Autopilot
Mode gain
Computer input panel
Pallet assembly - computer and interface unit
Secondary actuators (5)
Encoder/decoder
Computer bypass system and servodrive electronics
Sensor pallet - gyros and accelerometers

Figure 4-23. F-8 DFBW hardware elements.

Electrical power to the flight-control system is obtained from three independent buses, which are supplied by a dedicated engine-driven dc generator. Each bus is backed up by a 40-ampere-hour battery. These batteries can power the full digital system for 60 minutes in the event of a generator failure. The batteries are isolated from each other by diodes and circuit breakers and supply power whenever their voltage exceeds the voltage

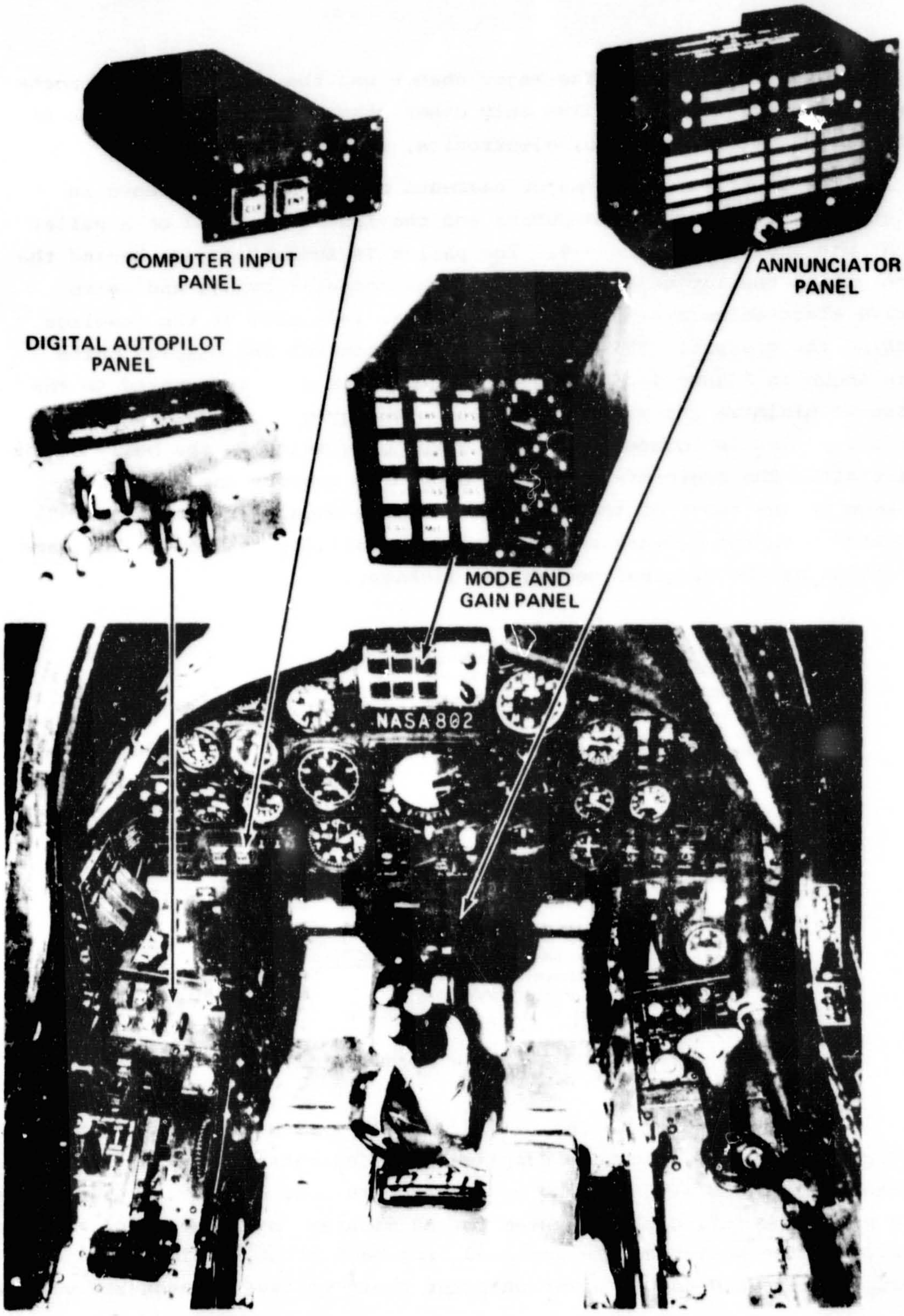


Figure 4-24. Cockpit panel assembly.

on the buses. A battery is always on an individual bus with no switching involved. The flight-control system is not connected to the existing aircraft power systems except for ac power, which is needed for the computer and pallet blowers and attitude and Mach. A ram air turbine can be deployed if necessary to supply emergency ac power for the blowers.

Three separate hydraulic power supplies are used for the three channels of the secondary actuators. Two of these are the existing system for the two channels of the primary power actuators. The third is the utility system used for landing gear, speed brakes, etc. One of the hydraulic systems can be powered in emergencies by the ram air turbine.

4.3.3 System Operation

A functional block diagram of the primary digital flight-control system in shown in Figure 4-25. The operation of the system is controlled by the software program that is executed in the digital computers. First, an outline is given of the basic structure of the software program. Then, the operation of the system is described, starting with computer fault detection and synchronization, and continuing with the basic flow of the flight-control problem from sensor data input, to control law computation, to actuator command output. Special emphasis is placed on how the system manages the redundant elements in each stage of the process.

4.3.3.1 Software Organization

The software that is stored in memory is a sequence of instructions to the computer processor. These instructions are performed sequentially to perform the primary function of the system: to produce the proper commands to the actuators controlling the aerodynamic control surfaces on the aircraft. To perform this primary function successfully and reliably, the software must perform many other supporting functions. These functions include:

(1) Maintain a synchronized mode of operation between the channels in the system.

(2) Recover operation of a channel which has sustained a momentary failure.

(3) Determine and act upon a hard failure of a channel.

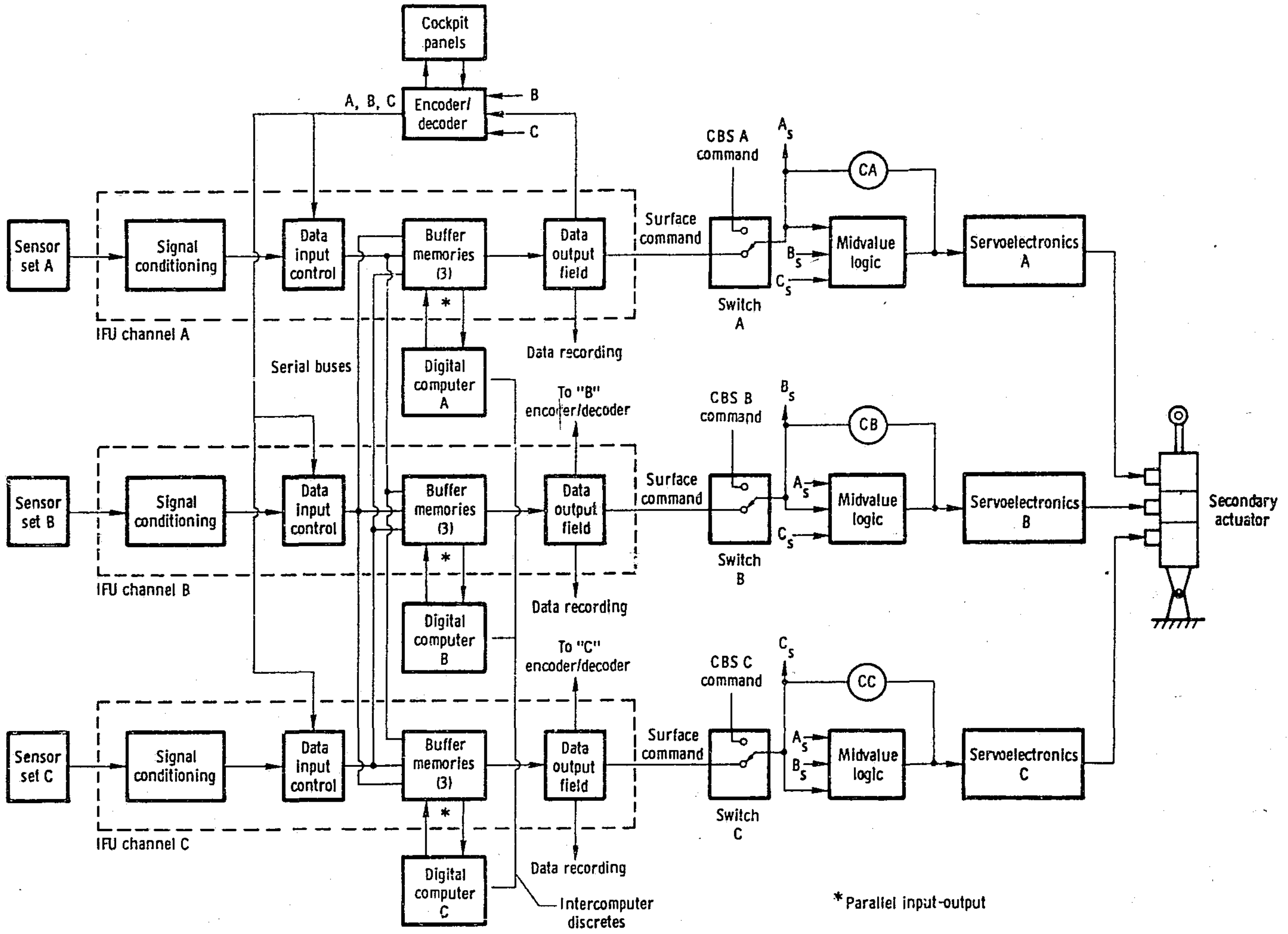


Figure 4-25. Primary digital system mechanization.

(4) Perform data selection and failure detection on input data and act accordingly.

(5) Initialize I/O functions and test for failures.

(6) Schedule and dispatch the various jobs performed by the software.

The software can be divided into four major program sections:

(1) Executive

(2) Control Law

(3) Redundancy Management

(4) Auxiliary

Each section has particular functions; together they interact and work to ensure the successful operation of the entire system—both hardware and software contained in the aircraft.

The Executive section performs program initialization, computer synchronization, intercomputer data exchange, interrupt processing, job scheduling and dispatching, downlink of data, and I/O functions.

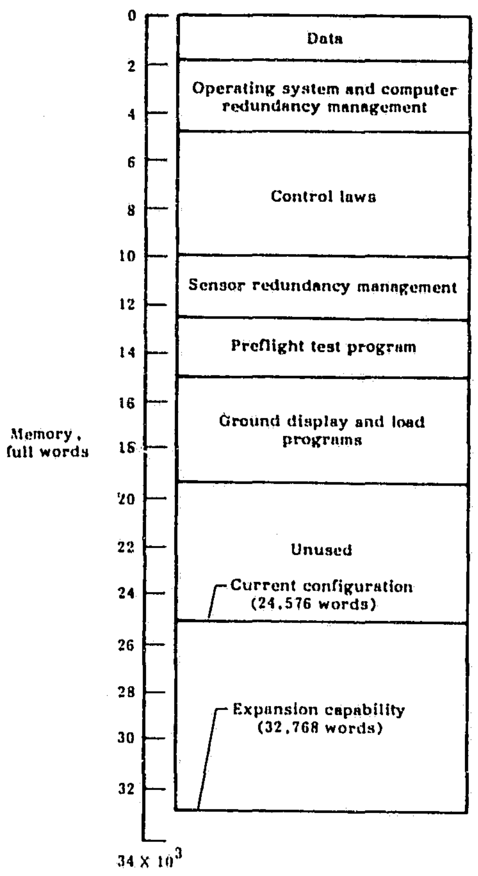
The Control Law section mechanizes various aircraft operational modes—Direct, Stability Augmentation System, Command Augmentation System, Autopilot and Remotely Augmented vehicle modes—and generates the commands to the aircraft control surfaces.

The Redundancy Management section performs input data selection, conversion, scaling, and biasing, and determination of both analog and discrete input failures.

The Auxiliary section contains the Computer Self-Test routines, the Initial Program Load routine, and the Preflight Test Program routines.

The approximate memory allocations for the major program elements are shown in Figure 4-26. The software is executed as a sequence of minor cycles, which are initiated by a timer interrupt generated within the computer, causing the program to stop doing whatever it was doing and begin executing the basic minor-loop program. The nominal time period for the minor loop is 20 milliseconds. This time can be varied for experimental purposes; however, most of the basic program functions are performed within 20 milliseconds. Some outer-loop control computations are partially performed within each minor cycle so that the whole

C-2



0
2
4
6
8
10
12
14
16
18
20
22
24
26
28
30
32
34 X 10^3
Memory, full words
Data
Operating system and computer redundancy management
Control laws
Sensor redundancy management
Preflight test program
Ground display and load programs
Unused
Current configuration (24,576 words)
Expansion capability (32,768 words)

Figure 4-26. Software memory allocation.

function is completed in an integral number of minor cycles, forming a major cycle. The sequence of execution of the program elements is given in Figure 4-27.

4.3.3.2 Computer Fault Detection and Synchronization

The first step in ensuring the proper operation of a digital-processor-based flight-control system is to assure that the computers are working properly. Once full confidence can be placed in the computers, their significant processing power can be used to monitor faults in the rest of the system, including sensors, actuators, and associated support equipment.

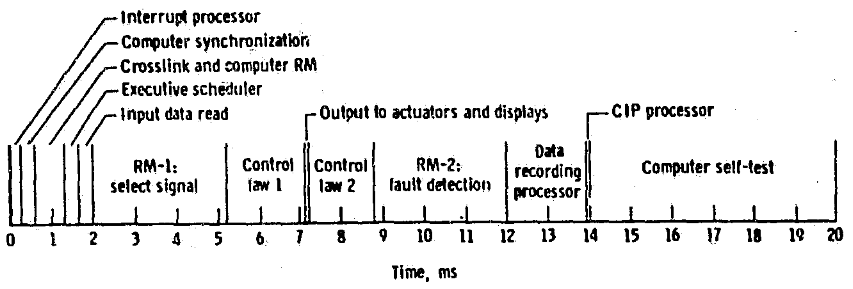


Figure 4-27. Software sequence and timing during one minor cycle: three channels, direct modes.

Fault detection can be divided into two different categories. One is self-test, where a unit performs tests to determine its own health. The other category involves monitoring or testing by other units. Fault detection in both of these categories is performed in a hierarchy of different levels. Faults are detected on as low a level as possible. However, for faults that cannot be detected on a low level and for protection against any failure of the lower level techniques, higher level tests are used. The total set of tests is designed so that each test complements the others; together, the tests are able to cover the entire system and assure the required level of system integrity.

The computers in the F-8 DFBW system are tested at several levels by both self-testing and external monitoring. These tests are performed both by special hardware and software.

Self-Test—Each computer/IFU channel determines its own condition by using hardware built-in test equipment (BITE) and by software self-test programs. The computer BITE detects faults in the execution of instructions, loss of power, parity errors in memory and failure of a go/no-go counter to be reset. The go/no-go counter is reset by a software command. Thus, this test will detect any kind of hardware or software problem that will keep the program from completing its basic computation cycle in the proper time. Problems detected by this test include the program getting caught in a continuous loop or branching to the wrong part of the memory.

The BITE in the IFU channel tests for time out, oscillator failure, and power failure. The timeout assures that the IFU performs its basic operations within certain maximum time limits.

There are two categories of response to the detection of a BITE fault. Certain faults such as loss of power are potentially transient and it is desirable to attempt to regain normal operation. This is achieved by requesting a restart, which will be discussed later. Because other faults are considered to be permanent, a signal is generated to declare permanent failure of a channel.

The other methods used by the computer for determining its own health are self-test software routines. There are two self-test routines. One routine is continually run during flight. The other routine is run only after an initial program load of the computers. This routine includes all flight tests plus certain other tests that could not be performed during normal operation without disrupting normal operation. These routines are designed to test the central processor unit and memory functions with a detection error confidence of 95 percent.

The computer/IFU channel also uses special circuits in conjunction with software programs to test the input and output interfaces. The analog command signal, certain discrete bits, and bits within the serial data words going to the encoder/decoder are wired back into the computer. Software routines in the computer check these wraparound inputs and compare them with what the output should have been. This test checks the critical output interface hardware and a majority of the input interface hardware.

Synchronization—The hardware and software self-test monitors in each computer/IFU channel will detect the majority of all possible failures, and, if a failure is detected, will produce a signal causing that channel to be disregarded. It is now possible to connect the three computers together in order for them to monitor each other to detect any fault that might not be caught by the hardware and software self-test. The first step necessary to allow the computers in the F-8 DFBW system to operate together is to synchronize their operations so that they are performing the same operations at very nearly the same time. Synchronization is necessary to cause data to be read from the sensors at the same time to allow fault detection by comparing redundant sensors. The synchronization of computations is also important to allow comparison of the analog command outputs from the computer. This comparison is performed in the analog electronics within the CBS. Another important task performed by the synchronization is computer fault detection. One of the best ways for each computer to determine the health of the other two is by their ability to come into synchronization. The computers are synchronized by a program in each computer, which sends discrete signals to each of the other two computers (Figure 4-28). The computers become synchronized when the program reads and verifies the discretes it has received from the other computers. If, after a short wait to allow for skew between processors, one computer fails to synchronize with the other two, the two remaining computers exit the sync program and continue normal processing. This synchronization only occurs at the beginning of each minor cycle. These 20-millisecond cycles are begun within 10 to 50 microseconds of each other.

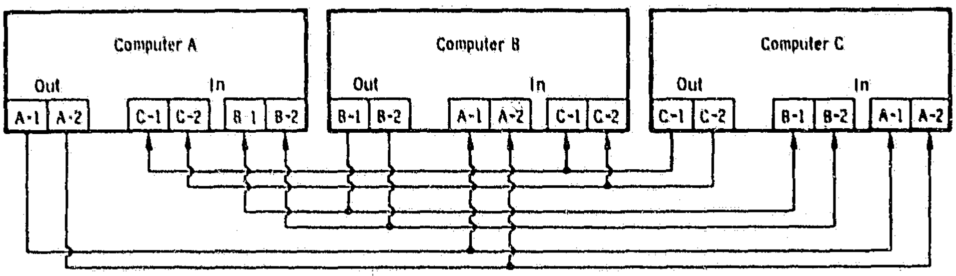


Figure 4-28. Synchronization discretes.

Cross-Channel Monitoring—If the computers can be successfully synchronized, the next step in their monitoring of each other is to transfer data among themselves. This transfer of data is done through the buffer memories in each IFU channel, and consists of six data words. The transferred data includes an identification of the computer channel, the computer's minor cycle count, the mode it is in, and its assessment of the failure status of the other two computers. Failure to receive data from a synchronized computer for 10 successive cycles results in a "hard fail" declaration and that computer cannot be used again. Whereas failure to receive data from a nonsynchronized computer results in the declaration of a "soft fail" and enables inclusion of this computer in the operating set when its sync discretes and data appears. If a computer's data is not properly identified for 10 successive cycles a hard fail is declared. If a computer's cycle count and mode does not agree with the other two computers, it requests a restart.

Restart—Restart is requested by a computer for a number of reasons including:

(1) Freshstart - initial power up.

(2) Power disruption.

(3) Crosslink fail—I/O or data.

(4) Software program and/or computer BITE detected errors.

Whenever a restart is requested, the three computers, by way of the crosslink, exchange enough data to guarantee that they are in agreement. This transmission includes the choice of the computer considered to have valid data and the data to be used by the offending computer. The data exchanged is 94 16-bit words, and includes such items as sensor and discrete failure history and control-law parameters.

To prevent continuous restart requests caused by a failure in the system, each computer maintains a count of all restart requests. If the number of restart requests made by any computer exceeds a prescribed tolerance, that computer is declared hard failed and its requests are subsequently ignored. The entire restart process takes approximately 8 milliseconds from recognition of request to resynchronization.

Failure Voting—It is necessary for two computers to agree before a third can be declared failed. If the self-test software or hardware in a channel declares itself failed, it is inhibited from voting on the other computers. A logical diagram of the hardware within an IFU channel that implements this process is shown in Figure 4-29. The output signal which declares a channel failed is sent to the BCS and causes that channel to switch to the analog channel.

Command Signal Voting in CBS—The final level of fault detection for the digital system takes place within the analog Computer Bypass System (CBS). As described in Section 4.3.1.6, the CBS selects the midvalue of the control surface commands generated by each of the three computer/IFU channels for each control surface. This selection technique guarantees that if there is any fault that would have a dangerous result, it would not affect the output. The CBS also compares the selected midvalue with the corresponding command from the individual channel. If the difference is greater than the allowable tolerance, that particular channel is switched to the analog signal for that channel. The analog signal is synchronized to follow the midvalue output, which is determined by the other two digital channels that are

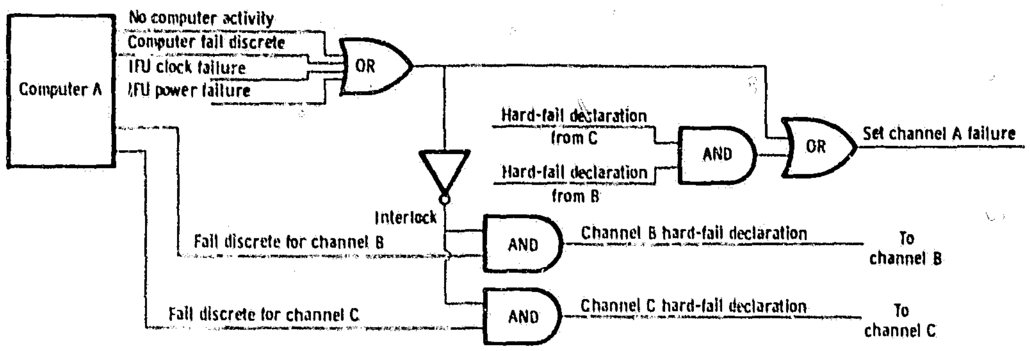

Computer A
No computer activity
Computer fail discrete
IFU clock failure
IFU power failure
OR
Hard-fail declaration from C
Hard-fail declaration from B
AND
OR
Set channel A failure
Interlock
Fail discrete for channel B
AND
Channel B hard-fail declaration
To channel B
Fail discrete for channel C
AND
Channel C hard-fail declaration
To channel C

Figure 4-29. Functional diagram of fault detection hardware in IFU (shown for channel A).

still operating. If there is a second failure of a digital channel, all channels are switched to the analog signal and synchronization is stopped.

With these multiple levels of computer self-test and external monitoring, the probability that a computer failure will be undetected is extremely small. With confidence thus established in the computers, they can proceed with their primary task: control of the aircraft. The first step in the flight-control process is to obtain the necessary input signal from the sensors and assure that no faulty input is used.

4.3.3.3 Sensor Data Processing and Redundancy Management

The data from all redundant sensors is available to each computer. The data is processed by redundancy management (RM) programs, which are performed in two phases. The only function of the first phase is to obtain the best estimate of the actual parameter value based on the available multiple sensor inputs, and provide this data for use in the control-law computations. The second phase performs fault detection and identification, and controls the reconfiguration of the select logic used in the first phase.

A typical triplex RM algorithm is shown in Figure 4-30. The first phase begins as a midvalue select mode, changes to an averaging algorithm after the first hard failure, and finally degrades to a default output value after the second failure. A hard-sensor fault is declared by the second phase when a sensor differs from the selected value by an amount greater than the allowable tolerance for a given number (N) of consecutive passes. Failure-status logic monitors the results of the tracking test, and, through hard-fail flags, causes the mode or function using that sensor to be inhibited. For example, should the entire roll-rate-gyro set be lost, roll stability augmentation system (SAS) would be inhibited. In some cases, annunciation is given to the pilot when an entire sensor set has been lost. The first failures of sensors in a triplex set are not annunciated to the pilot.

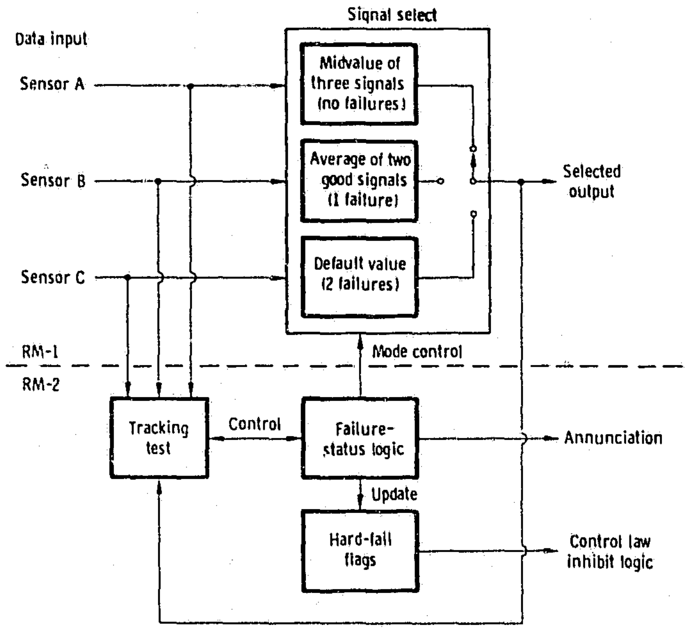


Figure 4-30. Triplex analog sensor redundancy management algorithm.

4.3.3.4 Control Laws

The control laws mechanized for the F-8 DFBW aircraft were selected to include several functions projected for use in the future active control applications that would required full-time full-authority control. The pitch, roll, and yaw axes have multiple pilot-selectable modes. Each axis contains a DIRECT mode, which provides unaugmented control of the aircraft as shown in Figure 4-31 for the pitch axis. In this mode, the surface command is the sum of stick and trim components. A pitch SAS mode uses scheduled pitch rate feedback to improve short period damping as shown in Figure 4-32. These modes provide less complex configurations in the event of failures in certain sensors in the more highly augmented pitch command augmentation system (CAS) and lateral-directional SAS modes which contain the active control functions.

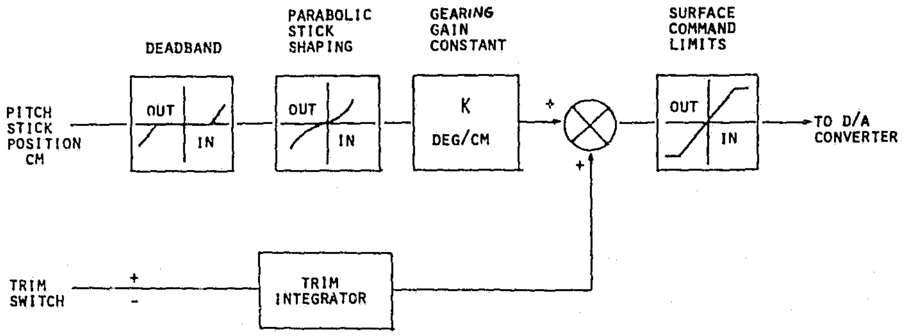


Figure 4-31. Pitch DIRECT mode.

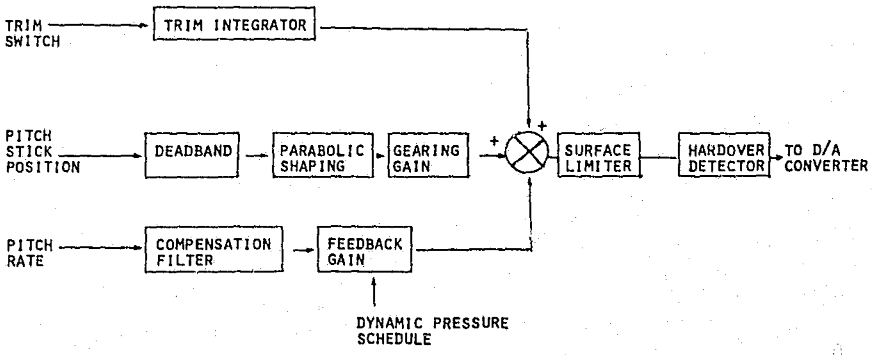


Figure 4-32. Pitch SAS mode.

All inner loop control law functions are computed at the minor cycle rate of 50 samples per second. Autopilot functions, scheduled gain updating, and other less time-critical operations are computed at the major cycle rate of 12.5 samples per second.

Pitch CAS Control Law—The pitch CAS mode includes several typical active control functions. If a new airplane were being designed, such control laws would be considered in the initial trade studies leading to the ultimate configuration. No structural or aerodynamic changes were made to the F-8C aircraft in conjunction with these control laws, however; therefore, it is not possible to extract performance tradeoff information directly from the F-8 DFBW flight-test results. The control-law research objective of the F-8 DFBW program is to evaluate the mutual interactions of the control functions in a full-authority flight-critical digital implementation.

Figure 4-33 illustrates the command augmentation functions implemented in the CAS mode. The basic controller combines a prefiltered stick deflection with pitch rate and normal acceleration. The resulting signal is routed to the actuation system by way of a variable gain (K_{C*}), which is scheduled with the dynamic pressure derived from altitude and Mach number. To minimize excessive stick forces during large changes in air speed, neutral speed stability is provided by an effective forward-loop integration. The integration is mechanized by the cancellation of the position feedback signal of the secondary actuators at low frequencies.

An active flap system provides both ride-smoothing and maneuver-flap functions. The ride-smoothing system reduces the rigid-body normal acceleration response due to turbulence, and the maneuver-flap system positions the flaps to provide the optimum lift-to-drag-ratio maneuvering flight.

The autopilot includes attitude, altitude, Mach, and heading-hold functions, an auto turn mode, and control-stick steering. An angle-of-attack limiter keeps the aircraft from exceeding a preselected critical angle of attack by always selecting the most favorable signal from either the normal controller or the α controller.

Lateral-Directional Control Laws—The lateral-directional control system contains independently selectable roll and yaw stability augmentation systems which are designed to operate together. The design goals for this system were to improve Dutch roll damping, and to provide good high angle-of-attack turn coordination. The lateral-directional control system is illustrated functionally in Figure 4-34. The $K(\alpha)$ designations represent functions which are scheduled with angle of attack.

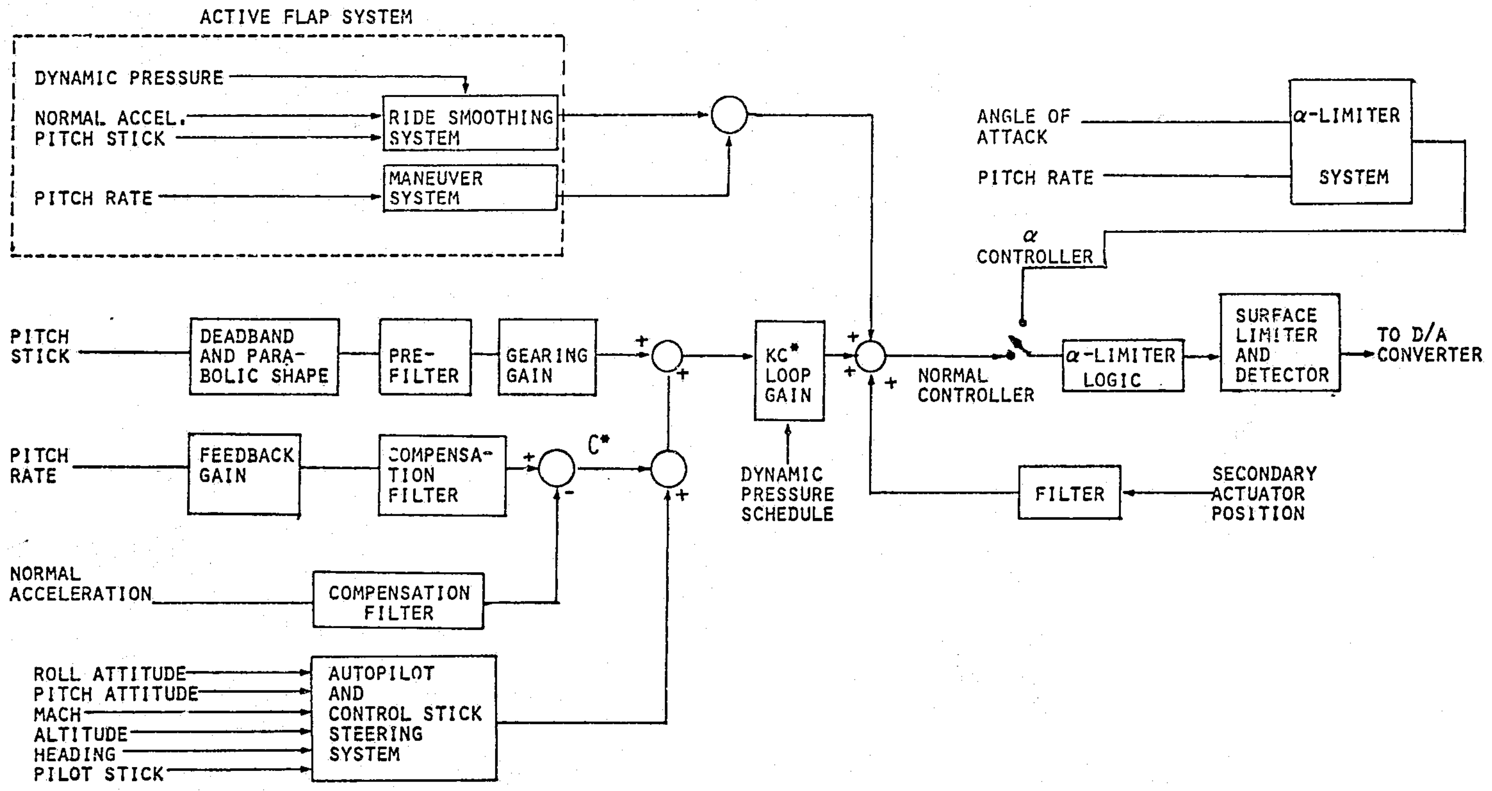


Figure 4-33. Pitch CAS system.

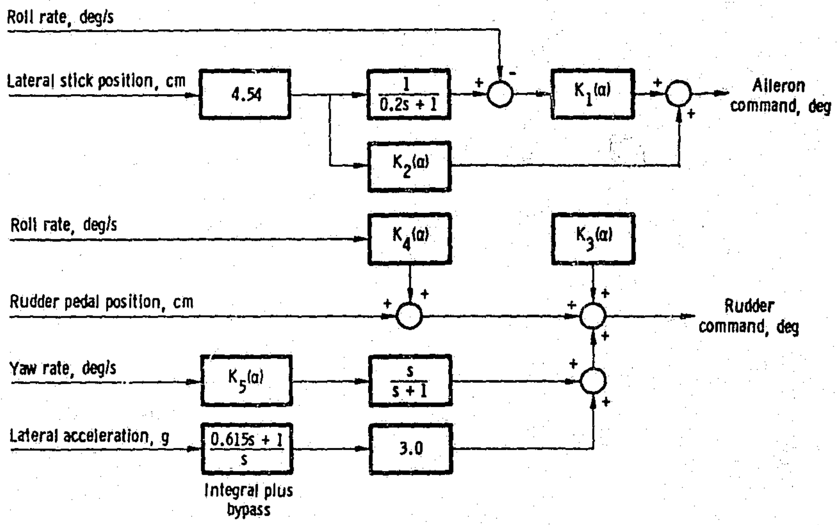


Figure 4-34. Lateral-directional control system.

The aileron command is composed of filtered stick position (previously shaped) and roll-rate feedback. The rudder command is composed of roll rate, yaw rate, and lateral acceleration in proportions which approximate sideslip rate. Rudder pedal commands and an interconnect command from the roll stick make up the pilot inputs. A small amount of integral feedback is used on the lateral acceleration signal to provide automatic directional trim in cruise flight, thus relieving the pilot of the necessity to trim the rudder.

Remotely Augmented Vehicle Mode—In addition to the control laws that are implemented within the airborne system, the F-8 DFBW system has a special remotely augmented vehicle (RAV) mode. This mode allows highly speculative and advanced control concepts to be programmed in a ground-based computer. The necessary data is transmitted to the ground and the computed commands are transmitted back to the aircraft as shown in Figure 4-35. The RAV mode is pilot selectable by axis. If there is any failure, the system automatically switches or can be manually switched back to the normal SAS mode. This capability allows the easy implementation and change of experimental control laws without disrupting the integrity of the flight system.

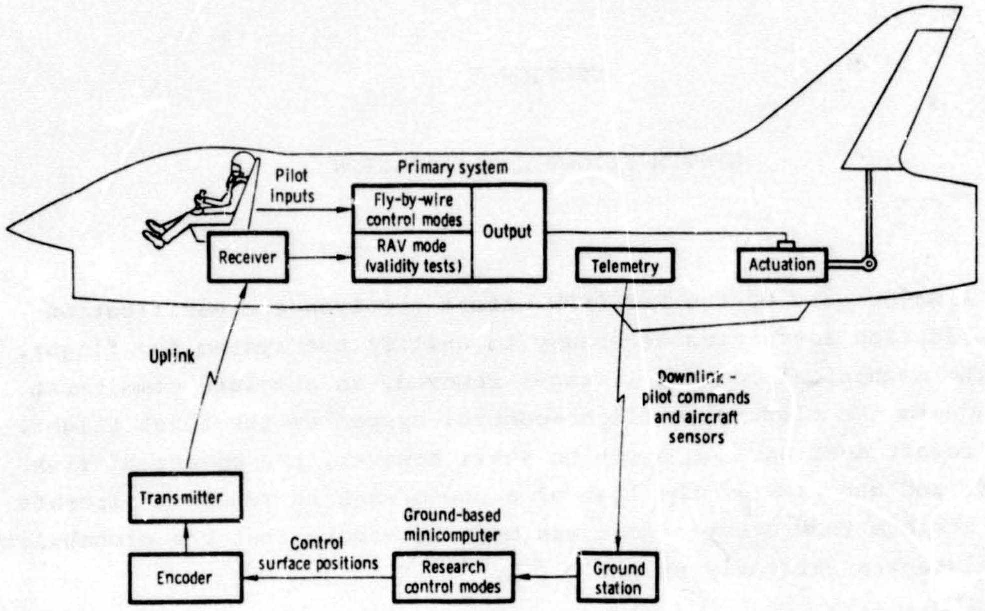

Primary system
Pilot inputs
Fly-by-wire control modes
RAV mode (validity tests)
Output
Receiver
Telemetry
Actuation
Uplink
Downlink - pilot commands and aircraft sensors
Transmitter
Ground-based minicomputer
Control surface positions
Encoder
Research control modes
Ground station

Figure 4-35. F-8 DFBW remote augmentation vehicle (RAV) concept.

SECTION 5

SYSTEM FLIGHT QUALIFICATION

A major part of the F-8 DFBW effort involved the verification and validation activities necessary to qualify the system for flight. With the mechanical control linkages removed, an absolute commitment was made to the electronic flight-control system on the first flight. The aircraft does have an ejection seat; however, the potential risk of life and the risk of the loss of a one-of-a-kind research aircraft is so serious that every effort was made to assure that the probability of failure was extremely small.

The qualification program for the F-8 DFBW can be roughly divided into three stages. The first stage is design validation. In this stage, analysis is performed to determine if the design itself meets all requirements, assuming the hardware is built as designed and has the normally expected component failure rates. The next stage is the laboratory testing of the system's component parts to assure that they are actually built as designed and meet all operational and environmental requirements. This stage includes both hardware and software. The final stage is the testing of the total integrated system. This testing takes place in the aircraft simulator (iron bird) or in the aircraft itself and involves all elements of the system, including the special flight-control hardware and software, sensors, actuators, electrical, and hydraulic power systems. The ultimate tests are the flight tests.

In the following sections of the report the activities performed in each stage of the system qualification process are described. Special sections will emphasize the software and systems-level test facilities, software verification, and the pilot's role in test and evaluation.

5.1 Design Validation

The first stage of the process of assuring that the F-8 DFBW system was qualified for flight was to validate the basic design of the system. Four of the steps done to validate the design will be discussed here. These include the theoretical reliability analysis, electronic design verification, failure modes and effects analysis, and sneak-circuit analysis. The software validation will be discussed in a separate section.

5.1.1 Theoretical Reliability Analysis

One of the first steps in validating the design of a flight critical system is to determine the approximate theoretical reliability of the system. This analysis is necessary to assure that the basic configuration chosen for the system has the inherent capability of meeting the reliability requirements for the functions to be performed. This reliability analysis is performed by obtaining the approximate reliability of each of the system's major component parts. These estimates are based on analysis and experience of the actual unit if it exists. If it is a new component, approximations are made based on the preliminary design of the unit and on experience with similar equipment. The fundamental statistical and reliability equations are applied to the particular system configuration being studied. These equations give the probability of system failure in performing the required operations as a function of the failure rates of the component units. The equations take into consideration the number of redundant channels, the number of channels needed by the redundancy management scheme, the reliability of the self-test, and other factors affecting the probability of total failure. The computed probability of failure is compared with the requirements for the system to determine the adequacy of the system configuration and associated hardware.

Theoretical reliablity analysis performs one of its most valuable roles in the configuration tradeoff studies. It becomes a major factor in determining the number of channels required, the methods used for redundancy management and failure detection, and in many other system design considerations. Once a configuration is chosen, this analysis gives the reliability requirements for the component parts of the system. It is particularly valuable in determining the sensitivity of the total probability of failure to the reliability of critical parts. This process thus identifies where particular care must be exercised in the design, verification, and validation of the system.

Theoretical reliability analysis can also be useful after the design is complete by being performed on the actual hardware that is built. This analysis would use the detailed system configuration implemented to give an updated estimate of the reliability actually expected for the system; it gives the baseline numbers against which the test results and actual operation of the system can be compared.

There were two early reliability analyses that contributed to the design of the F-8 DFBW system. These will be discussed briefly with some of the typical results shown.

5.1.1.1 Configuration Trade Study(18)

The first study was a preliminary design analysis of the F-8 DFBW system. It was not a direct part of the design effort, but was an early study in support of the program. This study covers a wide range of problems associated with the system, with particular attention paid to the secondary actuators. Of interest here is the reliability analysis of several configurations options. Preliminary analysis was performed on a broad range of configurations, and more detailed analysis on configurations that were most promising for particular applications.

The preliminary analysis was performed on a number of different configurations. The simplified digital and analog backup channels and typical failure rates used in the analysis are shown in Figure 5-1. Assumptions made for this analysis are:

(1) Failure rate of a digital channel in the range of 10^{-3} to 10^{-4} failures per hour.

(2) Failure rate of a three-axis analog channel of 4.5×10^{-4} failures per hour.

(3) Failures of the actuation function and electrical or hydraulic power not included.

(4) Mission time of 1 hour.

(5) Perfect comparison monitoring and switching.

(6) Operation required for two of three analog channels.

The results of this preliminary analysis are shown in Figure 5-2. The study shows the failure-rate probabilities for various combinations

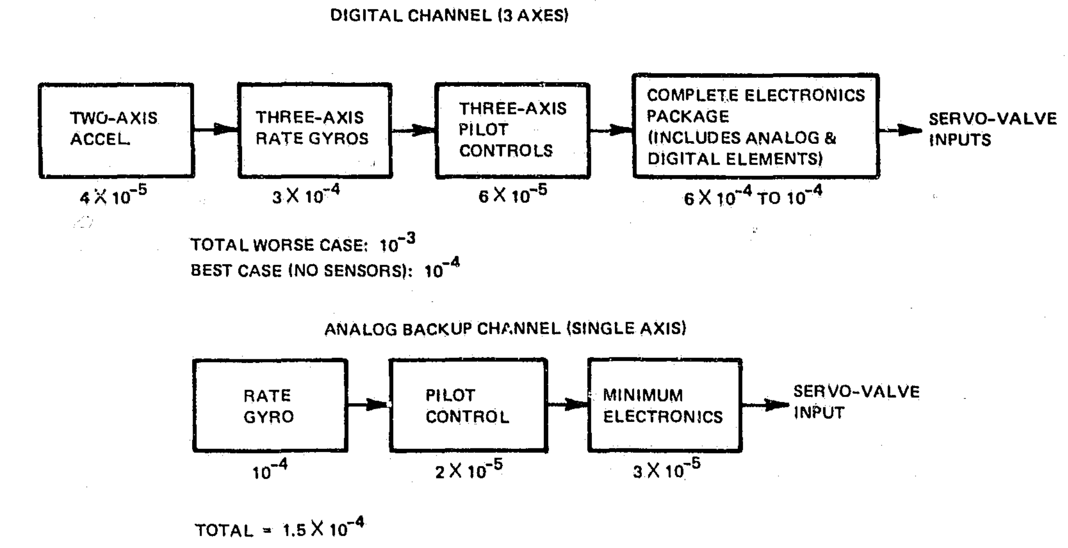
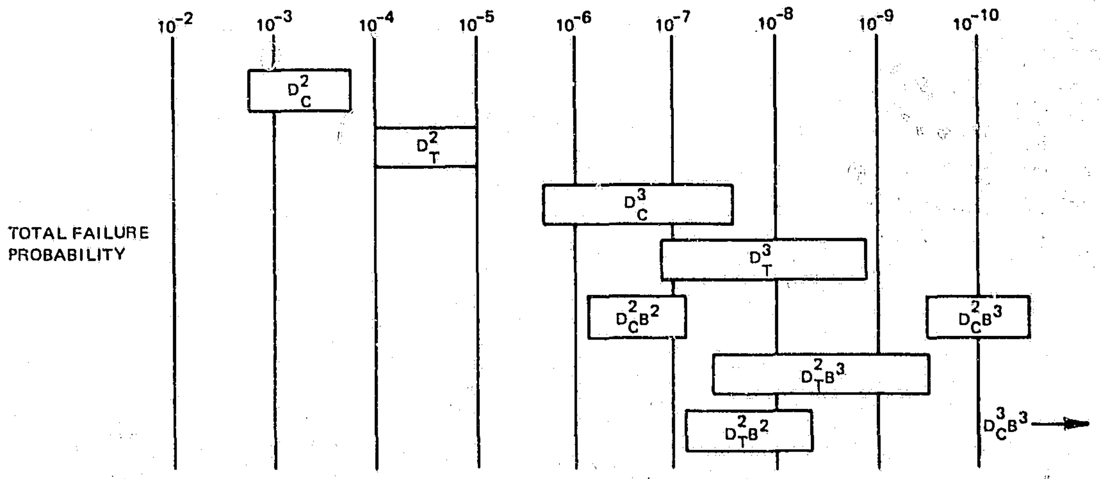


Figure 5-1. Control channel models (failure rates per hour).



NOTE:

D_C = DIGITAL CHANNEL WITH COMPARISON MONITORING

D_T = DIGITAL CHANNEL WITH 95% SELF-TEST

B = BACKUP ANALOG CHANNEL

Figure 5-2. Probability of total system failure for different configurations (from Reference 18).

of digital and analog channels. The failure monitoring for the digital channels is either comparison voting or voting with self-test capable of detecting 95 percent of all computational faults. It can be seen that this analysis shows the need for at least two digital channels and three backup channels to meet a 10^{-9} probability of failure. The simplified analysis for the configuration actually used in the F-8 DFBW (three digital and three analog channels) shows a higher theoretical reliability than is correct for the actual system. In the actual system, much of the servo analog electronics is common both to digital and analog channels. Failure rates in these common circuits must be added to failure rates for the independent parallel analog and digital channels. Using the same assumptions as in the other cases, the reliability prediction for the F-8 DFBW configuration would be approximately 10^{-10}.

More detailed reliability analysis was performed in this study on the configurations that were most promising. Only one part of the study will be given here to illustrate some of the analysis that was performed. This analysis shows that care must be exercised in the use of self-test. It can be seen in Figure 5-2 that self-test usually decreases the probability of failure. However, in one case, self-test actually increases the probability of failure. In the configuration with dual digital channels and triplex analog channels, if one digital channel fails and the failure is detected by self-test, the system can continue to operate on the other digital channel. However, if this channel fails and is not detected, the transfer to the analog backup channels is not made. The percent of self-test thus has to be very high before there is a net increase in reliability. The probability of total system failure as a function of percent self-test is shown in Figure 5-3. This plot shows that adding any self-test immediately reduces reliability with a minimum at 50 percent self-test. Reliability is not increased until the self-test in greater than 99.2 percent. This disadvantage of self-test is not generally true for most configurations. However, this study does show that the efforts of self-test must be carefully studied for the particular configuration to be sure the real effects are known.

5.1.1.2 F-8 DFBW Configuration Reliability Analysis

This theoretical reliability analysis was performed early in the program. This analysis gave initial estimates of the reliability

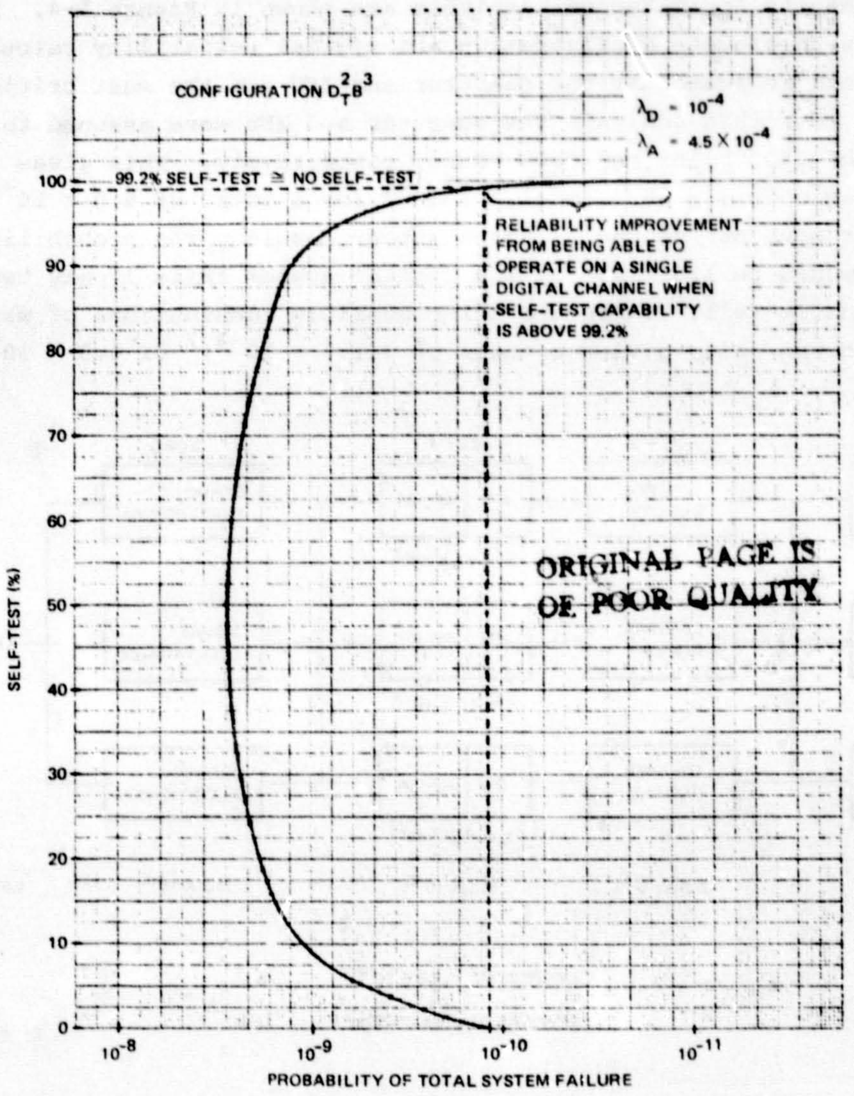


Figure 5-3. Reliability variation with self-test capability.

of the digital system and was used to study various configuration alternatives. In this study, the basic principles of probability and reliability were applied to the system. Equations were derived to give the probability that the digital system would fail and cause the system to revert to the analog backup system. These equations are a function of the probability of failure of all the different components and of the component organization.

The results for a typical analysis are shown in Figure 5-4. This figure shows an example configuration and typical reliability calculations. It can be seen that the computer and IFU are the most critical components. For this analysis the computer and IFU were assumed to have MTBFs of 3000 hours and 6000 hours, respectively. This gives failure rates of 3.3×10^{-4} and 1.7×10^{-4} for a total of 5.0×10^{-4} failures per hour per channel. For a 3-hour mission, the probability of failure would be 1.5×10^{-3}. The digital system fails if any two digital channels fail. There are three possible combinations of ways two channels can fail, giving a total of $3(1.5 \times 10^{-3})^2$ or 6.8×10^{-6}.

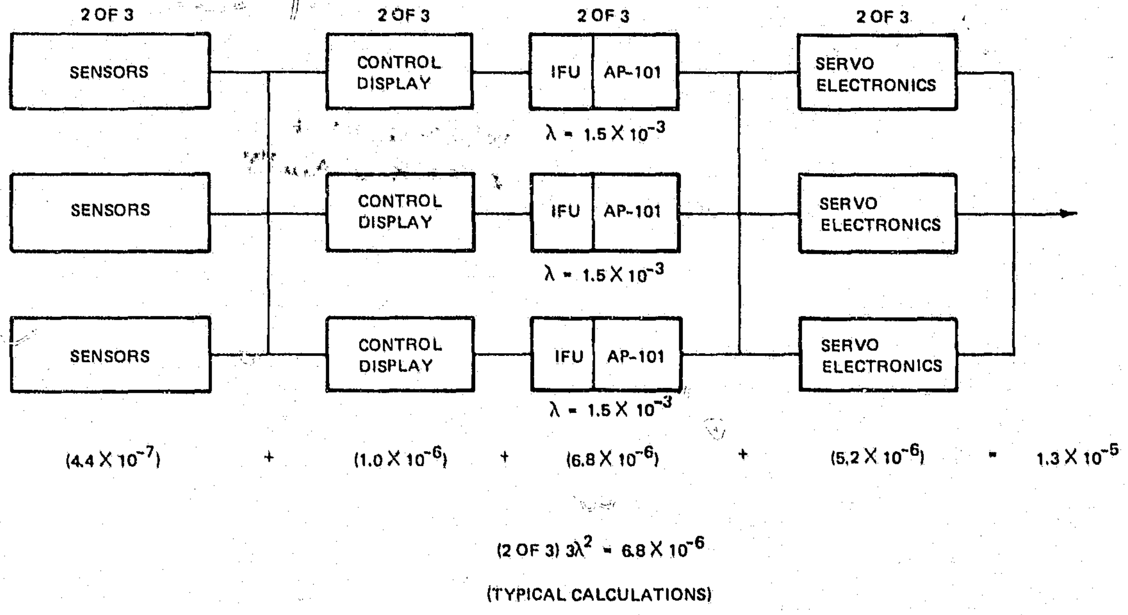


Figure 5-4. Typical results from the reliability analysis of Reference 19: probability of digital system failure for a 3-hour mission.

The analysis was performed for several different configuration options. A comparison of the results was helpful in determining the sensitivity of the reliability of the total system to the different configurations and the reliability of the critical components.

A typical analysis giving sensor interface options is shown in Figure 5-5. The different sensor computer configurations considered here are:

(1) Each triplex sensor connected to one of the IFU channels with the data transferred to all computers through the IFU buffer memories.

(2) Each sensor cross-strapped into two IFU channels.

(3) Two analog-to-digital converters used in each IFU.

Failure rates were assumed for the different components comprising the system. For these numbers, the sensor failure for the critical pilot inputs were:

(1) 4.4×10^{-7} for single converters.

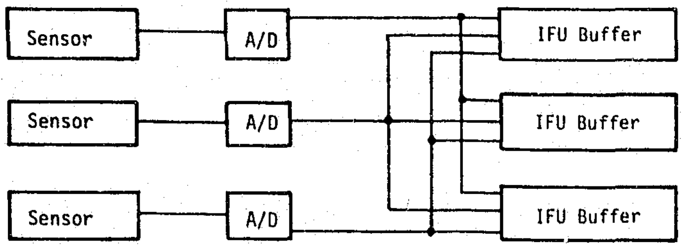
(2) 2.8×10^{-7} for cross-strapped converters.

(3) 0.8×10^{-8} for dual converters.

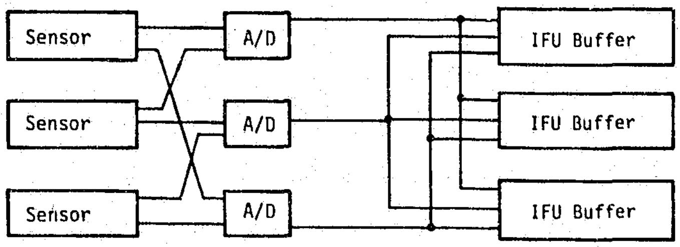
There is some improvement for cross-strapping, but it is not significant. In this analysis, the converter channel was assumed to be the same for each configuration. If the reliability of the converter channel were reduced 26 percent because of the added signal channels, the gain for cross-strapping would be wiped out. The use of two analog-to-digital converters reduced the probability for input signal failure to essentially the failure rate of the sensor itself. However, considerably more hardware is added which may not be justified since this term is small relative to the failure rate of the computers. This example shows how theoretical reliability analysis can be used to help determine the best configuration; once a configuration is chosen, the analysis shows if the system has the theoretical capability of meeting the reliability requirements assuming the components have the expected failure rates and there is no error in implementing the design.

5.1.2 Electronic Design Verification

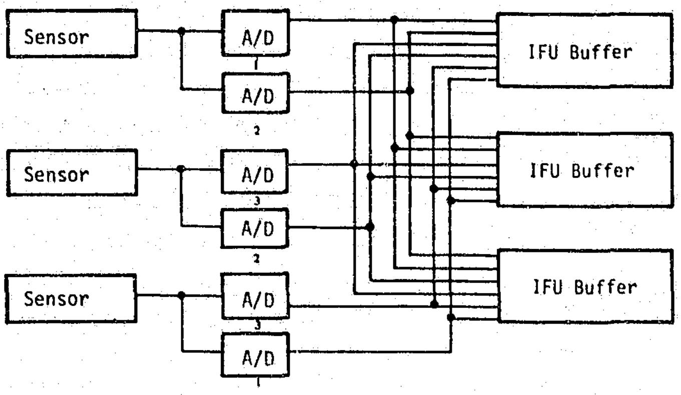
A further step in the qualification of the F-8 DFBW system was to verify that there was no error in the design of the system. Once it has been assured that the basic system has theoretical capability of meeting the reliability requirements, it must be assured that the detailed design meets the system requirements with no design faults.



(a) One converter for each set of sensors



(b) Each sensor set is cross-strapped to two converters



(c) Two converters for each sensor set

Figure 5-5. Sensor-converter configurations.

In a digital system, both the hardware design and the software design are critical. The hardware design verification process will be described briefly here. The software verification will be described in a later section.

The major units that were designed specifically for the F-8 DFBW system were the IFU and the Encoder/Decoder. The computer was designed by IBM for more general use and will not be discussed here.

The first step in assuring that there are no faults is for the design engineer to generate a careful original design. However, to protect against any misunderstanding of requirements or human error, an independent review was made. For the F-8 DFBW units, each circuit design was given to a design engineer with competence equal to the one doing the original design. This engineer reviewed the design in relation to the requirements for that design. A design review was then held among the reviewing designer, the original designer, and the design supervisor to resolve any discrepancies.

Another aspect of design verification is to assure that the proper parts are used. The parts were reviewed by the reliability staff to assure that they were adequate for the application. Where possible, parts are used with a proven reliability history. However, in an advanced development program there is a desire to use components that are as close to the state-of-the-art as possible. In these cases, the proven history may not be as great as would be desired. Experience on devices is monitored through industry and government-reliability exchange programs to help assess the risk. Any device that has shown a problem is watched carefully.

Of course the design is continually verified through each succeeding stage of the development program. The next test after the verification of the paper design is to build and test the breadboard prototype circuits. Design verification and validation continues implicitly through all of the hardware and systems testing that is described in the following sections.

5.1.3 Failure Modes and Effects Analysis

Another important part of the analysis performed to validate the design of the F-8 DFBW system was Failure Modes and Effects Analysis (FMEA). There were two primary FMEAs performed on the system: one was performed on the digital system and the other on the Computer Bypass System (CBS). An FMEA was also performed on the secondary hydraulic actuator.

The FMEA on the digital system was performed to assure that the design objective was met. The objective was to continue digital-system operation after any single failure and to switch to the CBS system after any other failure that does not allow safe digital operation.

This analysis was not performed to the detailed component level, but was a high-level analysis covering all units and signals in the system. Single-failure analysis was performed for:

(1) Input failures.

(2) Output failures including signals to actuators and pilot displays.

(3) Interchannel communications failures.

(4) Engage logic failures.

(5) Power failures.

There was also analysis of related dual failures. Multiple failures in only one channel were not analyzed since these failures could, at worst, cause the loss of one channel. Multiple failures which were not related such that their combined effects could be obtained from the individual single failure tables also were not considered. Only dual failures in more than one channel were considered such as:

(1) Dual serial I/O failures.

(2) Dual input failures.

(3) Dual output failures.

(4) Dual interchannel communications failures.

(5) Dual power failures.

To the level of two failures, the digital system design objectives are realized. The software cannot be examined for generic failures with 100 percent confidence. Within this limitation, the FMEA did not reveal a flight-critical single-point failure. All flight-critical situations resulting from two failures engage the CBS.

A more detailed analysis was performed on the CBS system. This analysis was in three parts:

(1) Analog portion (39 pages).

(2) Logic portion (37 pages).

(3) Status/Engage Panel (4 pages).

Only "hard over" and "passive" type failures were considered in the analysis. The probability of oscillatory failures is low in this type design, and, if significant in magnitude, would be voted out by the natural voting process. Only single or first failures were considered. However, when necessary and pertinent, particularly in the logic, the impact of certain multiple failures has been considered as well.

The results of the FMEA showed that there are no system first failures that will have any significant impact on either system performance or system monitoring capability.

5.1.4 <u>Sneak-Circuit Analysis</u>(20)

A Sneak-Circuit Analysis was performed on the F-8 DFBW system hardware. This analysis technique was developed to detect latent circuit conditions that may cause unwanted functions or inhibit desired functions. These sneak circuits can cause system failure that is not the result of a component failure. There may be four basic types of sneak circuits:

(1) Sneak Paths, which cause current or energy to flow along an unexpected route.

(2) Sneak Timing, which may cause or prevent the flow of current or energy to activate or inhibit a function at an unexpected time.

(3) Sneak Indications, which may cause an ambiguous or false display of system operating conditions.

(4) Sneak Labels, which may cause incorrect stimuli to be initiated through operator error.

The data used to perform this analysis is electrical continuity information based on wire run lists and electrical schematics. Support data in the form of assembly drawings, overall system lever interconnect diagrams, and electrical schematics describing the interface of out-of-scope components was also used for the analysis.

The data was coded for input into the Automated Sneak Programs used for Apollo and Skylab Sneak-Circuit Analysis. The computer program searches for continuity path and generates reports from which network trees can be drawn. These network trees were drawn in a topological form which facilitates the application of sneak circuit clues. These clues aid in the detection of the four basic types of sneaks defined earlier.

The outputs of the analysis are:

(1) Sneak-Circuit Report, which documents a fully investigated sneak circuit, including potential impact statement and recommendation for correction.

(2) Design Concern Report, which documents the analyst's observation of a design element, which, while sneak free, presents unnecessary components, improper implementation of redundancy, critical potential failure points, a single inconsistent treatment of typical multiple circuits, etc.

(3) Drawing Error Report, which documents discrepancies in the detailed schematics upon which analysis is based.

The Sneak-Circuit Analysis of the F-8 DFBW contained 19 Sneak Circuit Reports disclosing 76 conditions, 7 Design Concern Reports disclosing 12 conditions, and 28 Drawing Error Reports disclosing 468 conditions.

Unfortunately, the drawings that had to be used for the analysis were preliminary. Thus, most of the faults detected would have been detected anyway during system integration tests. The greatest value of the analysis seems to be that another independent group went through all the drawings in detail.

One sneak circuit was discovered which may not have been discovered easily otherwise. This circuit involved the Yaw Trim Switch, and is shown in Figure 5-6. The wiring from the yaw trim right or left engage switch to the yaw trim digital circuitry bypasses switch S12 and thus destroys its selectivity. For example, if S12 is in the auxiliary position and the left yaw trim is engaged, a sneak circuit as shown by the arrows supplies power to the normal side. Thus, power is always applied to both normal and auxiliary positions no matter what position S12 is in. This circuit condition is typical of the kind of problems that can be discovered by Sneak Circuit Analysis. More experience would have to be gathered before a decision could be made concerning the cost effectiveness of this method of finding this type of error.

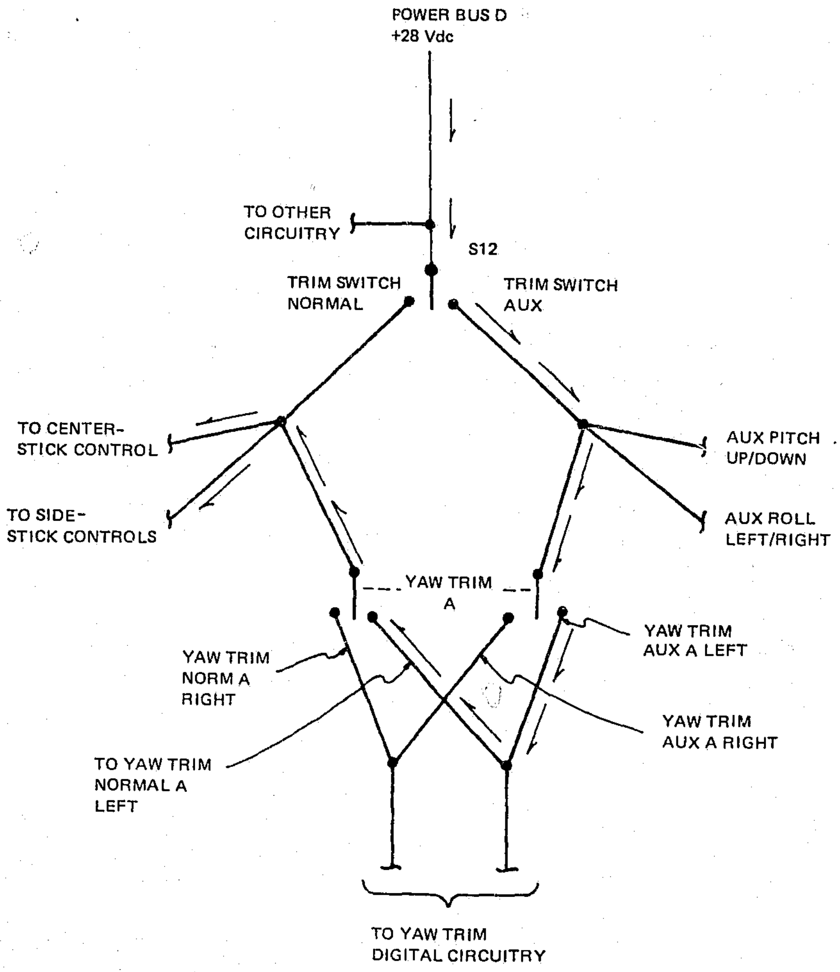


Figure 5-6. Sneak path in trim-control circuit.

5.2 Hardware Testing

5.2.1 Introduction

An adequately designed and implemented test program is a necessary building block for any successful system development. The F-8 DFBW test program may be thought of as being composed of two parts:

(1) Development tests.

(2) Acceptance and assurance tests.

5.2.1.1 Outline of Development Tests

These tests are, as a rule, devised by the contractor with buyer approval at the "plan" level, and consists of tests designed to support the development and fabrication of the system. The tests included in this category are:

(1) Component Tests, which provide data on characteristics and failure rates of component parts.

(2) Design Evaluation, which provides data needed to determine if intended system mechanization is capable of meeting requirements.

(3) Subsystem Test, which provides a means of verifying that subsystems, as built, will support system requirements.

(4) Integrated System Test, which provides data verifying that the system meets requirements.

Item (4) test results are the ultimate goal, with levels (1), (2), and (3) supportive of that goal. Figure 5-7 shows the intended time relationship of these tests.

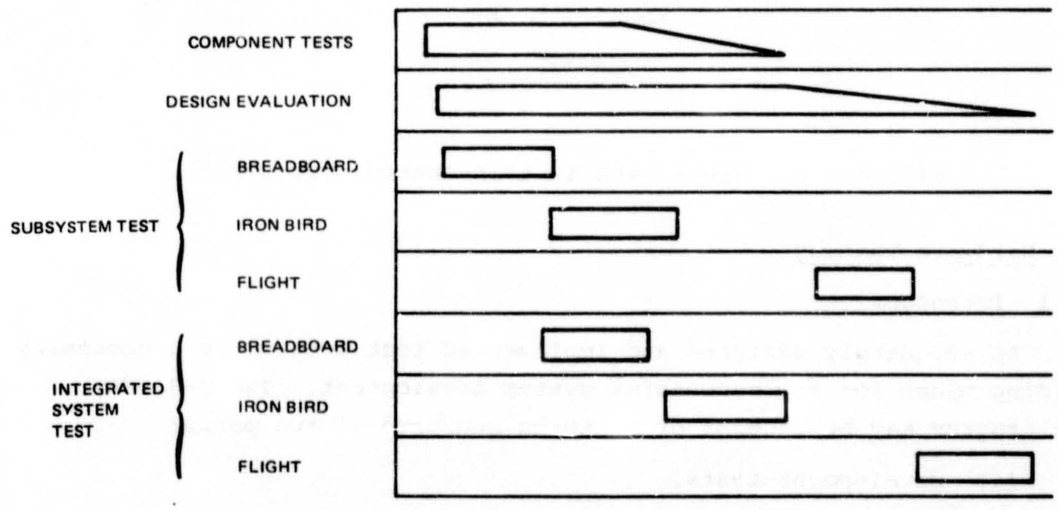


Figure 5-7. Time relationship of development tests.

5.2.1.2 Outline of Acceptance and Flight Assurance Tests

The acceptance and assurance tests are contractual requirements. The details of the test plans and procedures are devised by the manufacturer, and are reviewed and approved by the user. These tests are intended to show that the system meets its requirements as detailed in the statement of work. The tests consist of:

(1) Initial Acceptance Tests, which are performed at the manufacturer's facility, and use simulated interfaces.

(2) Flight Assurance Tests, which are performed at the manufacturer's facility, and test operation throughout the expected aircraft physical environmental range of vibration, temperature, and altitude.

(3) System Acceptance Tests, which are performed at the user's facility, and test operation in the DFRC iron bird, including a functional demonstration, failure modes and effects tests, reliability test, and EMI installation tests.

5.2.2 Development Tests Performance

5.2.2.1 Component Tests

Component testing varied greatly depending upon several factors. The tests performed depended on answers to the following questions:

(1) Was it tested by the manufacturer?

(2) Will it be difficult to replace during later stages of testing?

(3) Will adequate testing occur in later stages of testing?

(4) Is performance an important factor?

(5) Is it expensive?

Some examples of the above applied to specific components are:

(1) Rate Gyros and Accelerometers—Both of these items were tested by the manufacturer; however, the relatively high cost and the importance of adequate performance were sufficient to justify component testing. These sensors were purchased as off-the-shelf devices, and, in general, the performance exceeded that required by the DFBW system. Consequently, the tests were designed to ensure that DFBW system performance requirements were met.

(2) Integrated Circuits—In general, these devices were purchased to conform to MIL-STD-833B Level B where possible. Some exceptions were made as an expediency. MIL-STD-833B addresses manufacturing processes as well as tests of the finished device. Testing of ICs after receipt from the vendors was generally found to be costly and time inefficient.

5.2.2.2 Design Evaluation Performance

Design evaluation tests began as rudimentary tests early in the design stage and continued throughout the program, becoming most important during the build and operation of the first system breadboard. The objective of the design evaluation tests was to determine the adequacy of the design to meet system requirements. The most significant portion of the test relating to system redundancy was the failure modes and effects test and its relationship to the failure modes and effects analysis.

5.2.2.3 Subsystem Test Performance

The system is divided into four subassemblies:

(1) Interface unit/computers.

(2) Cockpit panels.

(3) Encoder/decoder.

(4) Sensor assembly.

An interface and control test console was built for each subassembly to enable verification of interfaces, operational redundancy and correct internal operation. This level of testing was performed on all three systems to ensure functional equivalence.

The tests were intended to verify that the major subsystem parts met the appropriate requirements. The tests were further designed to provide assurance that the systems (breadboard, iron-bird, and flight) were functionally identical. In retrospect, wiring errors have been found in the iron-bird system that were not uncovered during subsystem or integrated system testing, which lends support to the idea that no matter how well conceived or carried out a test is, it is still fallible. Testing, in general, and particularly at the subsystem and integrated levels, needs to be well thought out and thoroughly reviewed by design and system personnel to minimize oversights.

5.2.2.4 Integrated-System-Test Performance

The system-integration testing was performed in two stages.

(1) Operation of system with test software and static data.

(2) Operation of system with simulated cockpit and hybrid-flight simulator.

Initial test of the system used simulated aircraft interfaces and test software to verify correct interface operation. Preliminary tests of system response to transient power failures and loss of redundancy due to hardware failures were performed.

Following initial testing, to show correct system hardware operation, the system was interfaced with a simulated cockpit and a flight-simulation facility to verify operation of system in a simulated flight environment. Response of system to external signal interruptions and transient power failures was determined.

5.2.3 Acceptance and Flight Assurance Tests Performance

5.2.3.1 Initial Acceptance Test

The intent of the initial acceptance test was to demonstrate that the system was in working order prior to shipment to DFRC. The following tests were included:

(1) Verification of all analog input and output calibrations.

(2) Verification of discrete and serial I/O operations.

(3) Synchronization performance.

(4) Crosslink

(5) Built-in test.

(6) Simulated bus failures.

(7) Low- and high-power-bus operation.

(8) Interface with accelerometers and rate gyros.

(9) Correct display operation with and without power-bus failures.

(10) Computer redundancy management.

5.2.3.2 Flight Assurance Tests

The flight assurance tests were performed by the use of two procedures: vibration and temperature altitude. The system was tested at the subsystem level to provide maximum access to interfaces.

Vibration—Figure 5-8 defines the vibration input to the articles under test. The test consisted of a logarithmic sweep from 5 to 500 to 5 Hz over a 15-minute period. Resonant points were noted for each axis, and a resonant dwell of 2 minutes for each axis was performed following the sweep test.

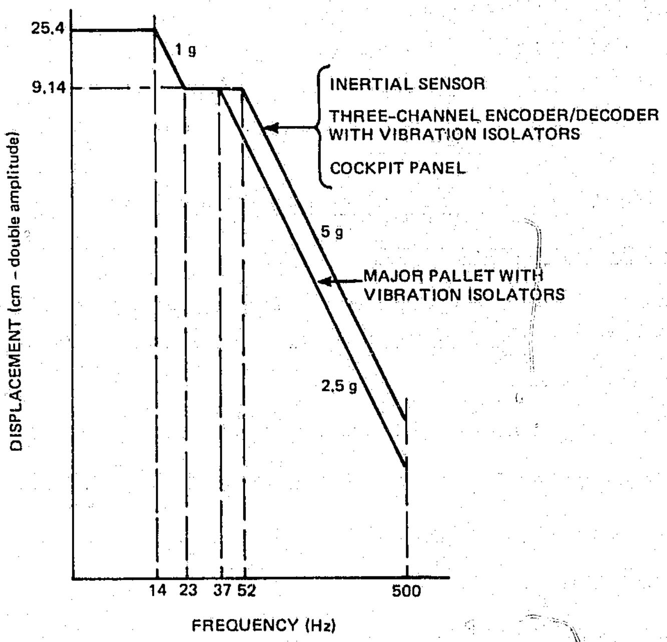


Figure 5-8. Flight assurance test—vibration profile.

Temperature-Altitude—The intent of the temperature-altitude test was to expose the system to combined extremes of temperature and altitude, which exceeded those expected in flight. Figure 5-9 shows the temperature altitude profile through which the system was operated. Cockpit panels were operated by remote manipulators during the test.

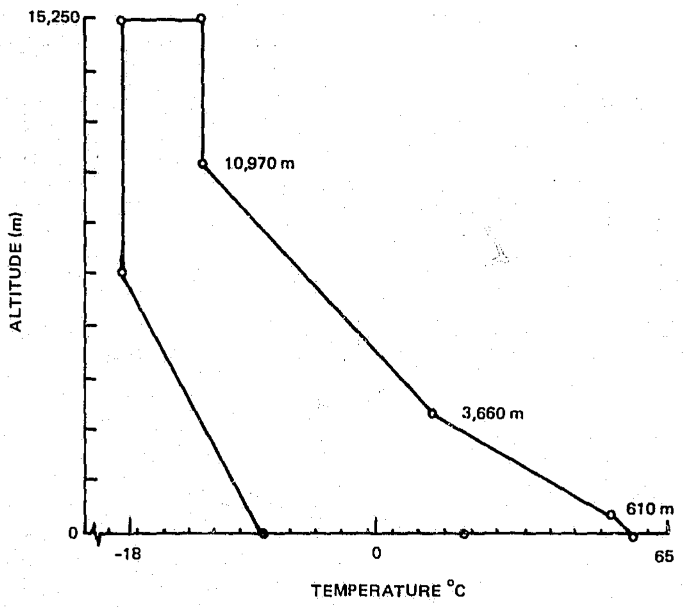


Figure 5-9. Flight assurance test—temperature/altitude profile.

Test Procedure—During exposure to the vibration and temperature-altitude test environments, the subsystem assemblies were connected to external test equipment, which provided control, interface stimulation, and monitoring capabilities. Figures 5-10 to 5-13 illustrate the test configuration. The test of the pallet assembly consisted of a combination of automatic test with automatic/manual monitoring and manual test with manual monitoring.

The automatic test was accomplished by "wrapping" analog outputs (dynamic profiles) through external wiring and back into the system as analog inputs. These inputs were interpreted by test software residing in the digital control computers. Notification of failure was provided to the operator by a coded message supplied on the interface console. Operator-induced "failures" were performed periodically during the test to demonstrate correct operation of failure-monitoring software.

The test of other subsystem assemblies was entirely by manual insertion of inputs and by manual monitoring.

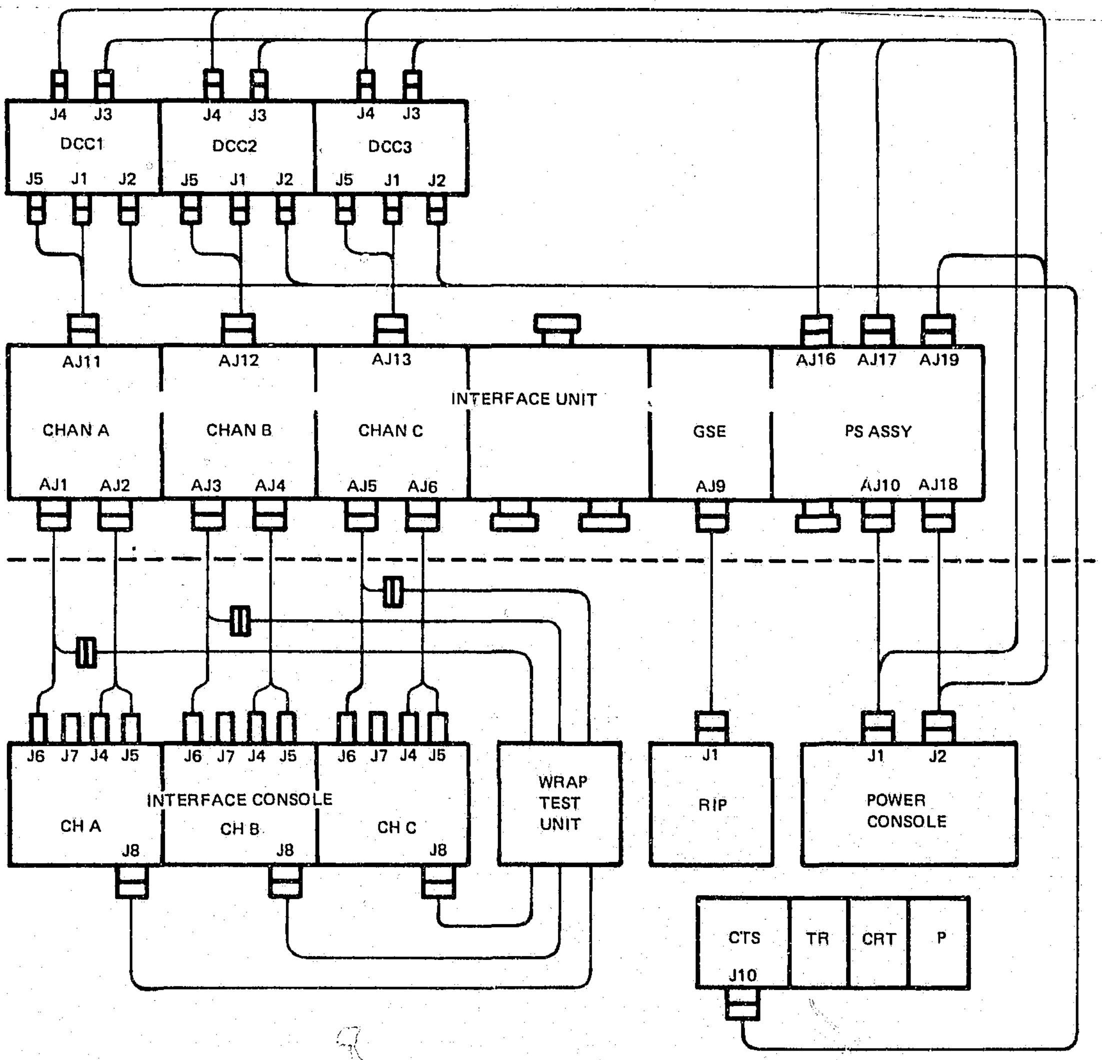


Figure 5-10. Static test technique.

5.2.3.3 System Acceptance Test

The system acceptance test consisted of three parts:

(1) Functional demonstration.

(2) Failure modes and effects test.

(3) EMI.

The above tests were performed with the flight system installed in the iron bird at DFRC. The failure modes and effects test was performed with the breadboard system installed in the iron-bird facility.

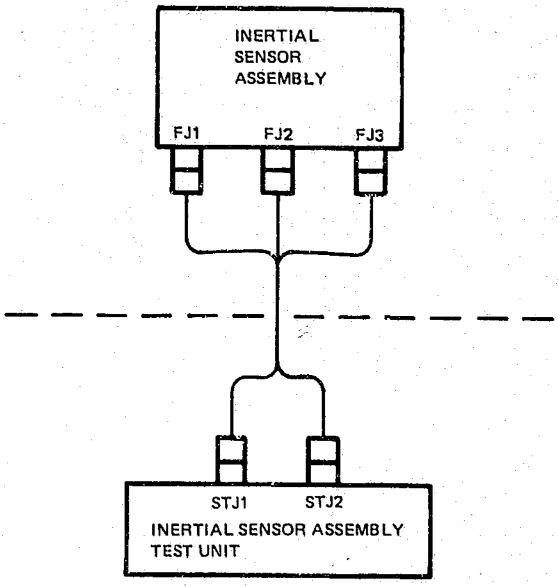


Figure 5-11. Inertial sensor assembly test configuration.

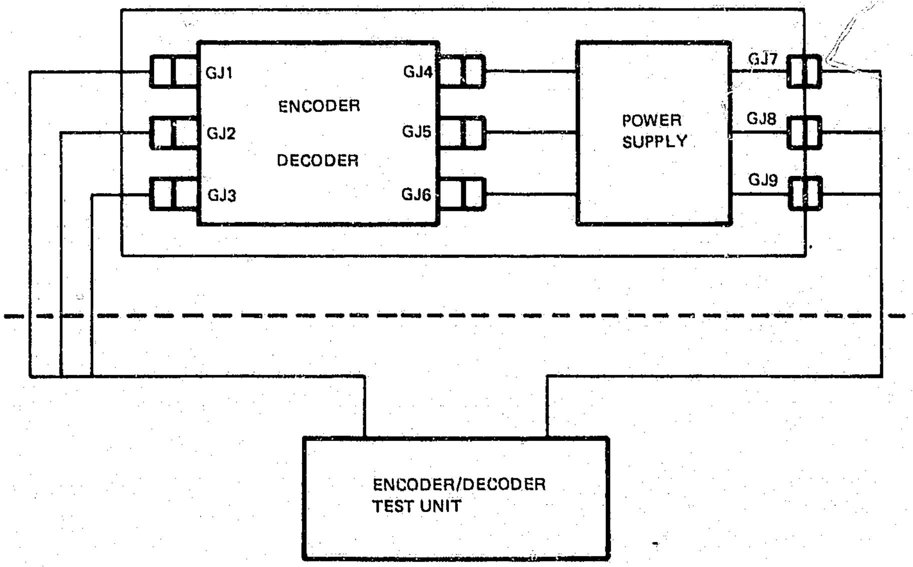


Figure 5-12. Encoder/decoder assembly test configuration.

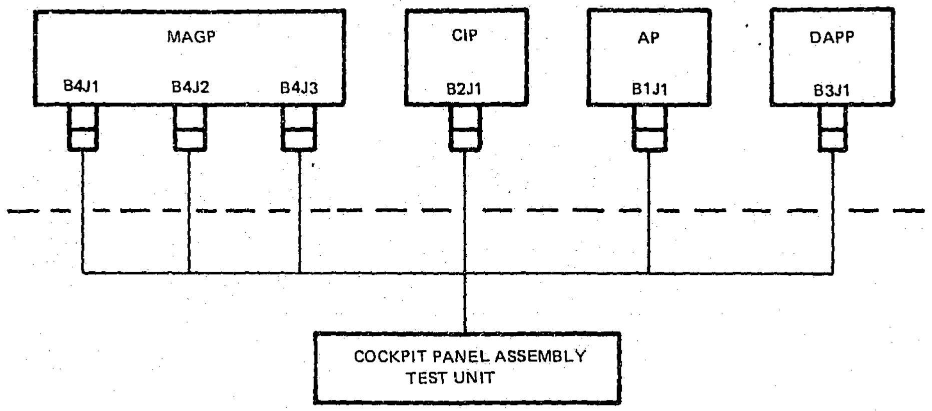


Figure 5-13. Cockpit panel assembly test configuration.

Functional Demonstration—The functional demonstration was intended to show correct operation of system installed in the iron bird by:

(1) Operation of subsystem assemblies.

(2) Operation of system using aircraft cables for subsystem interconnections.

(3) Interfaces with other aircraft interfaces.

5.3 Software and Systems-Level Test Facilities

There are a variety of test facilities available for verification and validation of digital flight-control systems. In fact, one of the important program decisions involved allocation of effort and resources in the test program to the various test facilities. All possible tests cannot be run on all possible facilities. A reasonable test matrix can be achieved only by partitioning tests so that tests are run on facilities producing the most realistic results, without unnecessary repetition.

In addition to major test facilities, some fundamental tools were also available for software development and digital computer testing. There were:

(1) Assembler and link editor support software for producing flight code and load tapes.

(2) Computer test set (CTS) for software and hardware debug (Figure 5-14).

(3) Function test program (FTP) for thorough testing of the digital computer.

(4) Real-time software for online debug of software.

(5) Tape drive, CRT, and printer for load, verify, and test.

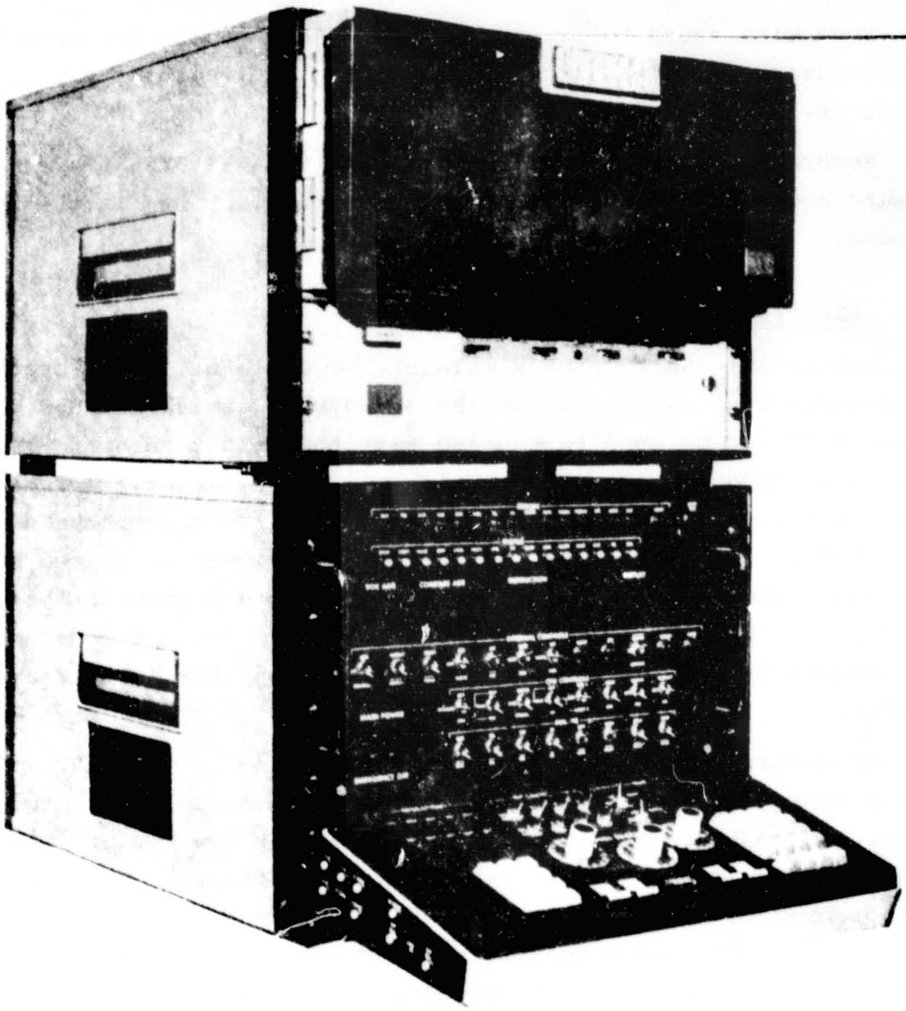


Figure 5-14. Computer test set.

It is important to note that the pieces of manufacturer-supplied ground-test equipment generally presume a single computer operation, and by themselves do not provide an adequate interface with a multi-computer system. This required multicomputer test tools to be built from the ground up. These tools included:

(1) Means to load multicomputer system.

(2) Special real-time displays of multicomputer operation.

(3) Special internal logs and tables to store multicomputer data following abnormal operation.

Even with these tools, it was found that visibility into the multicomputer system was lacking, resulting in lengthier troubleshooting procedures.

Several test facilities were used in the F-8 DFBW program for software and systems-level tests. These are described in the following sections.

5.3.1 Digital Computer Emulator

The AP-101 instruction simulator, or emulator, is a large software program that resides on an IBM 360/370 or equivalent host computer. The emulator can be used to run and test the flight software even before the actual flight computer is available. The emulator provides an instruction-by-instruction simulation of the flight computer's processing of the flight software. This allows the programmer to trace the exact execution sequence and to examine all intermediate computational results. This provides high visibility into the code and is a good debugging aid. Figure 5-15 illustrates the type of output available from the emulator.

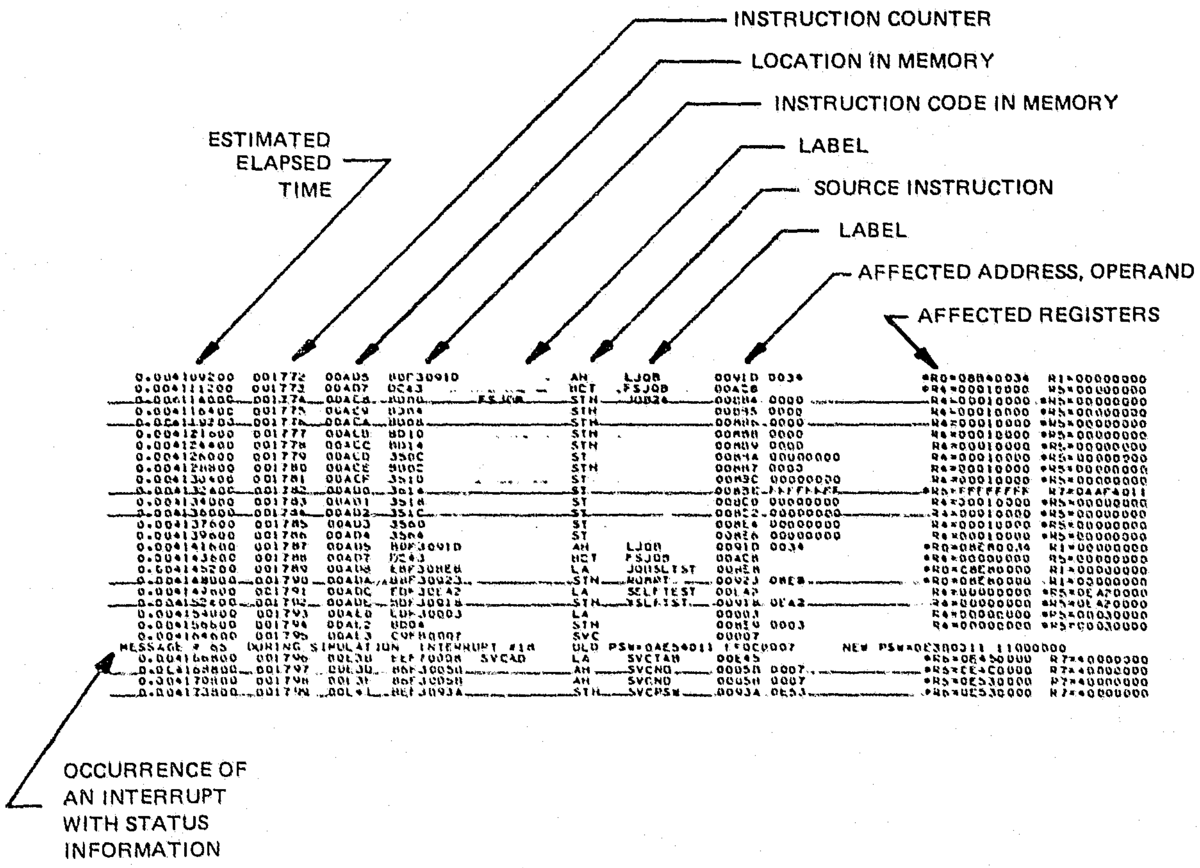
An emulator executes flight code very slowly, from 100 to 300 times slower than real time. This makes the emulator very expensive to use. An emulator also simulates only a single computer, and thus cannot be used to test multiflight computer software. Because the emulator executes in predetermined steps, it cannot accurately simulate the occurrence of random events, such as external interrupts. One additional aspect must also be recognized: the emulator yields results according to the flight-computer configuration as designed, and not always as it actually exists.

5.3.2 Single Flight Computer with Triplex Interfaces

This facility was used at the contractor's facility for software and interface hardware development. It provided early real-time experience on the flight software before all the flight computers were delivered. This testbed provided a controlled interface for initial systems tests. Figure 5-16 shows the functional operation of this testbed system.

This kind of a system, however, generally cannot be used with confidence to identify and fix problems associated with timing or intersystem communication. Outside of strictly single-string computations, it is never possible to say for sure what tests have validity when run on this type of system.

ORIGINAL PAGE IS
OF POOR QUALITY



```
  0.004109200  001772  00AD5  80F3091D               AH   LJOB      0091D 0034          *R0=08B40034  R1=00000000
  0.004111200  001773  00AD7  DC43                   HCT  .FSJOB    00AC8               *R4=00010000  R5=00000000
  0.004114000  001774  00AC8  4000        FSJOB      STH  JOB24     00084 0000           R4=00010000 *R5=00000000
  0.004116400  001775  00AC9  8304                   STH            00085 0000           R4=00010000 *R5=00000000
  0.004119200  001776  00ACA  8008                   STH            00088 0000           R4=00010000 *R5=00000000
  0.004121600  001777  00ACB  8D10                   STH            00088 0000           R4=00010000 *R5=00000000
  0.004124400  001778  00ACC  8D14                   STH            00089 0000           R4=00010000 *R5=00000000
  0.004126000  001779  00ACD  350C                   ST             0008A 00000000       R4=00010000 *R5=00000000
  0.004128800  001780  00ACE  900C                   STH            00087 0003           R4=00010000 *R5=00000000
  0.004130400  001781  00ACF  3510                   ST             0008C 00000000       R4=00010000 *R5=00000000
  0.004132400  001782  00AD0  3014                   ST             0008E FFFFFFFF      *R5=FFFFFFFF  R7=0AAF4011
  0.004134000  001783  00AD1  3518                   ST             000C0 00000000       R4=00010000 *R5=00000000
  0.004136000  001784  00AD2  351C                   ST             000E2 00000000       R4=00010000 *R5=00000000
  0.004137600  001785  00AD3  3560                   ST             000E4 00000000       R4=00010000 *R5=00000000
  0.004139600  001786  00AD4  3564                   ST             000E6 00000000       R4=00010000 *R5=00000000
  0.004141600  001787  00AD5  80F3091D               AH   LJOB      0091D 0034          *R0=08E80034  R1=00000000
  0.004143600  001788  00AD7  DC43                   HCT  FSJOB     00AC8               *R4=00000000  R5=00000000
  0.004145200  001789  00AD8  E8F30E8                LA   JOBSLTST  00E8                *R0=C8E80000  R1=00000000
  0.004148000  001790  00ADA  88F30023               STH  ROMPT     00023 0E8           *R0=08E80000  R1=00000000
  0.004149600  001791  00ADC  E0F30EA2               LA   SELFTEST  00EA2                R4=00000000 *R5=0EA20000
  0.004152400  001792  00ADE  50F30018               STH  %SLFTST   00018 0EA2           R4=00000000 *R5=0EA20000
  0.004154000  001793  00AE0  E0F30003               LA             00003                R4=00000000 *R5=00030000
  0.004156600  001794  00AE2  8D04                   STH            000E4 0003           R4=00000000 *R5=00030000
  0.004164000  001795  00AE3  C0F80007               SVC            00007
MESSAGE # 05  DURING SIMULATION  INTERRUPT #18   OLD PSW=0AE54011 1F0C0007    NEW PSW=0E300311 11000000
  0.004166800  001796  00E38  E8F70008    SVCAD      LA   SVCTAB    00E45               *R6=0E450000  R7=40000000
  0.004168800  001797  00E3A  88F30058               AH   SVCNO     00058 0007          *R5=0E4C0000  R7=40000000
  0.004170800  001798  00E3C  88F30058               AH   SVCNO     00058 0007          *R5=0E530000  R7=40000000
  0.004173800  001799  00E40  8EF3093A               STH  SVCPSW    0093A 0E53          *R6=0E530000  R7=40000000
```

Figure 5-15. Typical output from computer emulator.

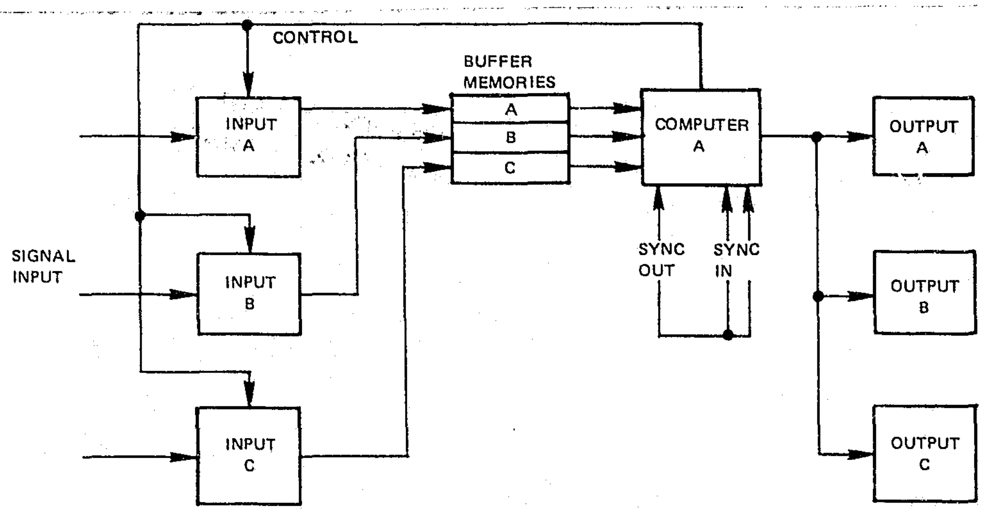


Figure 5-16. Single computer system testbed.

5.3.3 Multicomputer Triplex System in Iron Bird

There were two systems used in the F-8 DFBW development. Both used a triplex flight-computer set and a triplex interface unit. One interface unit was configured as a breadboard, but was electrically identical to the flight system. These triplex systems were installed in the NASA DFRC iron bird. Figure 5-17 shows the iron-bird configuration. A decommissioned F-8 aircraft formed the backbone of the iron bird. The equipment complement consisted of:

(1) Triplex primary digital system.

(2) Triplex analog-computer bypass system.

(3) Full set of flight actuators.

(4) Triplex electrical buses with batteries.

(5) Triplex hydraulic systems.

(6) Operable auxiliary aircraft systems (flaps, wing, gear/speedbrake actuators).

(7) Operable cockpit instrumentation.

The iron bird interfaces with a nonlinear simulation of the F-8 aerodynamics, which uses actual surface positions as input, and which drives both cockpit instruments and sensor lines to the flight-control system. The control surfaces are not artificially loaded, although predicted flexibility effects are included in the aerodynamic simulation.

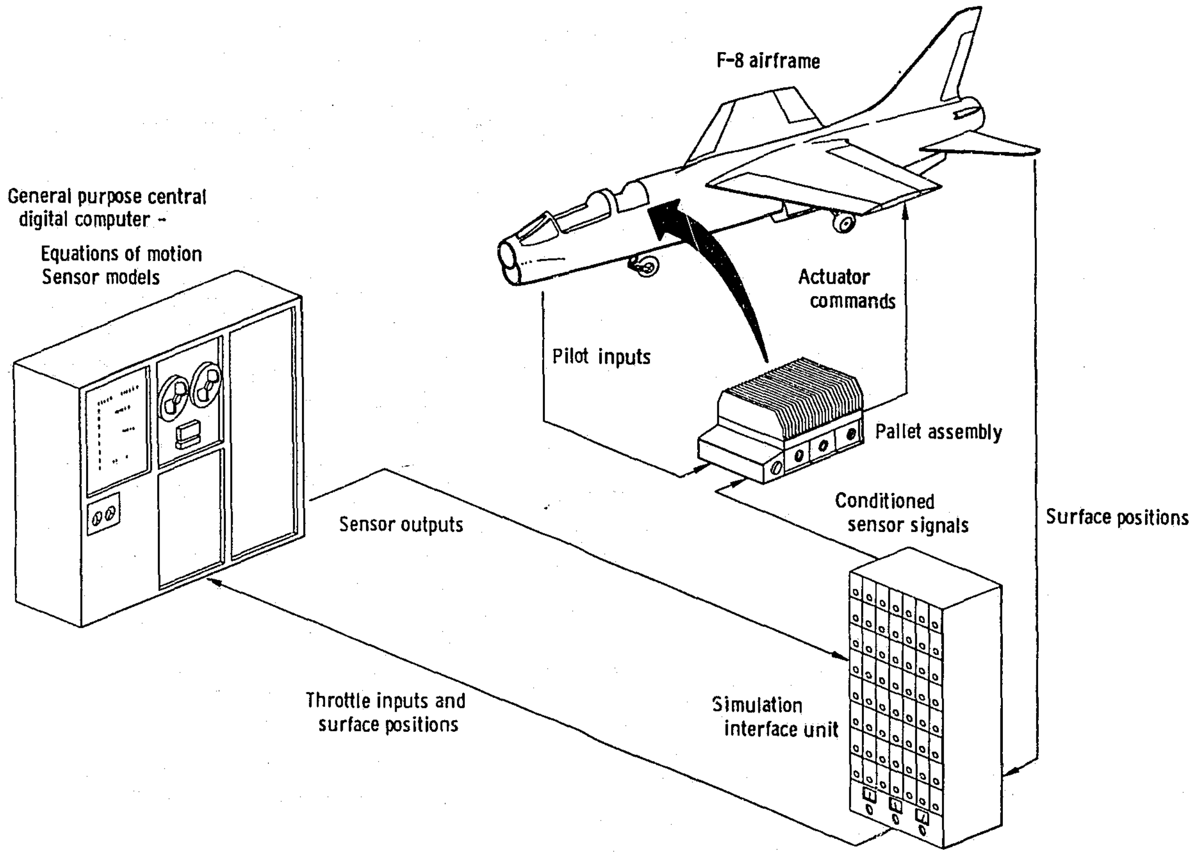

F-8 airframe
General purpose central digital computer -
Equations of motion
Sensor models
Actuator commands
Pilot inputs
Pallet assembly
Conditioned sensor signals
Surface positions
Sensor outputs
Throttle inputs and surface positions
Simulation interface unit

Figure 5-17. Iron-bird configuration.

The iron-bird facility was the key element in the system verification, validation, and flight qualification. It exposed the real system problems (described in a later section), allowed subsystem interface problems to be resolved, and especially provided a high degree of confidence in the verification and validation test results. The iron bird also provided the all-important pilot interface.

It should also be recognized that an iron-bird facility is expensive to construct and to maintain, being more like an airplane than a simulator.

5.3.4 Flight System in Aircraft

The aircraft itself is an important test facility, for only in the vehicle are all the subsystems in their true flight configuration. Final flight-readiness tests are best accomplished on the vehicle itself. If iron-bird testing has been properly accomplished, only higher level systems tests need be performed on the vehicle. The primary disadvantages of using the aircraft for substantially more than final system readiness tests are that the all-up airplane is available quite late in the development cycle, it is more difficult to tie in special test instrumentation, and there is a limited amount of time available on the aircraft for systems testing.

The flight-test program is also part of the system validation process. The flight environment provides those unmodeled characteristics that are not included in the simulation. The hardware itself is exposed to simultaneous temperature, vibration, and operational situations, which never seem to be covered in ground-test matrices. The pilot/crew interface with the system is stressed by mission or task factors that are not reproduced in ground simulation.

In the case of a flight-critical digital flight-control system, the flight regime represents a very hazardous verification and validation environment. Obviously, the core system must already be qualified to a high degree of confidence or must have a safe fall-back capability, or both, in order to make the first flight. In the case of the F-8 system, both conditions were required. With that in mind, the flight program proceeds as any other, but with a definite conservatism about it.

The flight environment, while providing the only truly credible qualification results, has obvious drawbacks as well. There are generally only a small number of test hours, visibility into the system operation is limited by instrumentaiton capability, and the cost is very high.

5.3.5 Actual Test Distribution

Tables 5-1 and 5-2 show the estimated test distribution on the basis of man-hour percentages for the F-8 DFBW system. Table 5-1 applies to the testing that was accomplished at the Draper Laboratory. The emulator was used primarily for the executive software verification, and also for testing of short code sequences early in the software development. Those functions that could be tested on a single computer system were verified almost exclusively on the single computer testbed. This facility was found to be a poor testbed for multicomputer software testing, however, because it failed to expose several problems which were found only when actual multicomputer operations began.

Table 5-1. Percent man-hour test-distribution estimates (CSDL testing).

Test Area	Test Facility		
	Emulator	Single Computer	Multi-Computer
Executive	50	25	25
Input-Output		80	20
Computer Redundancy Management		10	90
Sensor Redundancy Management		90	10
Self-Test (in-flight)	25	75	0
Preflight Test (ramp)		80	20
Control Laws	5	90	5
Cockpit Displays/Controls		90	10
Ground-Test Programs		90	10

Table 5-2. Percent man-hour test-distibution estimates (NASA DFRC testing).

Test Area	Test Facility		
	Iron Bird	Aircraft	Flight
Executive	100	--	--
Input-Output	90	10	--
Computer Redundancy Management	95	5	--
Sensor Redundancy Management	90	5	5
Self-Test	100	--	--
Preflight Test	50	50	--
Control Laws	95	5	--
Cockpit Displays/Controls	95	5	--
Primary/Bypass System Interface	90	10	--
Actuator/Electronics Interface	60	40	--
Electrical/Hydraulic Power	50	50	--
Sensor Hardware Interface	10	80	10

Table 5-2 shows a similar breakdown for the testing done at NASA DFRC. Only four areas received substantial attention on the aircraft itself:

(1) Preflight test software.

(2) Servo electronics/actuator interface.

(3) Electrical/hydraulic systems.

(4) Motion sensor interface.

The preflight test program was verified on the iron bird, but was validated and qualified on the aircraft because of the need to show proper operation with the airplane configuration. The servo electronics/actuator interface was performed on the airplane principally to off-load work on the iron bird, which was heavily committed to software and digital system testing. The electrical and hydraulic systems were laid out and tested initially in the iron bird, but of course, had to be qualified in the aircraft itself.

Although all sensor I/O and redundancy management software was verified/validated/qualified on the iron bird, the hardware interfaces were also tested and qualified on the airplane itself. Because of cost, the iron bird was not normally operated with actual sensors, and had no provision for an actual air-data interface. Special closed-loop resonance tests were also performed on the airplane in which structural mode excitation margins were determined. This required an all-up system and sensor configuration, which was available only on the actual vehicle.

The only deliberate validation tests slated for the flight-test phase of the program were the sensor failure-detection threshold tests, which are dependent on actual sensor-output characteristics in the flight environment.

5.4 Software Verification

Software verification—the determination that the generated code correctly performs its intended function—ideally begins in the design phase. A software program consists of many modules of varying size, which together accomplish the total intended purpose of the program. If the individual modules are adequately defined and controlled from the design phase on, the verification is made less complex and costly in both time and expense.

5.4.1 <u>Quality Assurance in Design</u>

Quality assurance in design depends upon two important factors:

(1) The software specification document.

(2) Programming ground rules.

Careful and full attention to these two factors will greatly ease the complexity and time necessary to perform software verification.

5.4.1.1 <u>Software Specification</u>

The interpretation of the requirements must be clear and straightforward and mutually agreed to by all parties involved. All changes or additions must be tightly controlled and documented. All approved changes must be published and communicated to all programmers working on code generation. Figure 5-18 is an example of software specification change form.

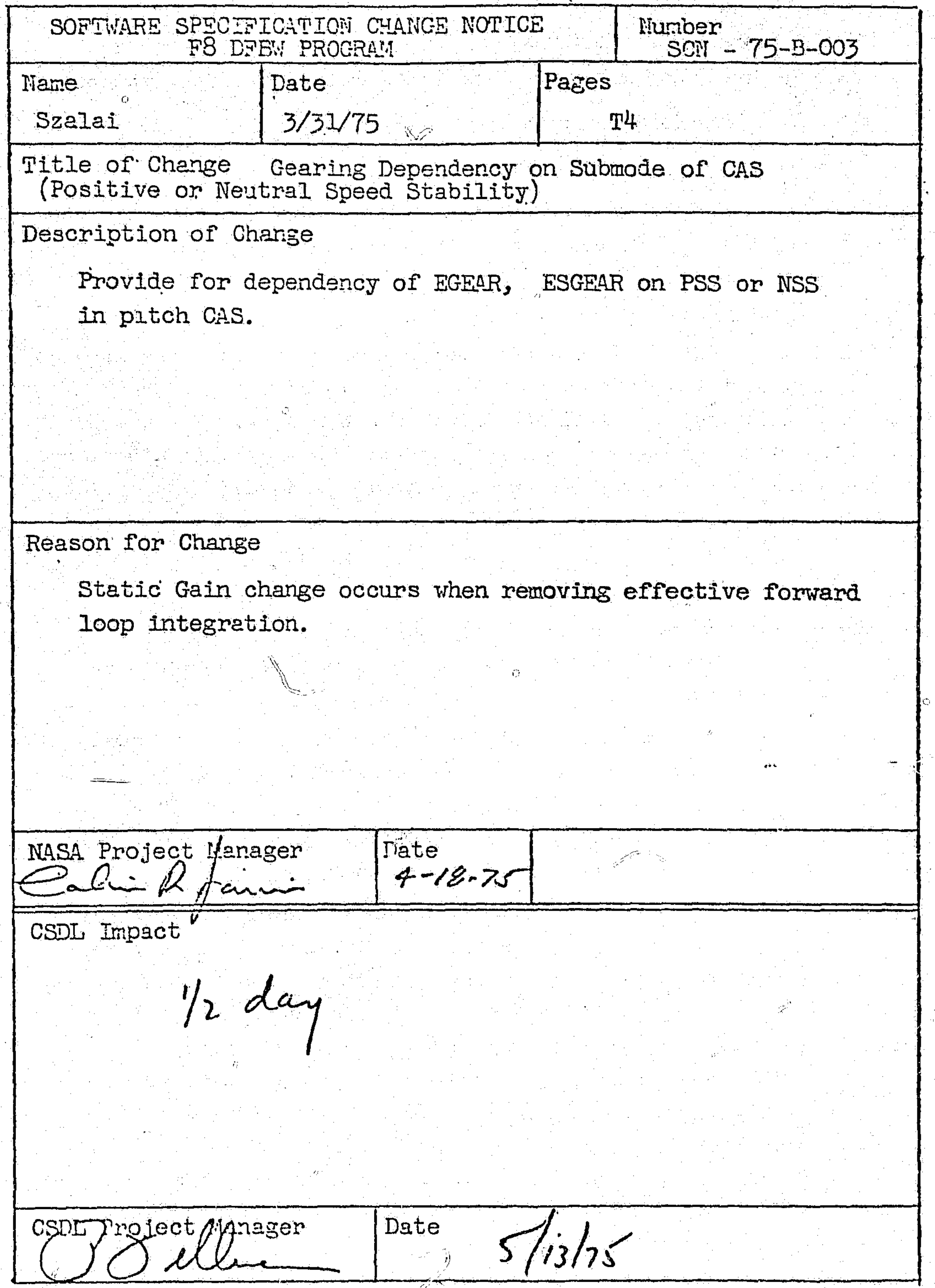
5.4.1.2 <u>Programming Ground Rules</u>

The establishment of programming ground rules is essential to the production, testing, and modification or updating of generated code.

<u>Logic Diagrams</u>—The first and foremost rule is: no code must be written until a logic diagram has been generated. As in the building of hardware, one needs blueprints to correctly build a piece of equipment. A programmer needs a logic-flow diagram to correctly code and document a particular computer function.

<u>Module Specification</u>—The specification of maximum module entry and exit points (no more than two of each), self-contained temporary storage, and minimum external references ensure that each module will be self-contained to the maximum level, and, once tested and proved to perform its function, will not be altered by the addition of another module to the developing program. Each module must be single purpose in function and that function must be carefully and fully specified. The execution of the module's function must be independent of other modules, except for common library routines which may be called by the module, and all necessary manipulation of arguments must be self-contained within the module.

The number and form of entry and exit arguments must be specifically called out and identified.



SOFTWARE SPECIFICATION CHANGE NOTICE F8 DFBW PROGRAM	Number SCN - 75-B-003

Name Szalai	Date 3/31/75	Pages T4

Title of Change Gearing Dependency on Submode of CAS (Positive or Neutral Speed Stability)

Description of Change

Provide for dependency of EGEAR, ESGEAR on PSS or NSS in pitch CAS.

Reason for Change

Static Gain change occurs when removing effective forward loop integration.

NASA Project Manager [illegible]	Date 4-18-75	

CSDL Impact

1/2 day

CSDL Project Manager [illegible]	Date 5/13/75

Figure 5-18. Typical Software Specification Change form.

Macros—The development and use of "macros" to accomplish common subroutines in each module again reduces the level of testing, which must be performed to verify each module.

Instruction Usage—Most computers, unless specifically designed to meet a given functional role, have an instruction repertoire far in excess of coding requirements. Many of these instructions are complex in execution and require extreme care in use. Totally eliminating or tightly controlling the use of complex instructions will make program verification and debug less complex and costly. Straightforward programming may cost some penalty in core requirements and execution time, but save considerably in the long run.

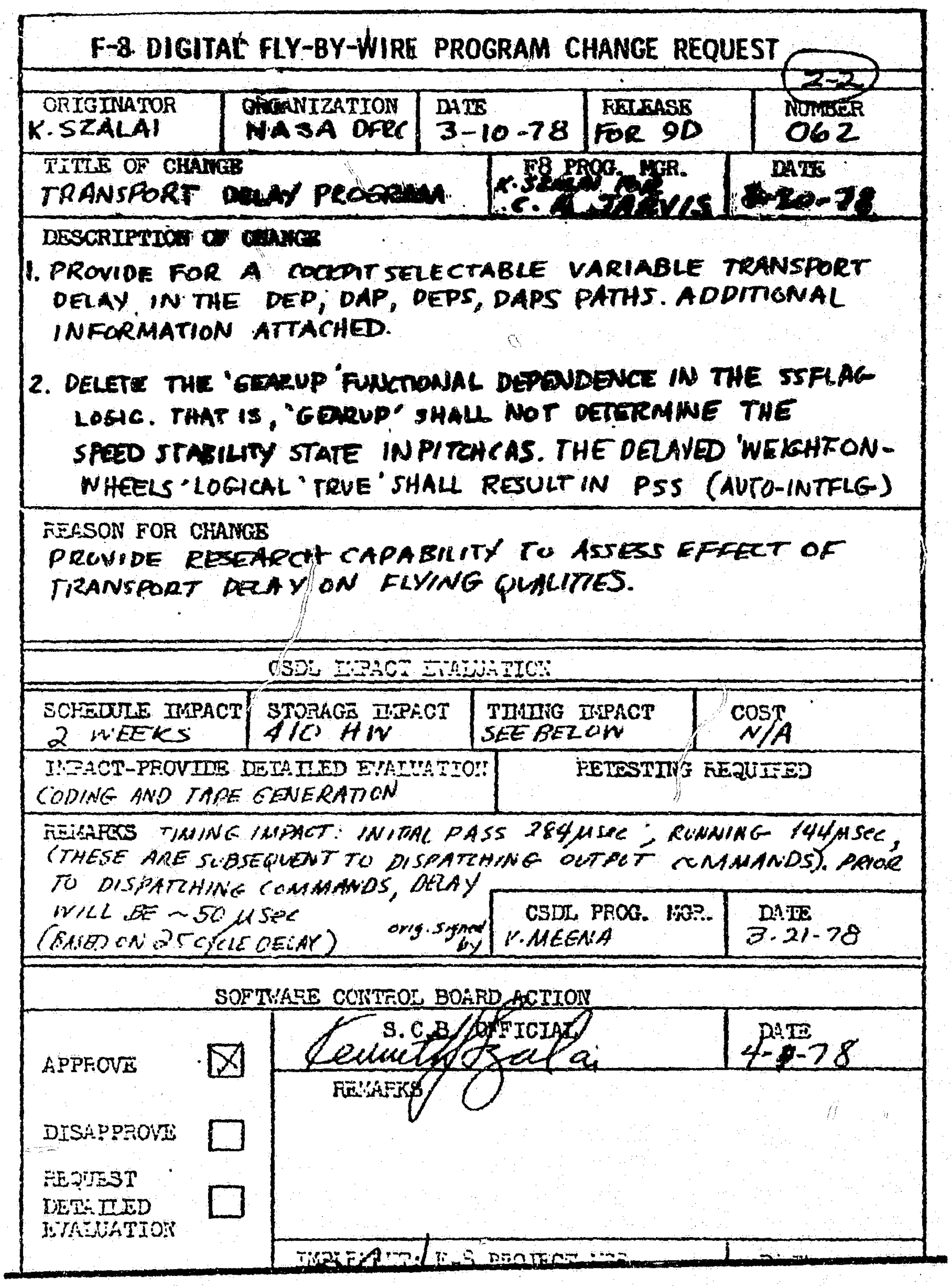
5.4.2 Module Verification

The verification of a module's function begins prior to code generation. The first step is the construction of the logic-flow diagram, which basically describes the module's functions and serves as the programmer's guide in writing the module code. The generated code is checked against this flow diagram by both the programmer and some other programmer, which both verifies the logic diagram and code generated.

Module Testing—The programmer establishes the type and degree of testing to which a particular module will be exposed. This written document is approved for completeness and correctness by a controlling body. The module's function is than tested on either a computer emulator or, even better, by execution in the acutal target computer. When testing execution of module code, extreme care is required to ensure that any dummy routines used in the testing of the module do not cover up some error in the module code.

Program Library—Once a module has been satisfactorily tested and proved to perform its intended function, access control to the module is restricted to the program librarian. The program librarian maintains a file of all verified modules and ensures that all changes made to a module in the library are completely documented and publicized.

Change Documentation—The documentation required to change a module in the program library should contain the change, the reason for change, effects upon code execution, changes in core requirement, and approval of a change control agent. Figures 5-19 and 5-20 are examples of change documentation forms.



F-8 DIGITAL FLY-BY-WIRE PROGRAM CHANGE REQUEST

2-2

ORIGINATOR K. SZALAI	ORGANIZATION NASA DFRC	DATE 3-10-78	RELEASE FOR 9D	NUMBER 062

TITLE OF CHANGE TRANSPORT DELAY PROGRAM	F8 PROG. MGR. K. SZALAI FOR C. A. JARVIS	DATE 3-20-78

DESCRIPTION OF CHANGE

1. PROVIDE FOR A COCKPIT SELECTABLE VARIABLE TRANSPORT DELAY IN THE DEP, DAP, DEPS, DAPS PATHS. ADDITIONAL INFORMATION ATTACHED.

2. DELETE THE 'GEARUP' FUNCTIONAL DEPENDENCE IN THE SSFLAG LOGIC. THAT IS, 'GEARUP' SHALL NOT DETERMINE THE SPEED STABILITY STATE IN PITCH CAS. THE DELAYED 'WEIGHT-ON-WHEELS' LOGICAL 'TRUE' SHALL RESULT IN PSS (AUTO-INTFLG)

REASON FOR CHANGE

PROVIDE RESEARCH CAPABILITY TO ASSESS EFFECT OF TRANSPORT DELAY ON FLYING QUALITIES.

CSDL IMPACT EVALUATION

SCHEDULE IMPACT 2 WEEKS	STORAGE IMPACT 410 HW	TIMING IMPACT SEE BELOW	COST N/A

IMPACT-PROVIDE DETAILED EVALUATION CODING AND TAPE GENERATION	RETESTING REQUIRED

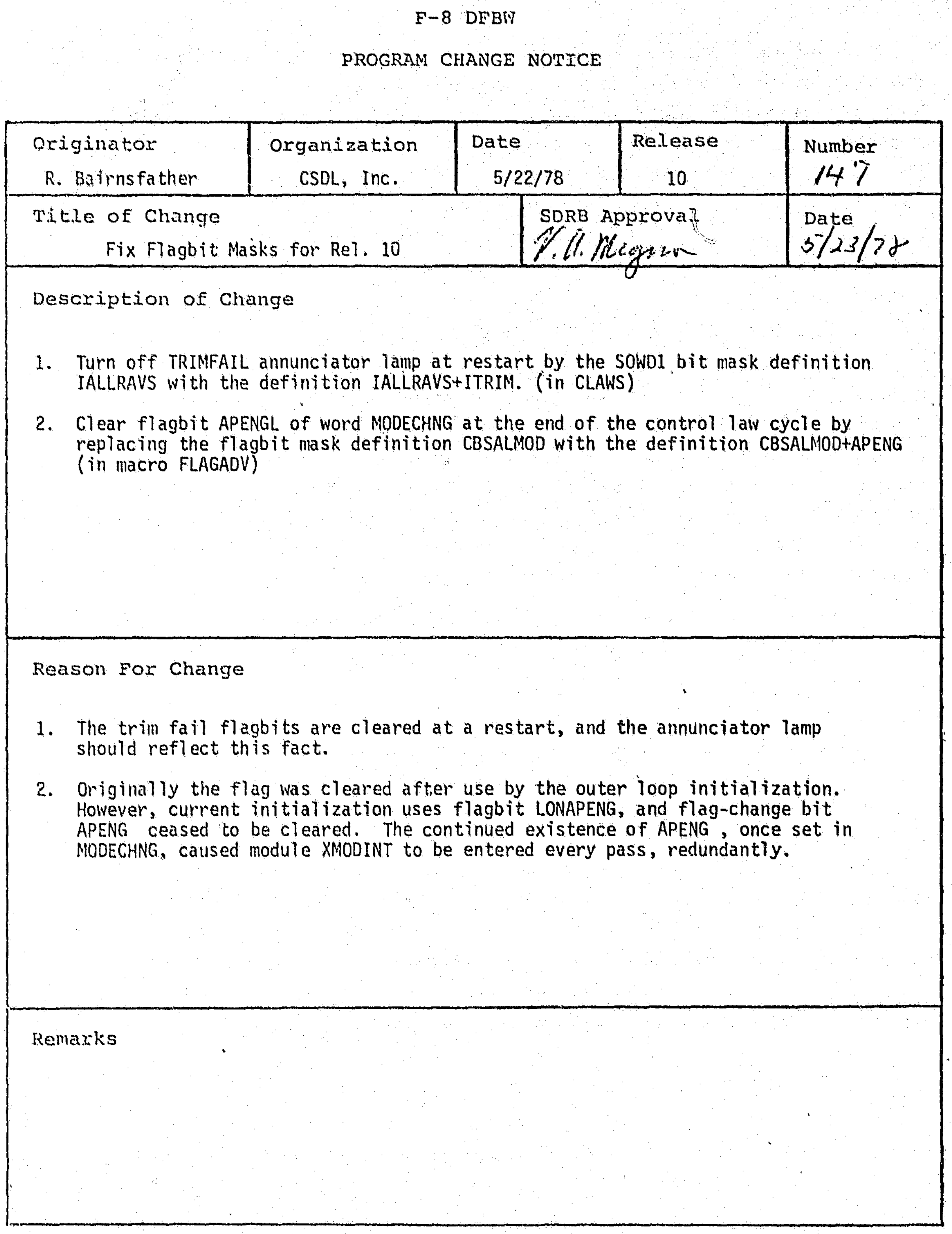
REMARKS TIMING IMPACT: INITIAL PASS 284 μSEC ; RUNNING 144 μSEC, (THESE ARE SUBSEQUENT TO DISPATCHING OUTPUT COMMANDS). PRIOR TO DISPATCHING COMMANDS, DELAY WILL BE ~50 μSEC (BASED ON 25 CYCLE DELAY) orig. signed by

CSDL PROG. MGR. V. MEENA	DATE 3-21-78

SOFTWARE CONTROL BOARD ACTION

APPROVE [✓]	S.C.B. OFFICIAL [illegible]	DATE 4-8-78
DISAPPROVE ☐	REMARKS	
REQUEST DETAILED EVALUATION ☐		

Figure 5-19. Typical Progress Change Request originated by NASA.



F-8 DFBW

PROGRAM CHANGE NOTICE

Originator R. Bairnsfather	Organization CSDL, Inc.	Date 5/22/78	Release 10	Number 147
Title of Change Fix Flagbit Masks for Rel. 10		SDRB Approval [illegible]		Date 5/23/78

Description of Change

1. Turn off TRIMFAIL annunciator lamp at restart by the SOWD1 bit mask definition IALLRAVS with the definition IALLRAVS+ITRIM. (in CLAWS)
2. Clear flagbit APENGL of word MODECHNG at the end of the control law cycle by replacing the flagbit mask definition CBSALMOD with the definition CBSALMOD+APENG (in macro FLAGADV)

Reason For Change

1. The trim fail flagbits are cleared at a restart, and the annunciator lamp should reflect this fact.
2. Originally the flag was cleared after use by the outer loop initialization. However, current initialization uses flagbit LONAPENG, and flag-change bit APENG ceased to be cleared. The continued existence of APENG , once set in MODECHNG, caused module XMODINT to be entered every pass, redundantly.

Remarks

Figure 5-20. Typical Program Change Notice.

Whenever a module in the library is changed, it receives a new revision number. The inclusion of this revision number within the body of the module serves as a means of verifying that an integrated program consisting of many modules contains the latest revision of the particular module. Figures 5-21 and 5-22 are the computer listings of the process used to implement the changes in the modules requested in Figures 5-19 and 5-20. Reverification of the module function must be performed before it is returned to the program library.

5.4.3 Module Integration on Flight Computer

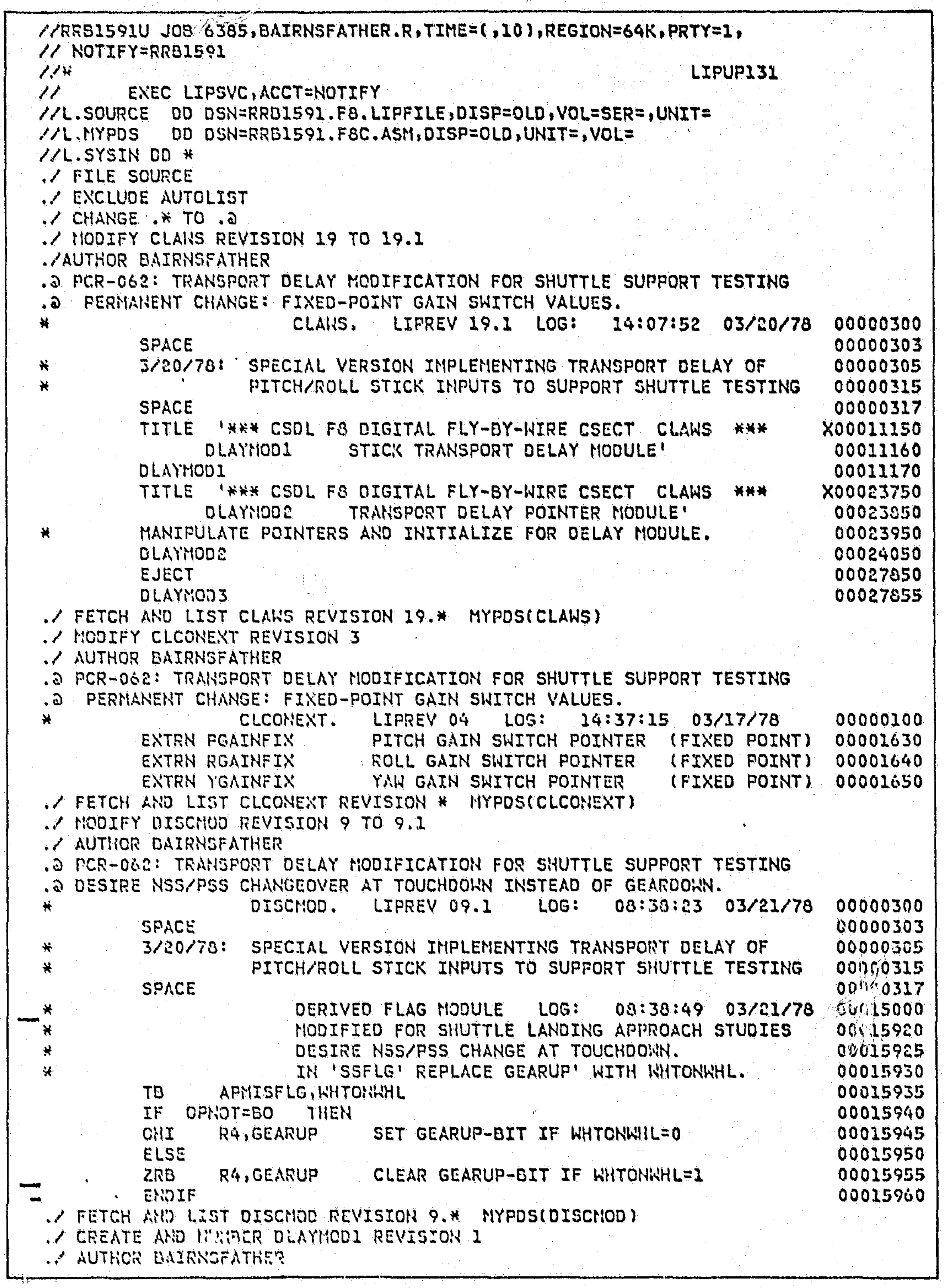
When the program library contains a sufficient number of verified modules to begin generation of an integrated program, the program librarian initiates a production of the program. The program generated will contain the verified modules and any dummy routines necessary for program execution. Tables 5-3 and 5-4 are examples of program contents for an executable computer program. Table 5-3 shows the modules for the execution, synchronization, redundancy management, etc. Table 5-4 shows the modules for the control laws.

The program is then executed in the flight computer, preferably in a full system configuration. Extensive use is made of available ground-test equipment to ensure that the program execution is correct and as planned.

The execution of a program on a computer emulator is not recommended as this does not fully and truly simulate actual target-computer or system operation. An emulator is not normally capable of generating all the random interrupts and time slicing which occurs in the actual system configuration. This factor is extremely important when dealing with multicomputer systems. On the F-8 program, the inability to test software changes on a multicomputer system at CSDL prior to release has proven troublesome. A program may execute correctly in a single computer and fail miserably when exposed to the randomness of operation in the actual multicomputer configuration.

5.5 System Validation

The system validation tests are performed on hardware and software that are essentially operational, i.e., all major hardware and software components are functioning in a nominal manner. This is



```
//RRB1591U JOB 6385,BAIRNSFATHER.R,TIME=(,10),REGION=64K,PRTY=1,
// NOTIFY=RRB1591
//*                                                        LIPUP131
//        EXEC LIPSVC,ACCT=NOTIFY
//L.SOURCE  DD DSN=RRB1591.F8.LIPFILE,DISP=OLD,VOL=SER=,UNIT=
//L.MYPDS   DD DSN=RRB1591.F8C.ASM,DISP=OLD,UNIT=,VOL=
//L.SYSIN DD *
./ FILE SOURCE
./ EXCLUDE AUTOLIST
./ CHANGE .* TO .@
./ MODIFY CLAWS REVISION 19 TO 19.1
./AUTHOR BAIRNSFATHER
.@ PCR-062: TRANSPORT DELAY MODIFICATION FOR SHUTTLE SUPPORT TESTING
.@  PERMANENT CHANGE: FIXED-POINT GAIN SWITCH VALUES.
*                  CLAWS.    LIPREV 19.1  LOG:    14:07:52  03/20/78  00000300
         SPACE                                                        00000303
*        3/20/78:  SPECIAL VERSION IMPLEMENTING TRANSPORT DELAY OF    00000305
*                  PITCH/ROLL STICK INPUTS TO SUPPORT SHUTTLE TESTING 00000315
         SPACE                                                        00000317
         TITLE  '*** CSDL F8 DIGITAL FLY-BY-WIRE CSECT  CLAWS  ***   X00011150
               DLAYMOD1     STICK TRANSPORT DELAY MODULE'             00011160
         DLAYMOD1                                                     00011170
         TITLE  '*** CSDL F8 DIGITAL FLY-BY-WIRE CSECT  CLAWS  ***   X00023750
               DLAYMOD2     TRANSPORT DELAY POINTER MODULE'           00023850
*        MANIPULATE POINTERS AND INITIALIZE FOR DELAY MODULE.         00023950
         DLAYMOD2                                                     00024050
         EJECT                                                        00027850
         DLAYMOD3                                                     00027855
./ FETCH AND LIST CLAWS REVISION 19.*  MYPDS(CLAWS)
./ MODIFY CLCONEXT REVISION 3
./ AUTHOR BAIRNSFATHER
.@ PCR-062: TRANSPORT DELAY MODIFICATION FOR SHUTTLE SUPPORT TESTING
.@  PERMANENT CHANGE: FIXED-POINT GAIN SWITCH VALUES.
*               CLCONEXT.   LIPREV 04    LOG:    14:37:15  03/17/78   00000100
         EXTRN PGAINFIX       PITCH GAIN SWITCH POINTER  (FIXED POINT) 00001630
         EXTRN RGAINFIX       ROLL GAIN SWITCH POINTER   (FIXED POINT) 00001640
         EXTRN YGAINFIX       YAW GAIN SWITCH POINTER    (FIXED POINT) 00001650
./ FETCH AND LIST CLCONEXT REVISION *  MYPDS(CLCONEXT)
./ MODIFY DISCMOD REVISION 9 TO 9.1
./ AUTHOR BAIRNSFATHER
.@ PCR-062: TRANSPORT DELAY MODIFICATION FOR SHUTTLE SUPPORT TESTING
.@ DESIRE NSS/PSS CHANGEOVER AT TOUCHDOWN INSTEAD OF GEARDOWN.
*                DISCMOD.    LIPREV 09.1   LOG:    08:38:23  03/21/78  00000300
         SPACE                                                        00000303
*        3/20/78:  SPECIAL VERSION IMPLEMENTING TRANSPORT DELAY OF    00000305
*                  PITCH/ROLL STICK INPUTS TO SUPPORT SHUTTLE TESTING 00000315
         SPACE                                                        00000317
*                      DERIVED FLAG MODULE    LOG:    08:38:49  03/21/78  00015000
*                      MODIFIED FOR SHUTTLE LANDING APPROACH STUDIES  00015920
*                      DESIRE NSS/PSS CHANGE AT TOUCHDOWN.            00015925
*                      IN 'SSFLG' REPLACE GEARUP' WITH WHTONWHL.      00015930
         TB     APMISFLG,WHTONWHL                                     00015935
         IF  OPNOT=BO   THEN                                          00015940
         OHI    R4,GEARUP      SET GEARUP-BIT IF WHTONWHL=0           00015945
         ELSE                                                         00015950
         ZRB    R4,GEARUP      CLEAR GEARUP-BIT IF WHTONWHL=1         00015955
         ENDIF                                                        00015960
./ FETCH AND LIST DISCMOD REVISION 9.*  MYPDS(DISCMOD)
./ CREATE AND NUMBER DLAYMOD1 REVISION 1
./ AUTHOR BAIRNSFATHER
```

Figure 5-21. Typical computer listing of change being made in library module.

```
//RRB1591U JOB 6385,BAIRNSFATHER.R,TIME=(,10),REGION=64K,PRTY=1,
// NOTIFY=RRB1591
//*                                                               LIPUP144
//       EXEC LIPSVC,ACCT=NOTIFY
//L.SOURCE  DD DSN=RRB1591.F8.LIPFILE,DISP=OLD,VOL=SER=,UNIT=
//L.MYPDS   DD DSN=RRB1591.F8C.ASM,DISP=OLD,UNIT=,VOL=
//L.SYSIN DD *
./ FILE SOURCE
./ EXCLUDE AUTOLIST
./ CHANGE .* TO .@
./ MODIFY CLAWS REVISION 21
./ AUTHOR BAIRNSFATHER
.@ PCN-147  :EXTINGUISH TRIMFAIL LAMP AT RESTART TO CONFORM TO FAILBITS
*                   CLAWS.   LIPREV 22    LOG:   16:50:39  05/22/78  00000300
        ZB    SOWD1,IALLRAVS+ITRIM   TURN OFF RAV LAMPS; NOT TOUCHED   00007600
./ FETCH AND LIST CLAWS REVISION *  MYPDS(CLAWS)
./ MODIFY FLAGADV REVISION 3
./ AUTHOR BAIRNSFATHER
.@ FCN-147 : REMOVE APENGL FROM MODECHNG; LONAPENG INITIALIZES OUTERLOOP
*                  FLAGADV.   LIFREV 04   LOG:   16:53:47  05/22/78  00000300
        ZB    MODECHNG,CBSALMOD+APENG      CLEAR ALL BUT AP MODS       00002900
./ FETCH AND LIST FLAGADV REVISION *  MYPDS(FLAGADV)
./ MODIFY AAALIPUP  REVISION 143
./ AUTHOR BAIRNSFATHER
.@   FOR FILE IDENTIFICATION: USE LIPUP # FOR REV NUMBER
*    LIPUP   144     REL 10       LOG:   16:48:05  05/22/78              00000100
./ FETCH AND LIST AAALIPUP REVISION *  MYPDS(AAALIPUP)
```

Figure 5-22. Typical computer listing of change being made in library module.

important because every minute of operating time is an implicit test, exercising hardware and software interfaces hundreds of times. During this period of close scrutiny, it is critical that as much of the system as technically possible be operating, so as to expose deficiencies that occur during the test phase, but outside of the test plan. System validation is accomplished by carrying out several different types of tests (refer to Section 4.2.4). This section relates actual system test experience.

5.5.1 Independent Verification and Validation

A test team at NASA DFRC that peaked at 7 man-months per month was responsible for these tests. These tests comprehensively verified individual functions and validated major system operation. Many of these tests were duplicates of those accomplished by the programming

Table 5-3. List of modules (except control laws) for executable program.

CSECT	LIPSVC Revision Number
F8PSA	38*
F8DAT	88*
F8TRN	29
F8FST	50*
F8MAN	8
F8SYNC	33
F8RUP	20
F8DLN	21*
F8SER	64*
F8IO	63
F8XLI	58*
F8CIP	40*
F8SLF	30
F8RM1S	75*
F8RM1	94*
F8RM1D	71*
F8PTP	74
F8DSP	33
F8LD	17*
MACRO	
EQERR	15*
EQJOB	1
EQICR	1
EQSVC	1
EQREG	1

Table 5-4. Control-law modules for an executable program.

```
 ABSI       REV  3      IF        REV  2     SENVAR    REV  1
#ADDONS     REV  3      INBITEXT  REV  2     SENVENT   REV  2
 ALFAMOD    REV  1      INITMOD   REV  12*   SENVEXT   REV  1
 ALIMDP     REV  5      INIRMOD   REV  2     SFCTRS    REV  17*
 ALSO       REV  2      KKMOD     REV  3     STKMOD    REV  5
 APLOGIC    REV  14*    LALTMOD   REV  2     STKPROC   REV  9*
 APMODING   REV  9      LATAP     REV  2     SYMLIMF   REV  7*
 APRESET    REV  4      LINIPOL   REV  3     SYMLIMH   REV  2
 ASYLIMI    REV  9      LMACHMOD  REV  3     THETAMOD  REV  1
 ASYLIMH    REV  2      LOCALCON  REV  14    TRIMINTG  REV  2*
 ATTRANS    REV  1      LOCALVAR  REV  11*   TRMLOGC   REV  3
 BDOTMOD    REV  1      LOGICALS  REV  9    #VARGAIN   REV  20*
 BETAMOD    REV  1      LONAP     REV  4     WCONENT   REV  6
 CASAFLT    REV  1      LPHIMOD   REV  2     WINGAINS  REV  16*
 CASCALL    REV  1      LPSIMOD   REV  3*    WINGLMOD  REV  3
 CASIUFLT   REV  1      LTHETMOD  REV  2     WINGPOT   REV  4
 CASTKPRC   REV  3      LTRNCOMP  REV  1     XDACMOD   REV  1
#CLAWS      REV  22*   #MATHLIB   REV  10*   XMODINT   REV  15*
 CLCONEXT   REV  5*     MISCONS   REV  5     YAWCL     REV  7*
#CLDATA     REV  16     MISENPRC  REV  2     YCONENT   REV  4*
 CLEQU2B    REV  15*    MLCMOD    REV  4*    YCONEXT   REV  3*
#CLFILTR    REV  13*    NLIME     REV  3     YCONS     REV  10*
#CLIFACE    REV  4      NYCMOD    REV  6*    YDIRCL    REV  1
 CLLOGIC    REV  9      OUTDATA   REV  4     YSASCL    REV  7*
 CLMODING   REV  5      OUTXDAC   REV  1
 CLVARS     REV  5*     PANGMOD   REV  1
 CLVENT     REV  17*    PARABESR  REV  4
 CTRMOD     REV  2      PCASCL    REV  4
 DACEXT     REV  1      PCONENT   REV  6
 DISCMOD    REV  10*    PCONEXT   REV  2
 DZONESR    REV  2      PCONS     REV  14*
 ELSE       REV  2      PDIRCL    REV  1
 ENDIF      REV  2      PGTABLE   REV  9*
 EOALARM    REV  3      PITCHCL   REV  2
 EOAPFLG    REV  1      PLIME     REV  3
 EOBADS     REV  4      PRDIRSAS  REV  4
 EOCIPMS    REV  1<     PSASCL    REV  3*
 EOCIPRV    REV  4     #RAVCLAWS  REV  15*
 EOCLRM     REV  3      RAVENG    REV  6*
 EODAPMOD   REV  1      RAVILOCK  REV  1
 EOSTPFLG   REV  1*     RAVKKMOD  REV  9*
 FADER      REV  5*     RAVLAW    REV  6*
 FILTADV    REV  7*     RCONENT   REV  2
 FILTCALL   REV  2      RCONEXT   REV  1
 FILTER     REV  1      RCONS     REV  12*
 FLAGADV    REV  4*     RDIRCL    REV  1
 FLAPMOD    REV  5*     RESTINT   REV  4*
 GAINMOD    REV  2      RICMOD    REV  5*
 GANSWPRC   REV  4*     ROLLCL    REV  7*
 GENCALL    REV  1      RSASCL    REV  3*
 GENFILT    REV  1      SENFILT   REV  3*

                               LOG:    15:17:16  05/23/78
```

team. A key difference in these tests was that they were carried out in the environment of a fully integrated and nominally operational system on the NASA iron-bird facility.

Software verification and validation testing covered the following functions:

(1) Control Laws

(2) Executive

(3) Computer I/O

(4) Computer Redundancy Management

(5) Sensor Redundancy Management

(6) In-Flight Self-Test

(7) Preflight Test

(8) Displays/Controls

(9) Primary/Bypass System Transfer Laws

(10) Downlink

A description of results representative of these categories of tests is given in the following sections.

5.5.1.1 <u>Control-Law Verification</u>

These tests broke down into two categories: those that were essentially single-string tests, and those that involved multicomputer operation. Single-string testing dominated the control-law verification. Figure 5-23 illustrates the control-law software verification test strategy. The software functions are isolated into single-input single-output modules where possible. Then, using test access points built into the program, the individual functions are verified against the software specification. For nonlinear functions, the entire function is tested in a dynamic sense by sweeping the input over a range larger than the function expects, and then examining the output. In the F-8 program, all such functional test results were automatically plotted from a real-time data-acquisition system.

The nature of software makes it imperative that all functional relationships be examined using an automatic cross-plot system, because a software fault may affect only one I/O point. High visibility is required of all software operations.

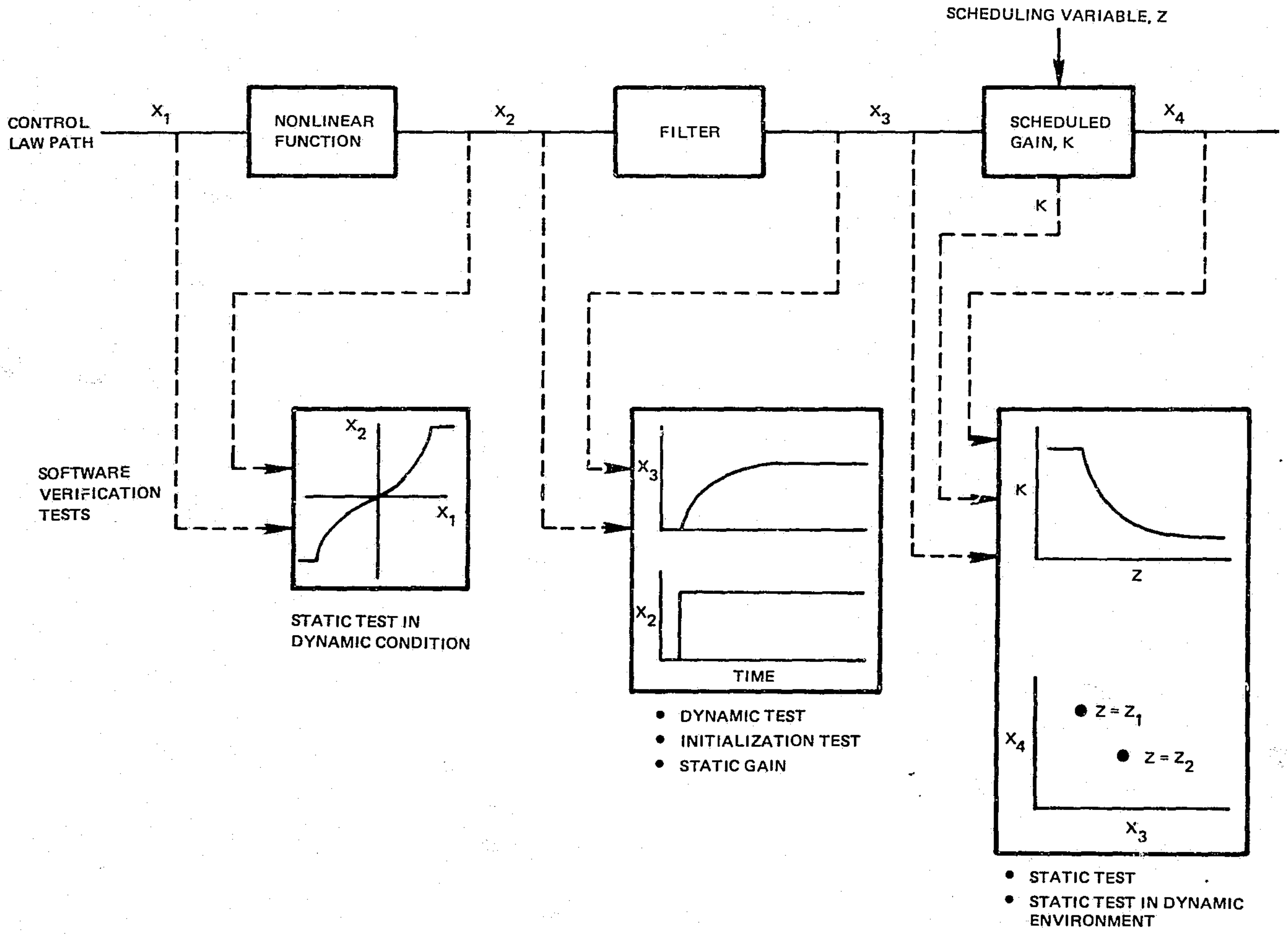


Figure 5-23. Single-string control-law software verification.

Dynamic elements, such as filters, are exercised in a dynamic environment, and responses are compared with independently computed responses based on the desired s-domain (continuous domain) characteristics. Static-sensitivity and initialization laws are verified as well. For functions that have more than a single input or output, a mesh test must be run to verify the multiple functional relationships. These static tests were again actually performed dynamically in the F-8 DFBW verification. Figures 5-24, 5-25, and 5-26 present actual F-8 DFBW software verification results for the three types of test shown in Figure 5-23.

A static test of each major flight-control mode was then performed. This is illustrated in Figure 5-27, and does not differ from techniques used in analog systems. In a digital computer having fixed-point arithmetic capability only, the static test must verify that no overflow can occur for the operating range of all variables. In the case of the F-8 digital computer, which contained floating point hardware, static tests were performed for only a single set of initial conditions.

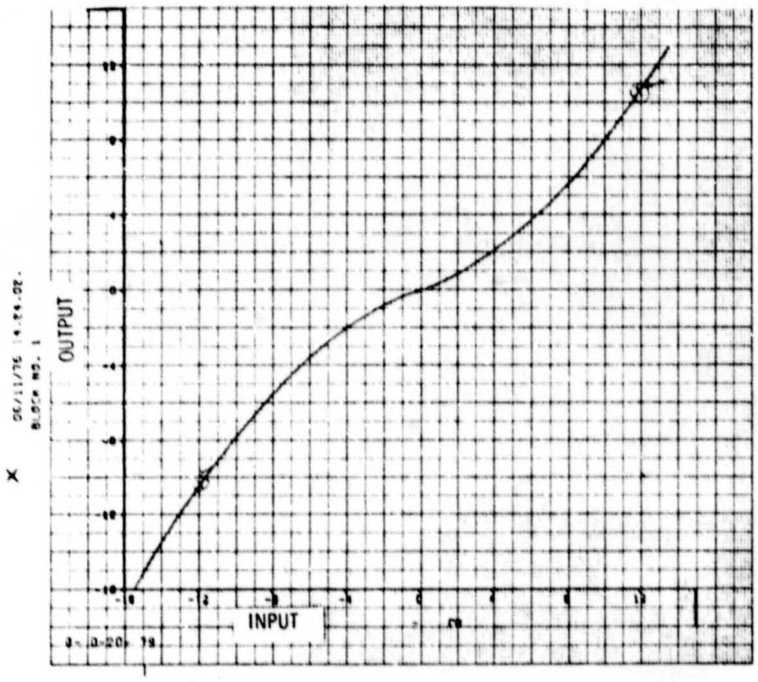


Figure 5-24. F-8 DFBW parabolic shaping verification result.

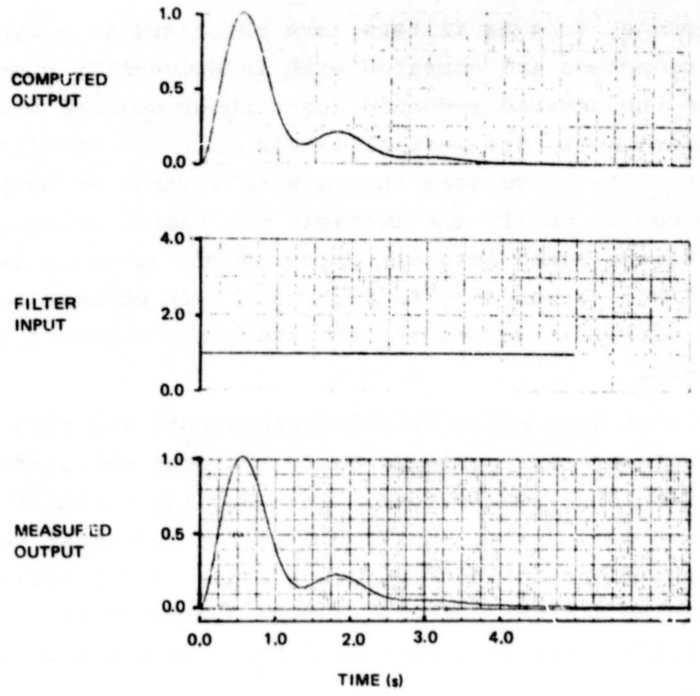


Figure 5-25. F-8 DFBW filter verification result.

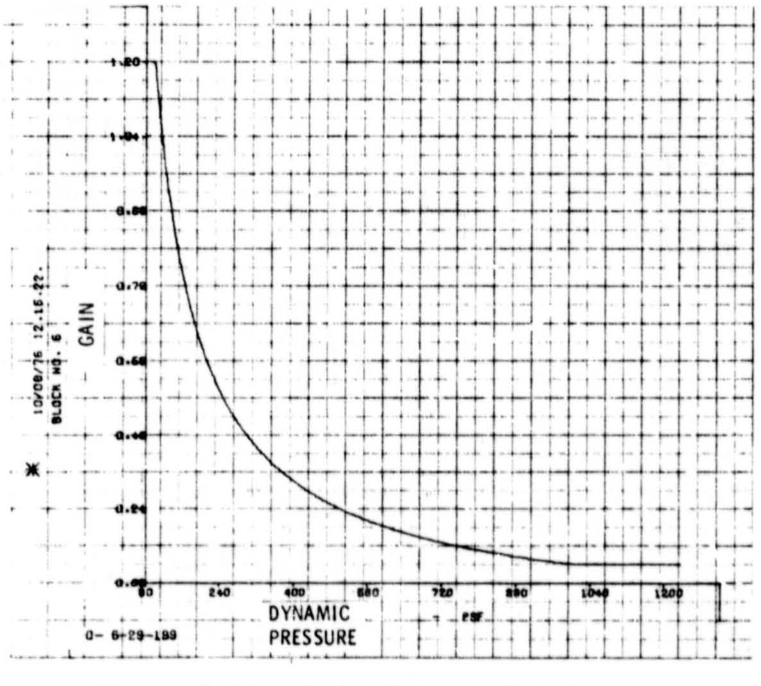
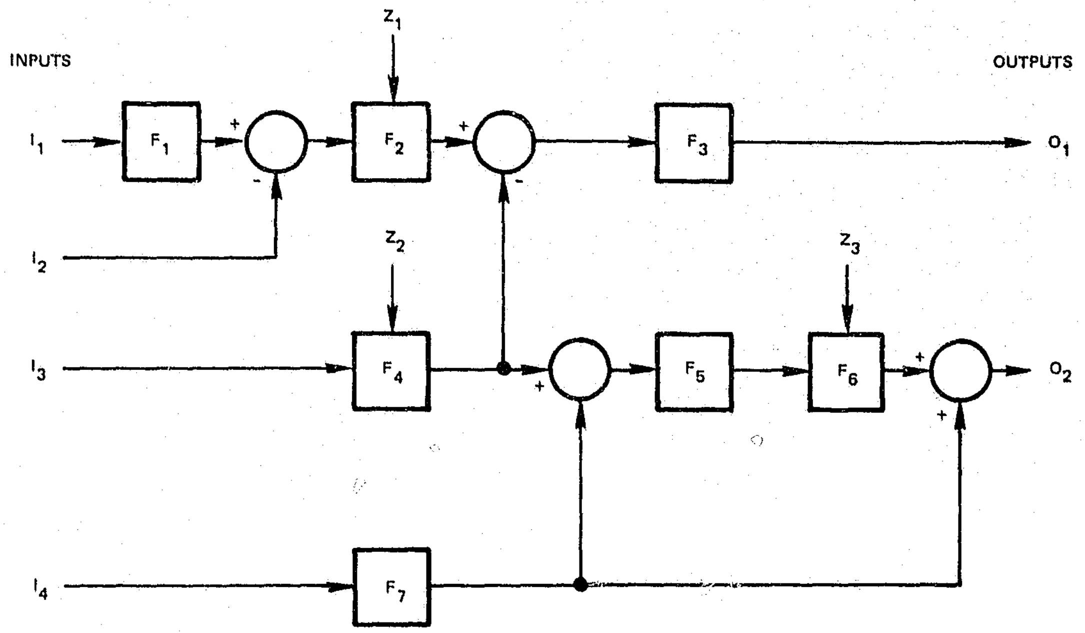


Figure 5-26. F-8 DFBW scheduled gain verification result.



FIX STATICALLY
I_1, I_2, I_3, I_4
Z_1, Z_2, Z_3
COMPUTE
O_1, O_2
COMPARE
O_1, O_2 MEASURED WITH O_1, O_2 COMPUTED INDEPENDENTLY FROM SOFTWARE SPECIFICATION REQUIREMENTS

Figure 5-27. Major pallet assembly test configuration.

In the F-8 control-law validation, some limited open-loop dynamic tests were conducted, which examined the dynamic relationship between the input and the output, without closing the loop through the actuators and airplane simulation. The responses were then compared with independent computations.

Control-law software validation consists primarily of performing closed-loop dynamic tests. These include obtaining closed-loop dynamic responses and comparing them with independent computations. In the case of the F-8 DFBW system, these responses were also compared with an independent control-law prototype program. This is illustrated in Figure 5-28. These tests are identical to those performed on analog systems. Actual system responses were obtained on the iron bird as in path A. In path B, the same aerodynamic model was used, but an independent real-time control-law program and software models of the actuators were used. In path C, a nonreal-time analysis program was used to generate predicted dynamic responses.

Figure 5-29 shows an actual comparison of the responses from the iron bird and all-digital simulator as described in Figure 5-28.

The control-law verification and validation tests were quite time consuming, but insofar as things are ever straightforward, they could be classified as relatively routine. These tests were performed essentially as they would have been if the system had been single string. Figure 5-30 illustrates the fact that each of the channels produces an output which matches that of its partners. The digital commands of the three computers are in fact identical to the last bit. The verification and validation task was substantially eased by the fact that test access had been built into the flight software.

5.5.1.2 Sensor Redundancy Management (RM) Verification

The sensor RM software contains many modes of operation, depending on the multiple sensor status. Each sensor triad or pair is processed by the RM software each 20 milliseconds, and the algorithm configuration may change in that period of time. This means that a very precise and repeatable verification method was needed. Because of the large number of sensors, the test needed to be automated as well. Table 5-5 lists the tests that were required for the analog sensor RM software.

The method developed to accomplish this software verification is illustrated in Figure 5-31. Within the real-time simulation is a multisensor model which can represent various sensor qualities as listed. The multisensor model is under real-time operator control, and was also programmed to generate a particular pattern. The multisensor output is then routed to the digital system in the iron bird.

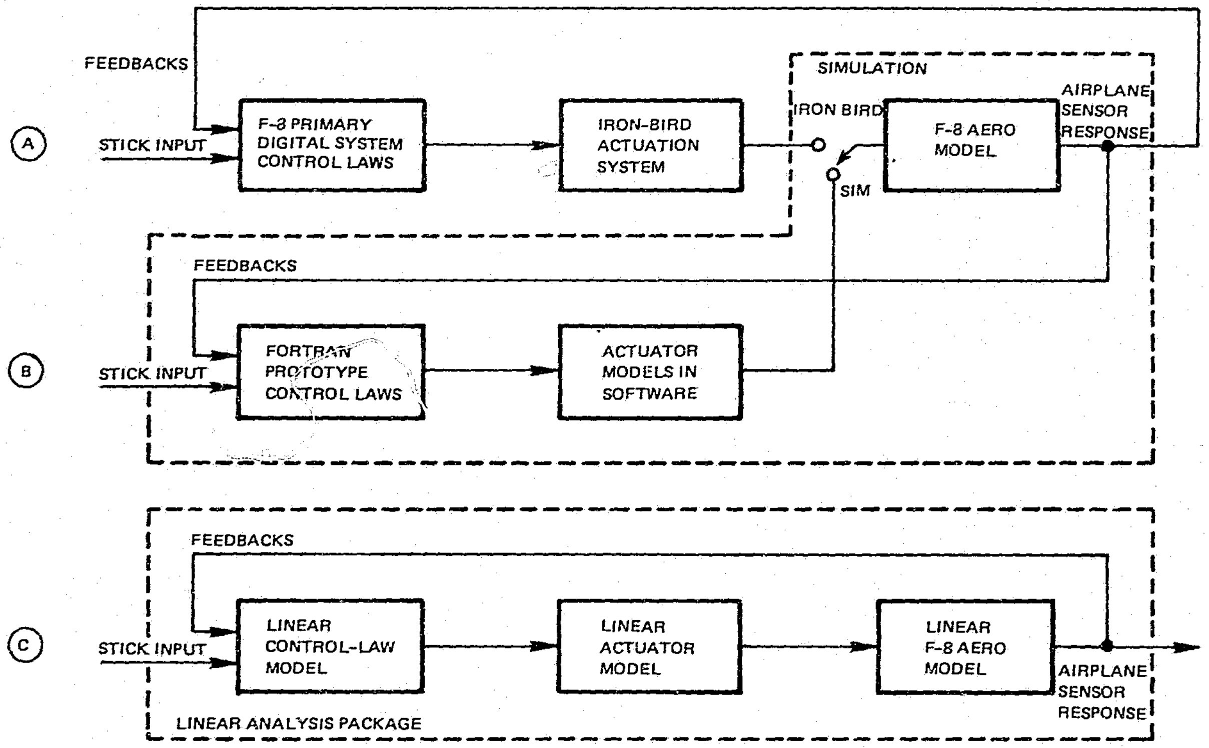
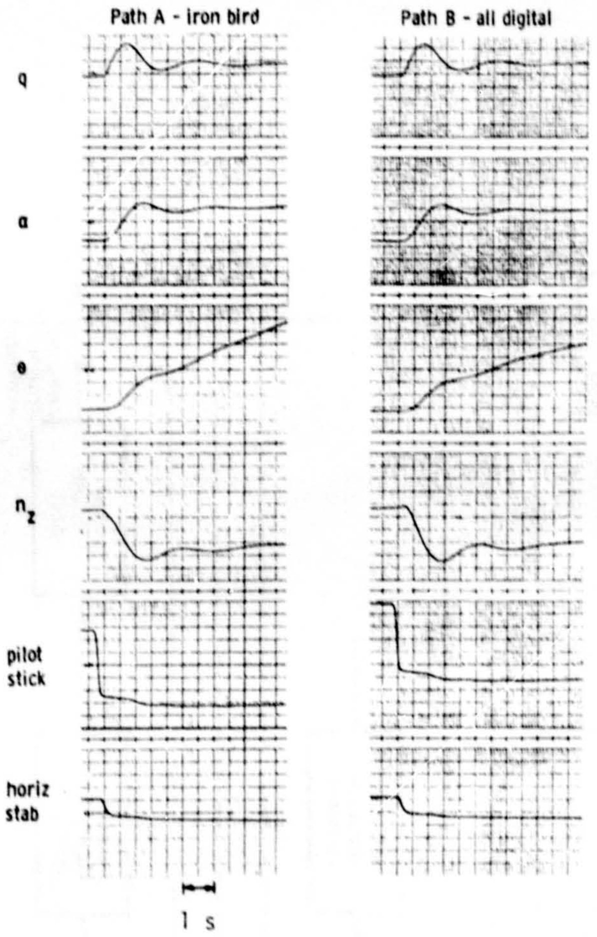


Figure 5-28. Dynamic control-law validation.



ORIGINAL PAGE IS OF POOR QUALITY

Figure 5-29. Iron-bird and all-digital simulation dynamic responses.

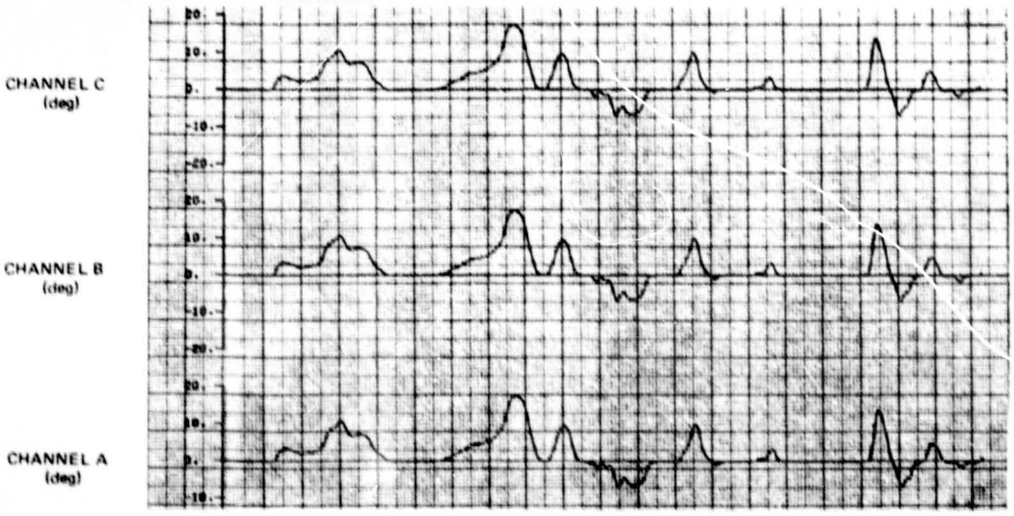


Figure 5-30. Triplex aileron command tracking.

Table 5-5. Analog sensor RM tests.

Test Category	Type of Tests
Triplex	Verify no failure for transient < window width.
	Verify correct midvalue selection.
	Verify first trip level.
	Verify proper downmode to averaging.
	Verify second trip level.
	Verify proper default value.
	Verify annunciator warning and log entry.
	Verify resulting inhibits/activations.
Duplex	Verify correct averaging.
	Verify trip level.
	Verify proper default value.
	Verify annunciator warning and log entry.
	Verify resulting inhibits.

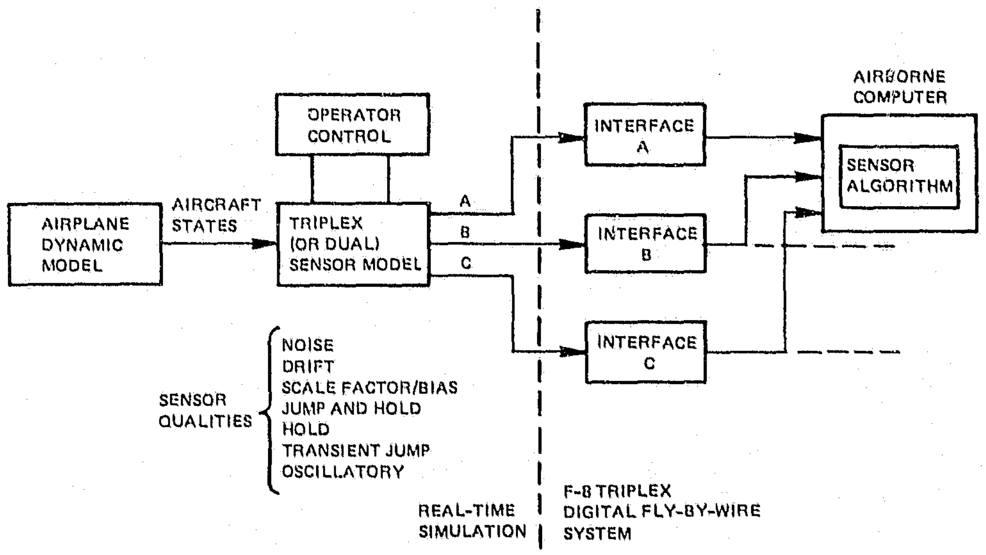


Figure 5-31. Real-time redundant sensor model.

The pattern generated by the sensor model is shown in Figure 5-32. Six separate test segments are run in approximately 35 seconds. This test pattern completely exercises the sensor-redundancy-management algorithm and verifies all of the functions listed in Table 5-5. Test Segment No. 1 verifies operation under provisionally failed conditions; Segment No. 2 verifies the selection algorithm for unfailed sensors; Segment No. 3 verifies midvalue selection, and Segments No. 4 through 6 verify fault-detection and isolation operation.

Because of the amount of logic in the sensor RM algorithms, an automated test was necessary to exercise all the software branches. This is extremely critical because a branch of software that escapes execution during the testing may contain an error that may cause the entire program to fail. Thus, one rule of software verification is: assure all software paths are executed at least once during testing.

5.5.1.3 Miscellaneous Single-String Software Verification

Several other software modules were tested without regard to the fact that they were being executed in a multicomputer system. Modules that fall into this category are in-flight self-test and preflight self test. Verification of these modules followed a simple pattern. It was first verified that no faults were declared in normal operation. Then, one at a time, faults were introduced of the type that were to be detected by these algorithms, and error logs were examined to see that the errors were in fact detected. Figure 5-33 displays an actual test result from the F-8 DFBW in-flight self-test program verification. In this example, the memory sum check function was tested by deliberately altering the contents of memory. Proper fault detection and proper response to the fault were actually determined in this test. These tests were accomplished for every self-test function, and again, were performed in a single-string manner.

Similar techniques were applied to the downlink routine, the executive, and to the special computer-input-panel-initiated routines. In summary, then, much software verification and validation can be accomplished as if the system were single string, provided those functions have modular independence from the redundancy management algorithms. This quality becomes more important when the time comes (and it always does) to make a change to the applications software.

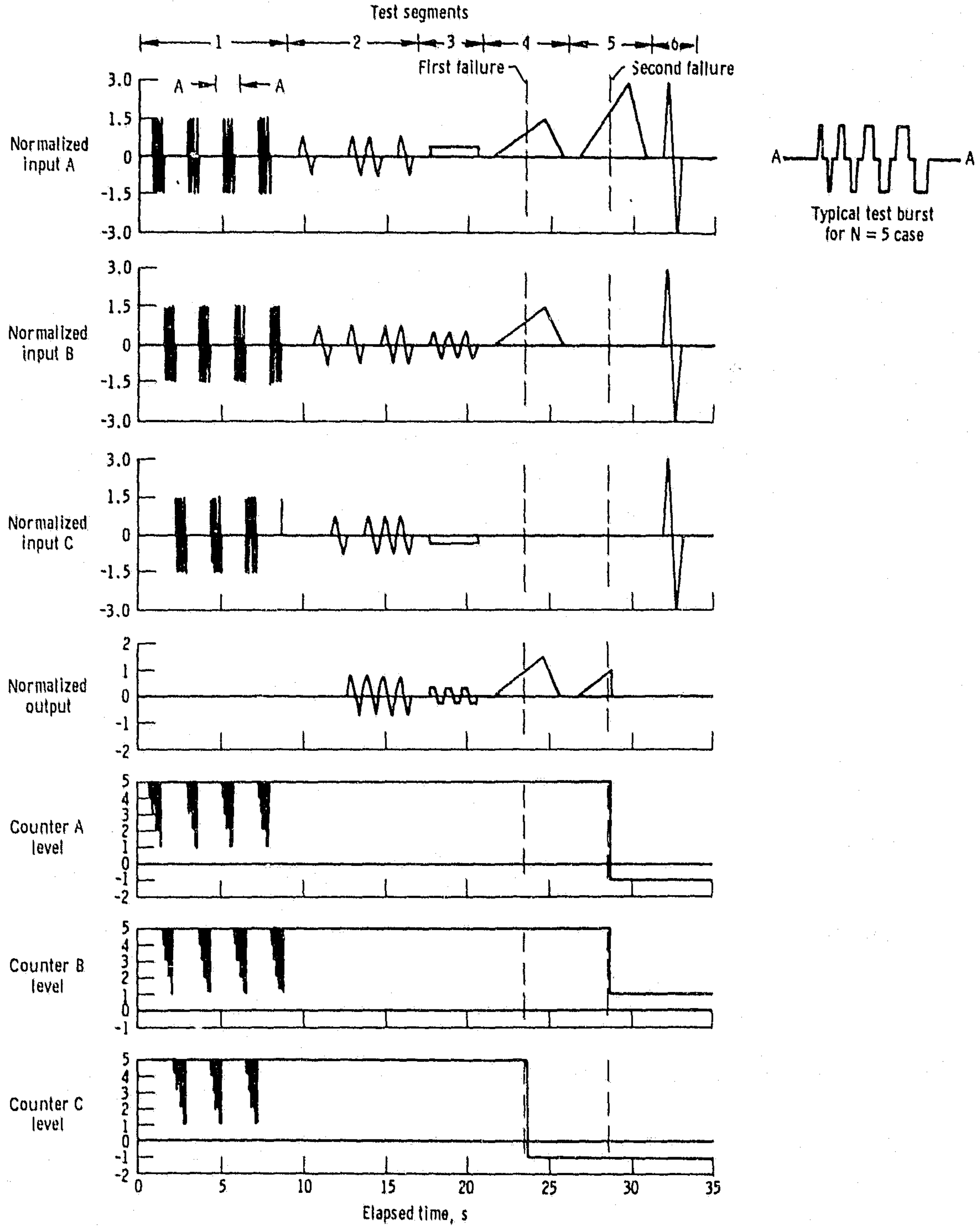
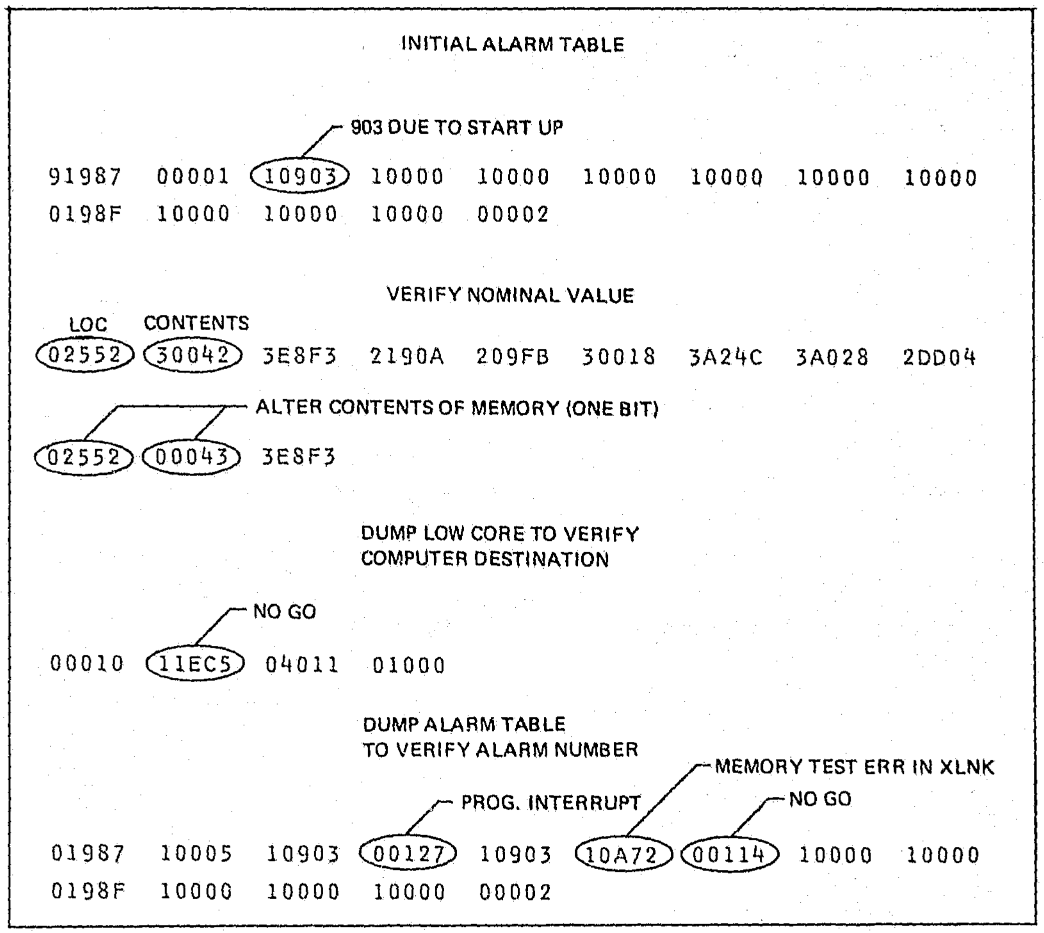


Figure 5-32. F-8 DFBW RM sensor model pattern.


INITIAL ALARM TABLE
903 DUE TO START UP
VERIFY NOMINAL VALUE
LOC
CONTENTS
ALTER CONTENTS OF MEMORY (ONE BIT)
DUMP LOW CORE TO VERIFY COMPUTER DESTINATION
NO GO
DUMP ALARM TABLE TO VERIFY ALARM NUMBER
MEMORY TEST ERR IN XLNK
PROG. INTERRUPT
NO GO

```
91987  00001  10903  10000  10000  10000  10000  10000  10000
0198F  10000  10000  10000  00002

02552  30042  3E8F3  2190A  209FB  30018  3A24C  3A028  2DD04

02552  00043  3E8F3

00010  11EC5  04011  01000

01987  10005  10903  00127  10903  10A72  00114  10000  10000
0198F  10000  10000  10000  00002
```

Figure 5-33. Memory sum check test example.

5.5.2 Multicomputer System and Software Verification

This category of verification and validation is the most critical, because the integrity of the entire flight-control system hinges on the ability to realize the theoretical reliability figures of redundant channels. In order for a redundant system to truly provide the theoretical MTBF, the probability that the entire system will fail due to a common mode fault must be smaller than the probability that successive independent faults will disable the system. At the present time, there is no known way of computing the probability that such a common mode fault exists. This means that design practice and verification testing must be relied on to produce a system that has a high degree of immunity to common mode faults.

By its very nature, verification and validation testing of the multicomputer system emphasizes abnormal operation because normal operation reveals nothing of the true character of the system. The fact that a triplex system operates in a synchronized condition for many hours does not in any way indicate that it can survive even the most elementary fault or that it can provide any more fault tolerance than a single-channel system. In fact, with regard to multichannel performance, normal quiescent operation yields data only on the nuisance fault statistics of the system.

The multichannel system validation test approach is listed in Table 5-6. The tests fell into four major categories and are described according to purpose and content in the figure. The first category of testing established nominal operation. The second category of test involves the systematic verification of the Failure Detection, Identification, and Recovery (FDIR) software, and also the Failure Modes and Effects Test (FMET). There is overlap in these tests. The stress tests are conducted in order to evaluate system operation in configurations and conditions which fall outside of the controlled verification tests. In many cases, this involves reducing system operating margins to zero to establish ultimate breakdown boundaries.

The piloted fault injection tests seek to verify that normal pilot response to fault conditions do not adversely couple with automatic FDIR actions. It is also essential that the system response to real faults, which occur during the test program, be analyzed. Each such fault is an extremely important test because real faults occur in circumstances and sequences not always considered in formal testing. In the F-8 program, each such case was intensely analyzed to evaluate the system FDIR. As it turned out, there was ample opportunity to observe the FDIR process for natural faults, due to early problems with computer reliability.

The actual experience gained during these tests, and the implications for future applications are described in the following sections.

5.5.2.1 Unfailed System Performance Tests

The first performance measure of interest was that of synchronization capability. The parameters of interest were the amount of time required for all channels to acquire sync, and the skew between channels as they exited the sync routine. Figure 5-34 illustrates the

Table 5-6. Multichannel system validation test approach.

Test Category	Purpose of Test	Types of Tests
Unfailed system performance tests	• Establish nominal multi-channel performance	• Synchronization • Timing • Intercomputer communication integrity
Fault detection, isolation, and reconfiguration (FDIR)	• Verify correct FDIR software response to inserted faults. • Verify no common mode faults for inserted single-channel faults.	• Module verification • Failure modes and effects tests
Stress tests	• Establish operating margins for extreme operating conditions. • Verify no common-mode faults for catastrophic single-channel faults.	• Timing overload • Phantom job • Random power transients • Parametric variations • Induced operator errors
Piloted fault injection test	• Verify no adverse pilot coupling with FDIR logic. • Demonstrate fault tolerance.	• FMET subset • Subsystem catastrophic faults

measured results of the sync acquisition tests. The primary system was operated in a closed-loop mission simulation for long periods of time, after which internal test counters were examined. The distribution of "look loops" shows that the channels acquire sync very quickly. The distribution did not change when the test was repeated on the flight system with a different set of flight computers.

Synchronization skew was measured using an oscilloscope and was found to be highly variable in the original software release. This led to the discovery that the synchronization job was being scheduled at a point that was subject to timing variations, and led to a software modification. The synchronization skew was found to be steady and typically less than 10 microseconds with the modified program.

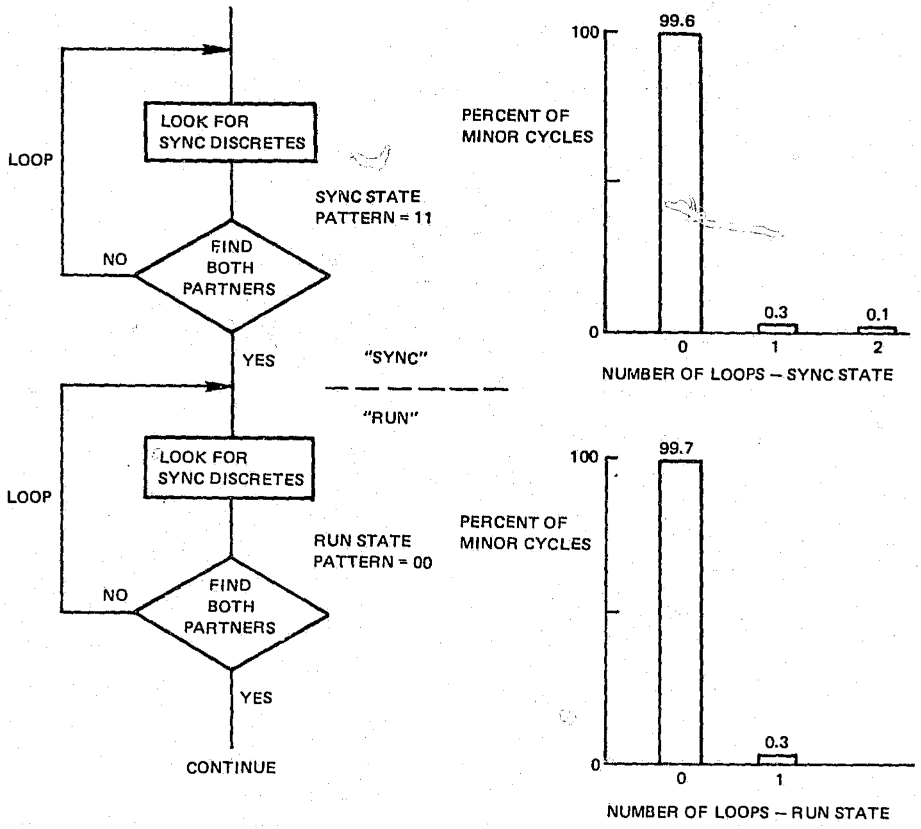


Figure 5-34. Synchronization performance.

These two tests produced insight into system operation that was not possible from cursory operational performance observations. Visibility into the internal operation of the logic was necessary to gain this understanding.

System timing is another performance measure of interest. Special test software was used to measure the execution times of major modules of the program. This measurement technique is illustrated in Figure 5-35. A special software routine computed execution time by measuring elapsed time and then subtracting overhead time. Timing figures were compiled over long periods of time, with both the average and maximum times noted. This data was used to establish the maximum timing load of the flight program under normal conditions. Figure 5-36 presents the data taken on the software release used for the first F-8 DFBW flight.

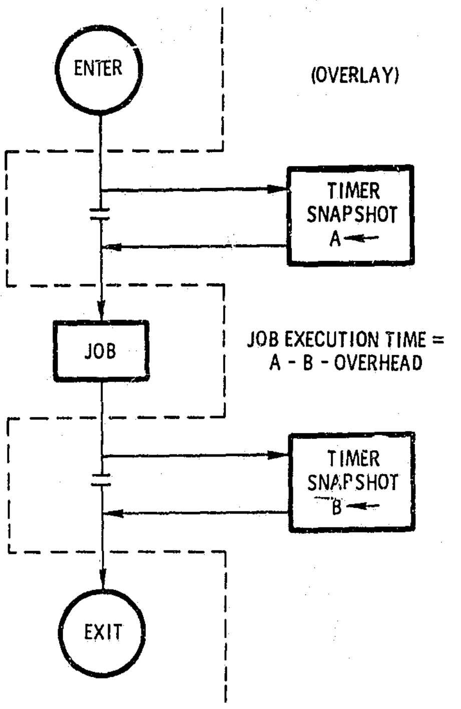


Figure 5-35. System timing technique.

In order to determine the quality of intercomputer communication and I/O, internal logs were examined at the end of the operating day. The following parameters were examined:

(1) Number of restarts.

(2) Number of I/O errors.

(3) Number of sensor miscompares or failures.

(4) Number of alarms indicative of faulty intercomputer communication.

Even for the very early releases it was observed that the normal and usual number for each of these parameters was zero. This was not to be the case, however, in later stress testing. The conclusion drawn from these unfailed performance tests was that the multichannel system operated with a high degree of integrity under quiescent operating conditions.

During this early test phase it was frequently observed that a channel would sporadically lose sync, or that a channel, or occasionally

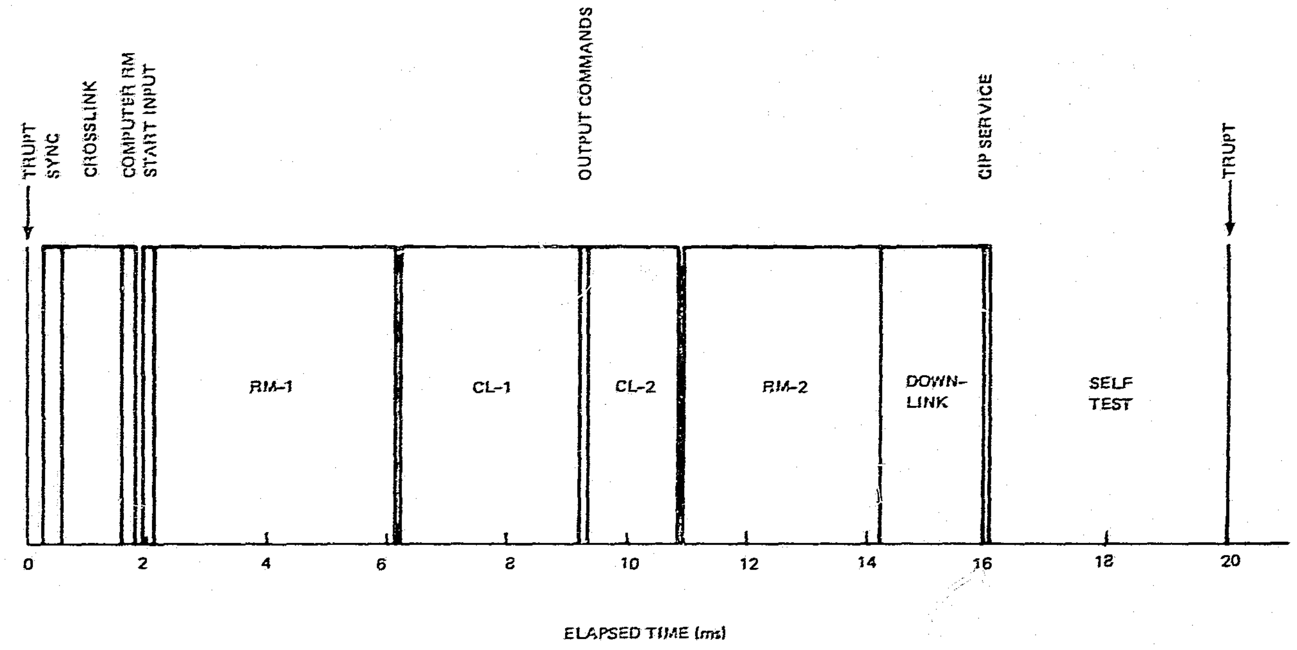


Figure 5-36. F-8 DFBW flight release timing for worst case, all modes engaged.

all three channels, would fail to achieve sync at turn on. This implicit testing was to generate the majority of software design changes, as is described later.

5.5.2.2 Failure Modes and Effects Tests (FMETs)

FMETs are designed to systematically verify that the digital system is fail-operational after a single disabling channel fault, and fail-passive (to CBS) after a second channel failure. These tests were performed after isolated software modules had been verified for correct functional operation. Table 5-7 summarizes the FMET strategy for both hardware and software faults.

Table 5-7. FDIR test procedure.

Inserted Fault	Verification Steps
Transient fault in one or more channels	• Successful restart • Continued operation • No permanent faults • No adverse surface motion
Permanent fault in one channel	• Restart attempted • Channel declared hard failed • Proper internal reconfiguration • Annunciation to pilot • System operational • No adverse surface motion
Permanent, like faults in two channels (successive)	• Restart attempted • Auto transfer to bypass system • Proper internal status • No adverse surface motion

The selection of the fault matrix presents a very difficult problem for a system such as this. It is apparent that a component-by-component or wire-by-wire open/short fault test matrix is impossible. The parts count and number of failure modes precludes even a modest attempt at verifying FDIR operation in this manner. Thus, the strategy used was to inject faults at the functional level and judge integrity on the basis of the FDIR response to these functional faults. An important thing to note at this point is that the system must be designed at the outset to be testable. If the FDIR scheme is extremely sophisticated, involving operations on a large amount of data and a large number of intricate steps to detect and isolate faults, it is quite possible that the system will be unverifiable. That is, the number of tests required to establish correct response, even for those faults that can be thought of, exceeds the test capability available.

The F-8 DFBW FDIR design and FMET approach are based on the following assumptions:

(1) The ability of a channel to maintain synchronous operation and to communicate with its partners is a primary indicator of channel health.

(2) A relatively small amount of information exchanged between channels can be used as a further basis for this determining channel health. This information can be thought of as vital signs.

(3) Fault detection, isolation, and reconfiguration can be based on the manifestation of a fault, rather than on the detection of the fault itself.

(4) The restart mechanism will provide automatic transient fault protection without having to identify the source of the fault.

These assumptions dramatically reduce the test matrix. It is now possible to approach the failure modes and effects tests in the following manner:

(1) Verify that the FDIR response to abnormalities in the small number of vital signs is correct.

(2) Verify that all built-in test hardware produces warning signals to the FDIR for the types of faults they are designed to detect.

(3) Verify that the restart process restores normal operation for transient abnormalities in the vital signs or for the built-in test warning signals.

(4) Verify that the restart mechanism will suspend its attempt to restore operation for unrecoverable faults.

(5) Finally verify the fail-operational fail-safe performance to all permanent faults.

These assumptions and observations produced a relatively modest test matrix which provides an extremely thorough coverage of the functional validation of the FDIR process. The test matrix is tabulated in summary form in Table 5-8. No assumptions were made relative to the similarity of channels. That is, faults were induced for all combinations of two channels.

Table 5-8. FDIR test matrix.

Location of Injected Faults	Approximate Number of Faults	Types of Faults Injected
Interface Unit	200	• I/O data lines • Crosslink communication lines • Synchronization lines • Loss of entire card • Loss of entire connector • Loss of power
Encoder/Decoder	400	• Data lines • Strobe lines • Clock lines
Computer Hardware	20	• I/O connector • Spurious interrupts • Loss of power
Computer Software	150	• Faults producing program interrupts • Faults producing machine interrupts • I/O communication errors • Vital sign errors

Breakout boxes were used to insert the majority of encoder-decoder faults and some of the interface unit faults. For most of the IFU fault injection tests, however, special hardware was used which was built-in specifically for the failure modes testing. This hardware provided the capability to break signal paths or to simulate fail-high or fail-low conditions. Specific data paths modified were:

(1) Interrupt lines (open or continuous)

(2) Crosslink transmission

(3) Input data transmission

(4) Input data reception

(5) Synchronization discretes

Figure 5-37 illustrates the mechanization used for inserting sync discrete faults into the system. Figure 5-38 shows how crosslink and data line faults were introduced.

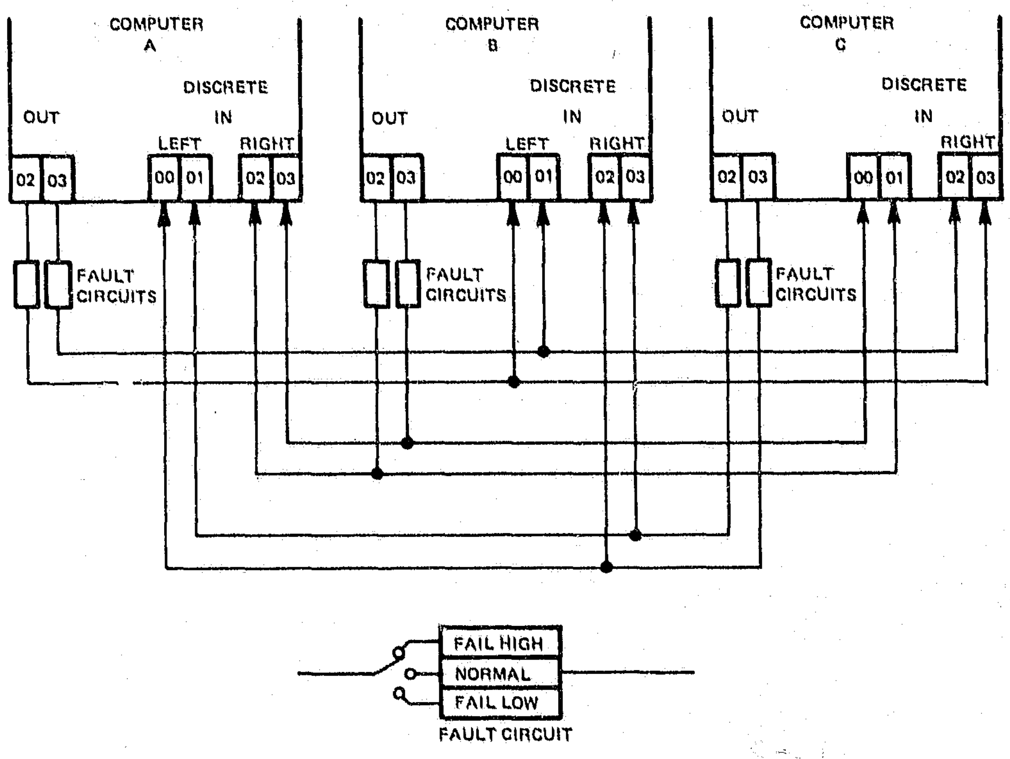


Figure 5-37. Mechanization of synchronization fault introduction.

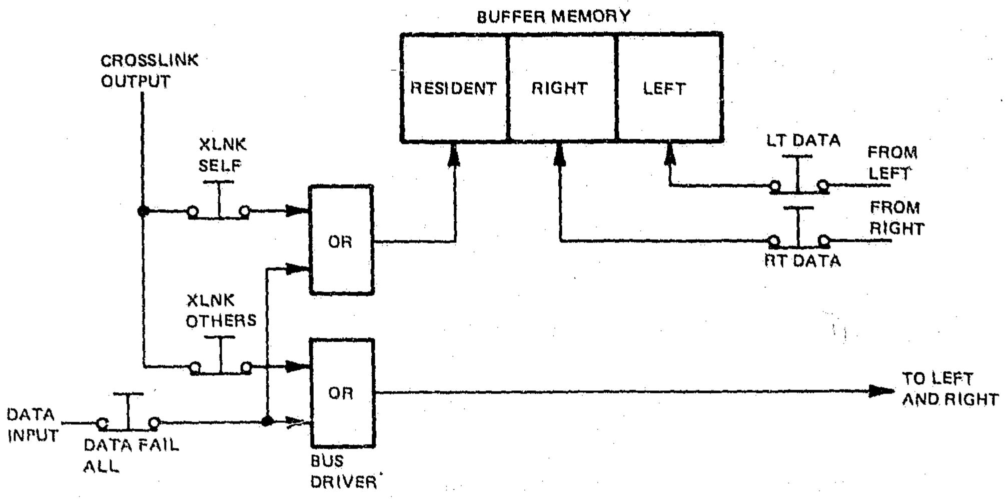


Figure 5-38. Mechanization of interchannel data bus and crosslink failure introduction.

The computer test set was generally used to insert software faults, although some special-purpose software was required to simulate certain faults. Table 5-9 lists in more detail the types of software faults introduced.

The results of these FMETs are summarized in Table 5-10. A remarkably small number of anomalies occurred, with only one being serious: the input data stream double fault. The encoder/decoder faults affects display operation in a benign manner. The key observations are that the fail-operational fail-safe requirement was demonstrated in all but one condition. Although stress testing was to expose several restart recovery problems, it is significant that the system operation was verified to the extent demonstrated. The second item of significance was that all transient faults introduced were survived by the system.

5.5.2.3 Stress Tests

This class of verification exposed more anomalies per test than any other. The stress test procedures are listed in Table 5-11 for each of the test types. These tests were, for the most part, conducted with the full iron bird operational, including actuators. Surface commands were monitored directly, and post-test core dumps were examined

Table 5-9. Examples of simulated software faults.

Faults Introduced	Result
• Illegal-operation code • Privileged instruction • Fixed-point overflow • Significance test • Unnormalized floating-point operation • Store protect violation • Exponent underflow and overflow • Divide check	Program Interrupt
• CPU and memory parity error • Timeout (No-Go discrete)	Machine Interrupt
• Discrepancy in channel-identification tag • Discrepancy in computer time • Discrepancy in computer functional mode • Recognition of restart request • Critical error in crosslink transmission list • Error in self-test program	Direct Restart

to establish recovery or fault sequences. The results of these tests are summarized in Table 5-12.

The stress tests exposed problems not apparent in the more controlled and systematic module verification tests and distinct failure modes and effects tests. Both classes of tests are necessary, of course, but stress testing tends to produce sequences of operation that were not considered in the design phase or produced in the controlled fault testing.

The stress tests exposed problems that existed almost exclusively in the restart recovery process. The modifications improved nuisance-fault immunity substantially without materially affecting fault-detection capability.

Table 5-10. Summary of FDIR test results.

Anomalies
Synchronization
• Continued operation for some sync faults
Input Data Stream
• No CBS down mode for dual input data line loss
Encoder/Decoder
• One type of first fault not detectable
• Software wrap test incorrect
Computer Software
• Some faults detected by means not predicted
Observations
• System was fail-operational for every first fault
• System was fail-safe for all but one second fault
• All injected transient faults were survived

Timing Overload—A common-mode software fault, which causes a program "hang up", is designed to cause a transfer to the bypass system. Temporary overloads in one channel may be recoverable through the restart process. No anomalies were found in these tests.

Phantom Job—This test exposed a software error in the downlink routine and also showed that an asynchronous job could cause interference in the sync program. Asynchronous jobs were ruled out for future implementation. No other anomalies were found.

Parametric Variations—Minimum operating values were determined for key parameters, as shown in Table 5-13. The nominal program values were increased in some cases where margins were too small.

Power Faults—Major problems with the sync and restart software were identified in this test series. An example of the type of fault observed is shown in Figure 5-39, where anomalous surface command

Table 5-11. Stress-test procedures.

Test Type	Procedure
Timing Overload	Increase sample rate until minor-cycle period equals compute time.
Phantom Job	Insert asynchronous job which "drifts" through minor-cycle period, and also overflow job queues.
Parametric Variations	Reduce timing tolerances and fail-counter values during quiescent and power-transient conditions (during restarts).
Power Faults	Interrupt or reduce prime 28-Vdc power in one or more channels.
Induced Operator Errors	Erroneous operation of mode panels and cockpit controls.

Table 5-12. Stress-test results.

Test Type	Result
Timing Overload	• No errors found
Phantom Job	• Error in downlink software exposed • Interference with sync process noted
Parametric Variations	• Minimum acceptance values determined • Margins sufficient
Power Faults	• Exposed major software problems in recovery process • Sync and restart software design errors • Insufficient tolerances on failure counters
Induced Operator Errors	• No system FDIR software errors found • Applications software errors found

Table 5-13. Parameteric variation test results.

Parameter	Final Flight Program Value	Minimum Good*
Consecutive I/O Interrupts	10	2
Consecutive Restarts	10	6
Crosslink Wait Time	200 μs	100 μs
Sync Wait Loops	10-10	6-2
Consecutive Crosslink Errors	10	4

*Under most severe power transient conditions.

behavior was observed on only one of several consecutive power cycles. The problems were caused by a combination of design errors and insufficient tolerances on timeout and fault counters. The modified software demonstrated an ultra-high tolerance to power transients, as shown in Figure 5-40. An uneventful transfer to the bypass system occurred for reduction in prime dc level below the 22-volt set point.

Induced Operator Errors—No system FDIR errors were detected by these tests. This was due to the modular separation of applications software and system FDIR software. A few applications software errors were detected, however.

5.5.2.4 Piloted Failure Modes and Effects Testing

The formal and systematic open-loop failure modes and effects tests are genally conducted under static conditions, that is, under conditions which approximate cruise flight. Although many failure effects can be evaluated independently of the environment in which the system is operating, it is not possible to evaluate all failure effects in this manner. In the F-8 DFBW program, a closed-loop piloted FMET series was performed. Figure 5-41 lists the type of tests accomplished. These tests were performed in the iron bird in a real-time simulation mode with all systems active. The pilot was aware of the fault to be injected and the time it was inserted. This was done so that the pilot could analyze the fault and select conditions where the consequences

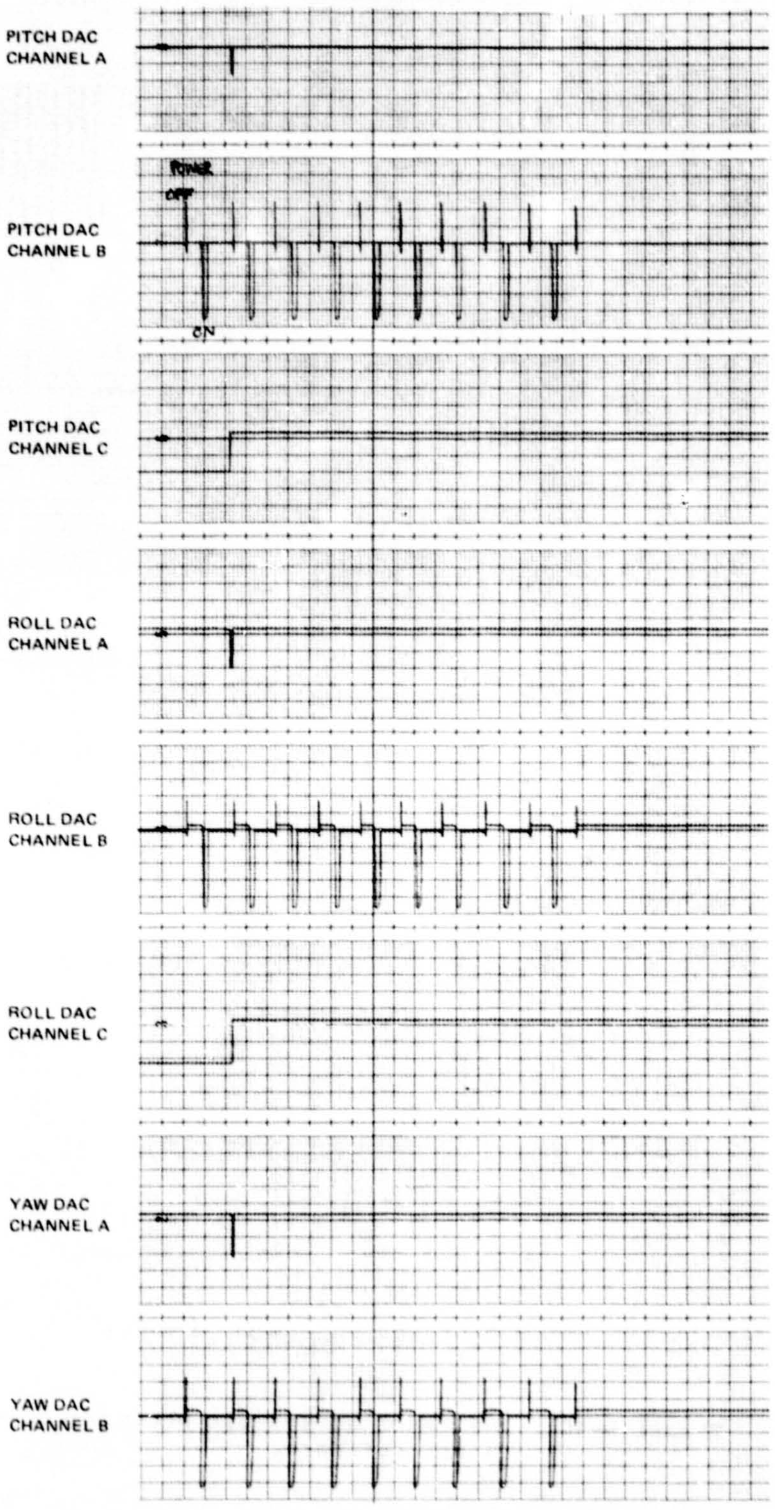


Figure 5-39. Restart anomaly example.

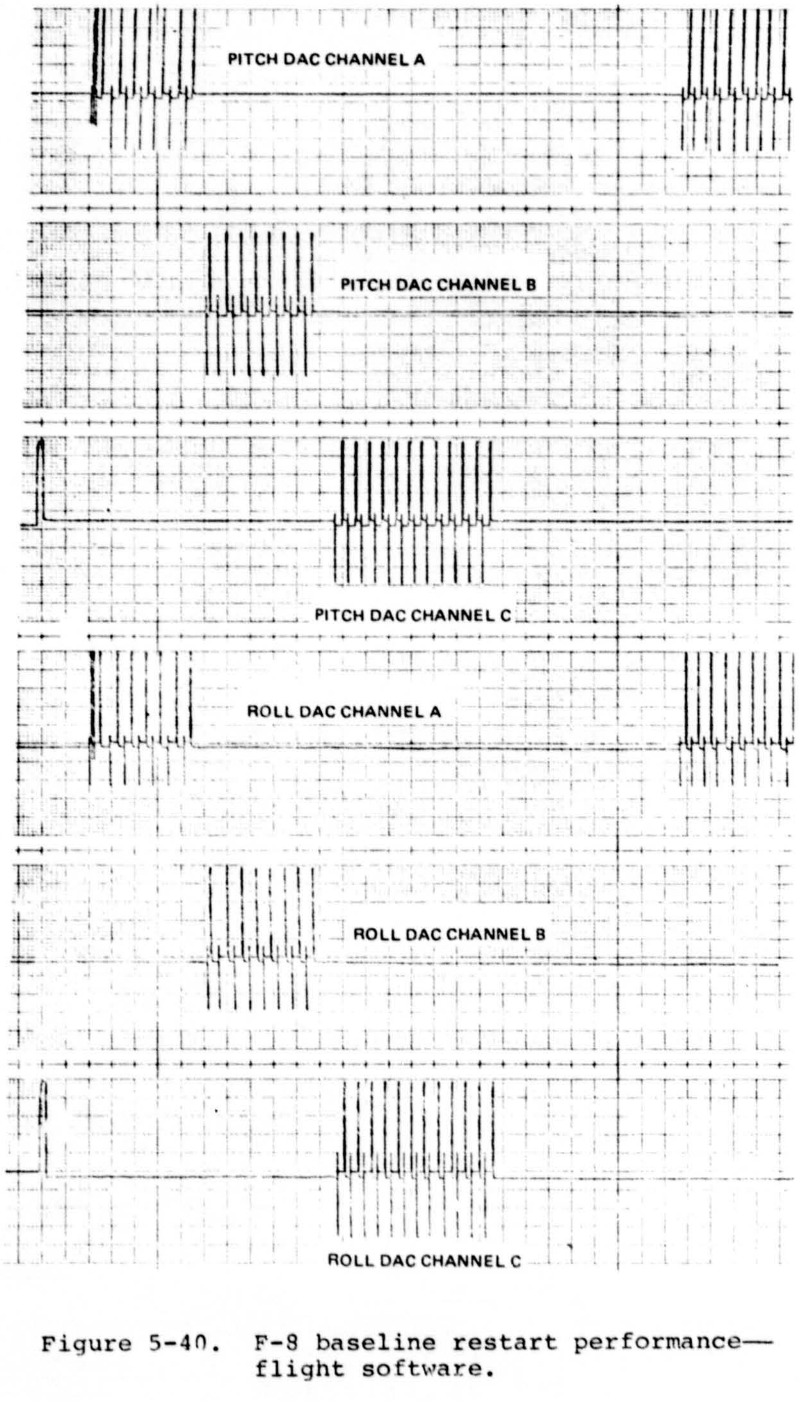
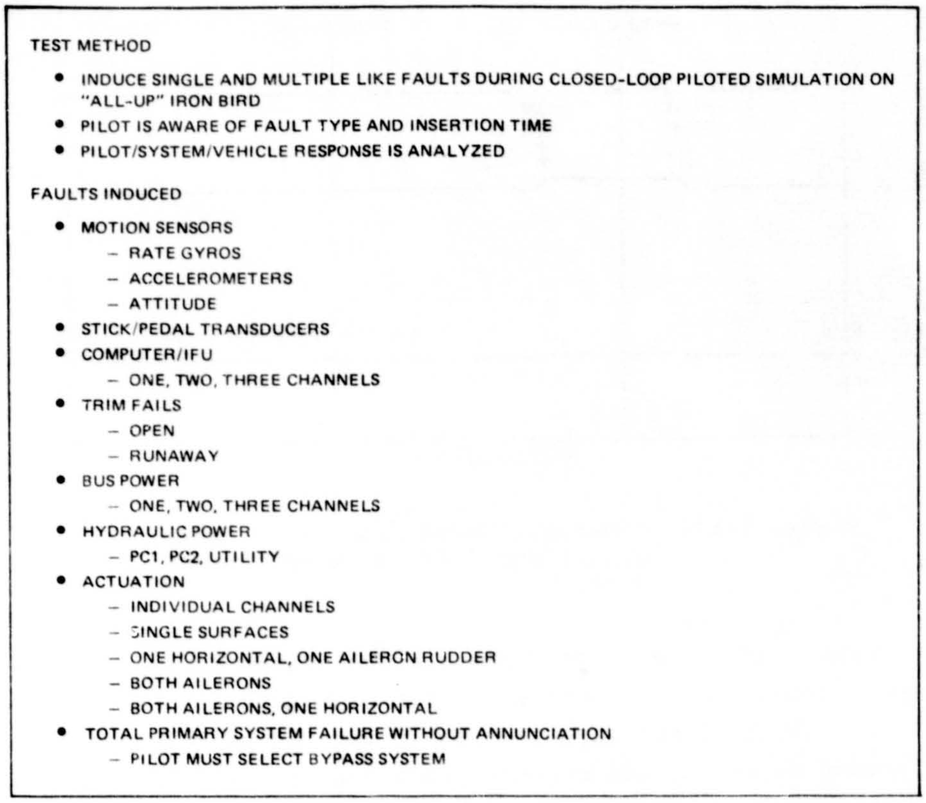


Figure 5-40. F-8 baseline restart performance—flight software.



TEST METHOD

- INDUCE SINGLE AND MULTIPLE LIKE FAULTS DURING CLOSED-LOOP PILOTED SIMULATION ON "ALL-UP" IRON BIRD
- PILOT IS AWARE OF FAULT TYPE AND INSERTION TIME
- PILOT/SYSTEM/VEHICLE RESPONSE IS ANALYZED

FAULTS INDUCED

- MOTION SENSORS
  - RATE GYROS
  - ACCELEROMETERS
  - ATTITUDE
- STICK/PEDAL TRANSDUCERS
- COMPUTER/IFU
  - ONE, TWO, THREE CHANNELS
- TRIM FAILS
  - OPEN
  - RUNAWAY
- BUS POWER
  - ONE, TWO, THREE CHANNELS
- HYDRAULIC POWER
  - PC1, PC2, UTILITY
- ACTUATION
  - INDIVIDUAL CHANNELS
  - SINGLE SURFACES
  - ONE HORIZONTAL, ONE AILERON RUDDER
  - BOTH AILERONS
  - BOTH AILERONS, ONE HORIZONTAL
- TOTAL PRIMARY SYSTEM FAILURE WITHOUT ANNUNCIATION
  - PILOT MUST SELECT BYPASS SYSTEM

Figure 5-41. Piloted FMET series.

might be most severe. In many cases, the same fault was inserted several times to permit the pilot to examine the results under various dynamic conditions.

This phase of testing is very critical. Faults are coupled with pilot response and the total system performance can be assessed in realistic circumstances. This is important not only in uncovering problem areas, but also, if the design is sound, in giving the pilot a measure of confidence in overall system integrity. This is ultimately a critical step in the flight qualification process.

In the F-8 system, serious faults tend not to produce transients in the aircraft motion. For example, successive hard-over faults in the pitch-rate-gyro set are undetectable from vehicle response characteristics (Figure 5-42). In the figure, the aircraft was deliberately mistrimmed before the first and second failures. The hard-over faults produce no vehicle motion. Faults of lesser magnitude and rate do produce slight vehicle motion, but, on the other hand, are handled almost routinely by the pilot.

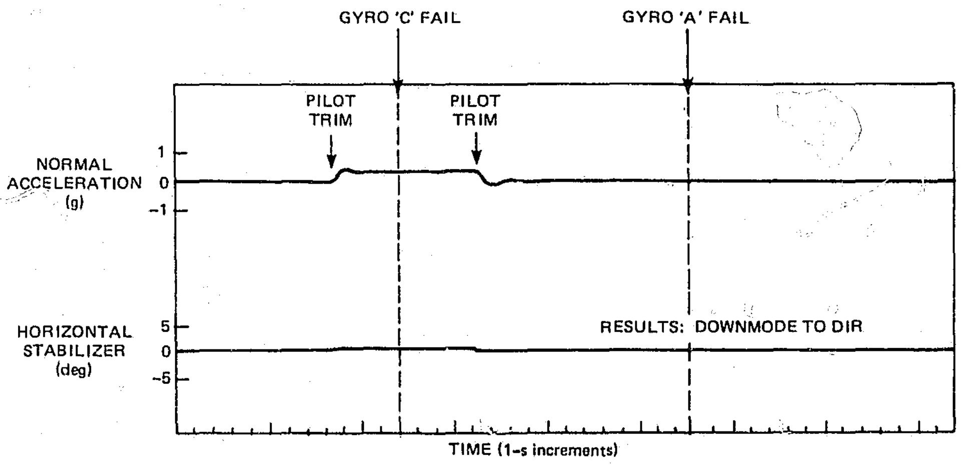


Figure 5-42. Successive hard-over pitch-rate-gyro-induced failures—pitch SAS mode.

The results of these pilot FMETs are summarized in Figure 5-43. Problems were identified and corrected, although very few actually occurred. It must be noted here that the pilot had been involved in the development of the system reconfiguration policy and had evaluated interim configurations prior to the formal FMET series.

The overall test results were significant. For all survivable faults, the pilot was able to maintain control of the aircraft. This was not the case in the Phase I F-8 DFBW program, where a category of faults in the single-channel digital system coupled with normal pilot responses produced some unrecoverable conditions. This was due to detection and reconfiguration processes taking up to 1 second. No such cases exist in the triplex Phase II DFBW system.

PROBLEM AREAS

- OPEN FAILURES ON STICK/PEDAL LVDTs NOT DETECTED QUICKLY ENOUGH (BIAS ADDED)
- SOME ANNUNCIATION OF FAILURES CONFUSING (LOGIC CHANGED)
- RUNAWAY RUDDER TRIM DETECTION TIME TOO LONG (LIMIT TRIM AUTHORITY)
- CHANGE OF STICK TRIM WITH AUTOMATIC RECONFIGURATION REQUIRES PILOT ACTION

OVERALL TEST RESULTS

- NO LOSS OF CONTROL FOR ANY SURVIVABLE FAULT CONDITION
- AUTOMATIC RECONFIGURATION DID NOT COUPLE WITH PILOT RESPONSE
- THE AIRPLANE CAN BE LANDED WITH THE FOLLOWING SURFACES ACTIVE:
  - ONE HORIZONTAL STABILIZER
  - ONE AILERON OR RUDDER (LAKE-BED RECOVERY WITH NO AILERONS)
- FAIL-OPERATIONAL RESULTS FOR ALL FIRST FAULTS
- PILOT NOT REQUIRED TO TAKE UNUSUAL ACTION FOR ANY SURVIVABLE FAULT

Figure 5-43. Results of piloted FMET series.

The independence of all control surfaces allowed the aircraft to be controlled sufficiently well enough for a landing with only one horizontal stabilizer and one roll control surface (one aileron or the rudder) active. All first faults were demonstrated to leave a fail-operational system for the primary flight control modes (some outer loop functions are fail-passive).

A final important result was that for any theoretically survivable fault, it was not required that the pilot take any unusual action in order to maintain control. Naturally, it was required that the pilot occasionally halt induced aircraft rates and retrim to level flight for some multiple-failure conditions. This test series contributed significantly to the project team's confidence that the primary DFBW flight-control system could survive massive faults and still permit the pilot to recover the aircraft. The validity of the tests was further accepted because they were accomplished on a test rig that came very close to duplicating the flight environment.

5.5.3 Piloted-Mission-Profile Testing

These tests consisted of pilot evaluations of the functional performance of the DFBW control system on the iron bird during real-time simulation. These tests served several purposes:

(1) Refinement of control laws for best flying qualities.

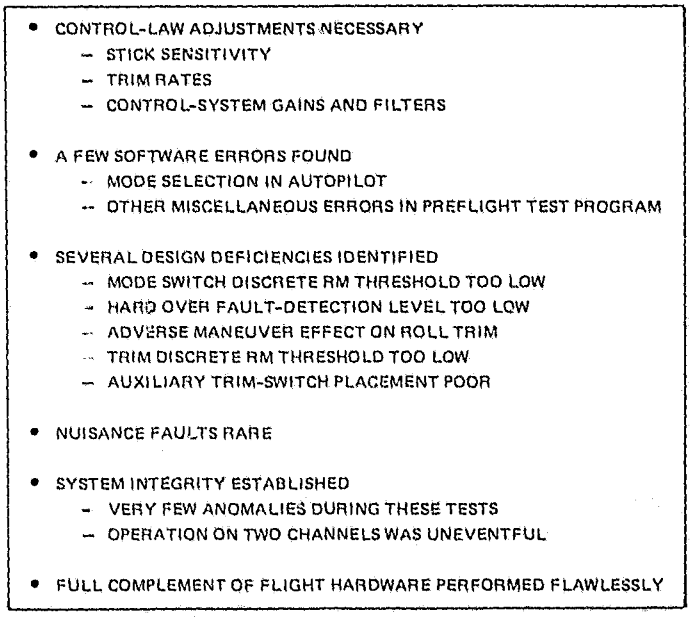
(2) Exposure of functional performance problems due to design or software deficiencies.

(3) Determination of the degree of system nuisance-fault immunity under normal operating conditions.

(4) Demonstration of system integrity to the pilot, and, in fact, to the entire project team, during conditions closely simulating flight.

It is not only important in the test phase to demonstrate that the flight-control system can tolerate faults, but also that over a period of months the system can be relied upon to perform its intended functions without problems. This is another critical step in the flight-qualification process. A system which fails frequently under normal operating conditions does not produce the confidence that it will perform the intended job when needed most. A nuisance fault is more than a nuisance. In a system like the triplex F-8 DFBW primary flight-control system, a nuisance fault is indistinguishable from a permanent hardware failure if it causes loss of a channel or a sensor. The integrity of a system then can be thought of as that quality which makes it both reliabile and fault tolerant. It is a system that can be depended on. The mission-profile tests provide information on the integrity of the system.

The results of these tests are summarized in Figure 5-44. As expected, several control-law refinements were necessary. This type of refinement extended into the flight phase of the program as well. A few software errors were exposed, but they were relatively minor. On the other hand, several design deficiencies were identified. Listed in Figure 5-44 are those found which were significant.



- CONTROL-LAW ADJUSTMENTS NECESSARY
  - STICK SENSITIVITY
  - TRIM RATES
  - CONTROL-SYSTEM GAINS AND FILTERS
- A FEW SOFTWARE ERRORS FOUND
  - MODE SELECTION IN AUTOPILOT
  - OTHER MISCELLANEOUS ERRORS IN PREFLIGHT TEST PROGRAM
- SEVERAL DESIGN DEFICIENCIES IDENTIFIED
  - MODE SWITCH DISCRETE RM THRESHOLD TOO LOW
  - HARD OVER FAULT-DETECTION LEVEL TOO LOW
  - ADVERSE MANEUVER EFFECT ON ROLL TRIM
  - TRIM DISCRETE RM THRESHOLD TOO LOW
  - AUXILIARY TRIM-SWITCH PLACEMENT POOR
- NUISANCE FAULTS RARE
- SYSTEM INTEGRITY ESTABLISHED
  - VERY FEW ANOMALIES DURING THESE TESTS
  - OPERATION ON TWO CHANNELS WAS UNEVENTFUL
- FULL COMPLEMENT OF FLIGHT HARDWARE PERFORMED FLAWLESSLY

Figure 5-44. Summary of piloted mission-profile test results.

With early software releases, design problems caused some nuisance-fault declarations at turn-on, but nuisance sync loss or fault declarations during operation were extremely rare. No nuisance faults occurred with the release of the software that was flown on the first flight. The overall integrity of the DFBW system, the bypass system, and the actuators was positively established during these tests. The systems were operated in real-time simulation throughout the flight envelope and in a variety of maneuvers and dynamic conditions. Anomalies were rare. During periods of time when only two digital computers were available for the iron-bird facility, operations were routinely carried out with no difficulties. Surprise faults introduced during these simulations were handled routinely by the pilot and control was never lost.

These tests resulted in some software and system modifications, but no major system redesign was necessary. The final series of pilot evaluations were accomplished with the actual flight pallet and bypass-system hardware which would be used for the first flight. The performance of the system was flawless. This fact was another critical setp in the flight qualification process.

5.5.4 Aircraft Integration Tests

Final integration tests were accomplished on the F-8C aircraft itself. The total system operating time in the aircraft prior to flight was 400 hours. In addition, over 1000 hours of operating and test experience had been achieved in the iron bird. The tests performed in the aircraft are listed in Figure 5-45. It was not possible to close the simulation of the F-8C around the aircraft itself, so the tests were open-loop in terms of the aerodynamic environment.

The tests listed in Figure 5-45 are all straightforward and were accomplished relatively easily. Some problems were identified during this phase of testing:

(1) Power-transient susceptibility of bypass system during primary-system power cycling.

(2) Scale-factor discrepancies in instrumentation system.

(3) Errors found in preflight test program software in routines not tested on iron bird.

(4) Tolerance adjustments required in several preflight test program routines.

(5) Channel failure frequently caused by computer hardware problems.

(6) Excessive mechanical offset of stick LVDT.

EMI testing was composed for two separate test categories. The first involved the superposition of noise on the source 28-Vdc power

INSTALLATION TESTS
- POWER CONNECTIONS
- CONTINUITY CHECKS
- SENSOR SIGN VERIFICATION

END-TO-END TESTS
- STICK-TO-SURFACE CALIBRATIONS
- DYNAMIC SURFACE RESPONSES

FUNCTIONAL CHECKS
- MODING OF FLIGHT-CONTROL SYSTEM
- PRIMARY/BYPASS TRANSFER DYNAMICS

RESONANCE TESTS
- INCREASED LOOP GAINS TO IDENTIFY STRUCTURAL RESONANCE PROBLEMS

PREFLIGHT SELF-TEST CALIBRATION
- SELF-TEST TOLERANCES
- VERIFY OPERATION/PASSABILITY

EMI SUSCEPTIBILITY/RADIATION
- EFFECT OF AIRCRAFT SYSTEMS ON DFBW SYSTEM
- EFFECT OF DFBW SYSTEM ON AIRCRAFT SYSTEMS

ENGINE RUN
- FUNCTIONAL TESTS
- DRY RUN OF PREFLIGHT TEST

Figure 5-45. Aircraft integration tests.

lines with the system operating in a normal manner. The second test involved the operation of all electrical systems on the aircraft during normal operation of the DFBW system. There were no observed anomalies during either test.

Formal lightning qualification tests were not performed on the Phase II system. These tests are planned for 1979. Thus, the F-8 DFBW aircraft is currently restricted from flying during electrical storms.

All other test results were positive. With control system loop gains set to four times their maximum value, no adverse structural-mode interactions occurred. The identified problems were corrected and the aircraft was prepared for high-speed taxi tests and first flight. The high-speed taxi tests are considered to be potential flights in the event of an unusual problem which would force a takeoff. Thus, the system has to be declared flight-ready prior to these tests.

5.5.5 Flight-Readiness Determination

The determination that the F-8 DFBW system met the requirements for first flight was based on three factors:

(1) Technical performance of the system.

(2) Flight Readiness Review Board findings.

(3) Flight operations assessment.

The technical performance of the system ultimately forms the basis for all flight-readiness decisions. That is to say, the actual tested, measured, and observed performance of the system over several hundred hours of operation on the iron bird and on the aircraft are weighted much more heavily than the theoretically predicted performance over the next million hours.

Reliability or MTBF figures were used only to assess the probability of loss of the aircraft due to independent hardware faults. This procedure consisted of updating the estimated MTBF figures for various hardware subsystems with values derived from actual field experience. In the case of the F-8 DFBW system, no credit was taken for the bypass system in determining that the primary system was ready for flight, based on hardware MTBF figures. The bypass system was installed primarily for protection against a common-mode fault in the primary system for which no MTBF figure was available. The specific requirements that had to be met by the DFBW system in order to be considered technically ready for flight are:

(1) The FMEA shows that no single fault can result in the loss of the entire primary system, outside of a potential common mode software fault.

(2) The FMET shows:

  (a) No exceptions to the FMEA which reduces fault tolerance.

  (b) No primary-fault sequence which prevents a transfer to the bypass system.

(3) Operation of the proposed flight software on the iron bird and on the aircraft must be without system anomalies which indicate a level of fault tolerance less than the FMEA/FMET results.

(4) No open anomalies exist which would indicate a fault tolerance less than predicted in the FMEA~~,~~ or demonstrated in the FMET.

(5) No function which could be used in flight exists which has not been executed.

(6) No known fault sequence exists for which the pilot cannot determine his proper course of action from onboard information only.

(7) Each test conductor has verified that tests for which he is responsible have been performed and passed, or have noncritical open anomalies.

The independent Flight Readiness Review Board found no cause to restrict or postpone the first flight, based on a review of the test data for each subsystem in the DFBW system. The Flight Operations Directorate, to which the project pilots report, found no data or experience which would prevent a first flight.

5.6 Pilot's Role in Validation and Evaluation

The pilot's role in the validation testing and system evaluation in the F-8 DFBW program was critical to the flight-qualification process because it was the pilot who ultimately had to make the Go/No-Go decision relative to the first flight. Pilot participation in the F-8 DFBW program did not start during the validation phase, but rather during the conceptual design, 2 years earlier. For an experimental system like the F-8 DFBW system, this was an obvious necessity. In the F-8 program the same pilot that participated in the requirements definition phase also was prime pilot in the validation and evaluation phases. This was a very beneficial situation in terms of bringing the system to the flight-qualified stage.

The pilot's participation in the process of validating the system design can be represented as occurring in the following phases.

5.6.1 Breadboard System Evaluation

System breadboards were built and installed in the iron bird prior to delivery of the flight hardware. Pilot evaluations of these systems in closed-loop simulated flight led to several design changes and firmly established the design in several areas, such as displays and controls. Formal validation-like testing was not accomplished during the early evaluations. Rather, the pilot attempted to stress test the overall system to expose design deficiencies. Changes were only moderately expensive if they could be defined during this period of time.

5.6.2 Failure Modes and Effects Testing

As was previously noted, the formal FMET involved over 750 tests. It was not possible, nor is it generally desirable, to have the pilot involved in all of these tests. There was generally an in-line analysis of computer log data required following each fault case. It is unreasonable to expect a pilot to participate in this form of testing. In the F-8 DFBW program, the pilot did evaluate each different type of failure mode, however. Here again, if the system has 1001 different failure mode manifestations, it would be mandatory for a pilot to look at most of them. In the case of the F-8 DFBW system, the number of different manifestations was small, and hence, the pilot could evaluate all of them.

The pilot's responsibility during the FMET phase is to make the following determination for each of the failure modes examined:

(1) Is the fault potentially catastrophic in the worst possible condition at which it may occur?

(2) Is the pilot's required response reasonable and natural, given the demands of the total flying task?

(3) Is the fault annunciation adequate and clear?

(4) Are additional fault-handling systems or features required to assist the pilot in recovering from the fault?

(5) Are there actions which must be avoided in the pilot's recovery process?

(6) Are there special emergency procedures which must be written to assure proper recovery actions are understood?

In addition to these explicit determinations, the pilot must also form an overall impression of system integrity. This can be done only if the same pilot is able to assess the fault-response characteristics of the system over several weeks or months. This assessment, as qualitative as it is, plays an important part in determining flight readiness. These qualitative factors might include:

(1) Consistency of system response to repeated faults of the same type.

(2) Predictability of system response to a fault, given its response to similar faults.

(3) Success in recovery, including type of recovery action and post-recovery flying qualities.

These factors are difficult to quantify, but they establish in the mind of the pilot an impression of the integrity of the system.

5.6.3 Mission-Profile Iron-Bird Testing

During this phase of testing, the system was operated in as normal a fashion as possible. Surprise faults were induced as part of the pilot training process, however. The majority of the work done by the pilot in this phase of testing consisted of flying quality evaluations, and did not differ materially from the type of work usually done on a closed-loop simulator in preparation for an experimental or prototype airplane/system evaluation.

The implicit system evaluation again is most important during these tests. The factors which the pilot must determine are:

(1) Does the system perform consistently during the evaluations, or do mysterious deviations occur for which no explanation can be found?

(2) Are nuisance faults common or rare?

(3) Is familiarization time short or does it take some length of time to proficiently accomplish the planned mission profile?

(4) Are surprise faults able to be handled routinely?

In these types of observations, the pilot's role is no different than that of the operator on any piece of equipment. He will, by sheer experience, formulate an impression of the system that is independent of predicted MTBF or fault-tolerance characteristics. It is this qualitative assessment, which is a complex integration process in the human mind, that ultimately determines that a technically qualified system is in fact flight ready. Where he is not satisfied, the pilot must be able to identify specific problem areas, however, in order that corrective action can be taken.

5.6.4 High-Speed Taxi and Flight Test

The pilot's role in these phases of test and evaluation is, of course, primary. Although the DFBW system comprises the experiment, it is also the primary flight-control system of the F-8C. Therefore, the pilot's first task is to evaluate the characteristics of the airplane. The sole criteria of the acceptability of the primary DFBW system in the high-speed taxi test and first flight is that the airplane is controllable, assuming the system does not fail. Thus, the pilot is primarily responsible for assessing the vehicle/system as an entity in the early flight testing, and in this task, his role as a test pilot is fairly well understood.

The high-speed taxi tests and first flight were conducted according to a detailed flight plan. The test philosophy was to demonstrate normal system performance. This philosophy precluded the deliberate insertion of faults during the flight-test program.

5.7 System Flight-Test Experience

For purposes of this discussion, the engine runs and taxi tests are considered part of the flight-test program. As a matter of policy, DFRC requires the airplane and system to be flight qualified prior to the taxi tests. The flight-test program is the ultimate validation test. Here, the system and airplane demonstrate the mission suitability of the design in the actual flight environment. This section reviews the experience with the DFBW system during the flight-test program, both on the ground and in flight.

5.7.1 Engine Runs

Three full-scale engine runs were accomplished, with the third run including low-speed taxi tests. The systems performed extremely well during these three engine runs, with no anomalies. The low-speed taxi run, preceded by a full ramp preflight test, was accomplished on 11 August 1976. During this test, the various flight-control modes were engaged during low-speed taxi. Postrun computer data analysis showed no alarms or anomalous operation. On the basis of these results, testing proceeded to the high-speed taxi runs.

5.7.2 High-Speed Taxi Tests

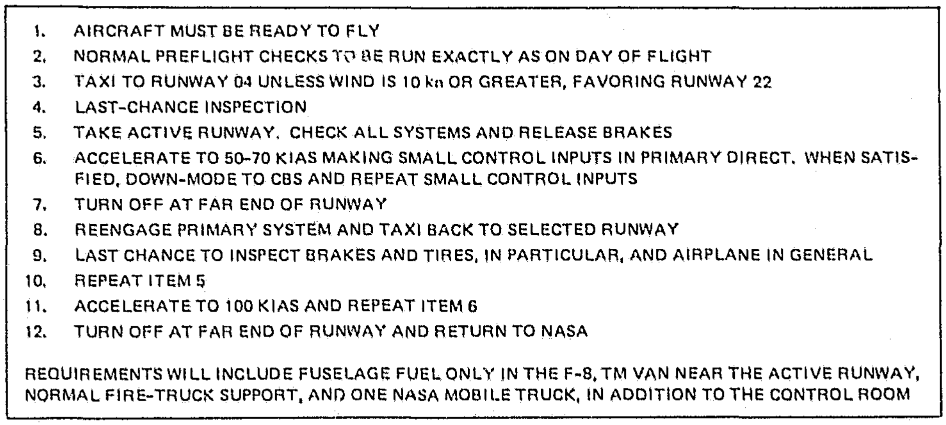
The taxi test plan is reproduced in Figure 5-46. It consisted of two runs, the first to 70 knots indicated airspeed (KIAS), the second, to 100 KIAS. The unstick speed for the airplane weight during these tests was approximately 146 KIAS. The results of the taxi test on 23 August 1976 were as follows:

(1) Directional control was excellent.

(2) Pitch and roll control were not overly sensitive.

(3) No aircraft electrical system EMI effects occurred.

(4) Channel A failed during the shutdown procedure.



1. AIRCRAFT MUST BE READY TO FLY
2. NORMAL PREFLIGHT CHECKS TO BE RUN EXACTLY AS ON DAY OF FLIGHT
3. TAXI TO RUNWAY 04 UNLESS WIND IS 10 kn OR GREATER, FAVORING RUNWAY 22
4. LAST-CHANCE INSPECTION
5. TAKE ACTIVE RUNWAY. CHECK ALL SYSTEMS AND RELEASE BRAKES
6. ACCELERATE TO 50-70 KIAS MAKING SMALL CONTROL INPUTS IN PRIMARY DIRECT. WHEN SATISFIED, DOWN-MODE TO CBS AND REPEAT SMALL CONTROL INPUTS
7. TURN OFF AT FAR END OF RUNWAY
8. REENGAGE PRIMARY SYSTEM AND TAXI BACK TO SELECTED RUNWAY
9. LAST CHANCE TO INSPECT BRAKES AND TIRES, IN PARTICULAR, AND AIRPLANE IN GENERAL
10. REPEAT ITEM 5
11. ACCELERATE TO 100 KIAS AND REPEAT ITEM 6
12. TURN OFF AT FAR END OF RUNWAY AND RETURN TO NASA

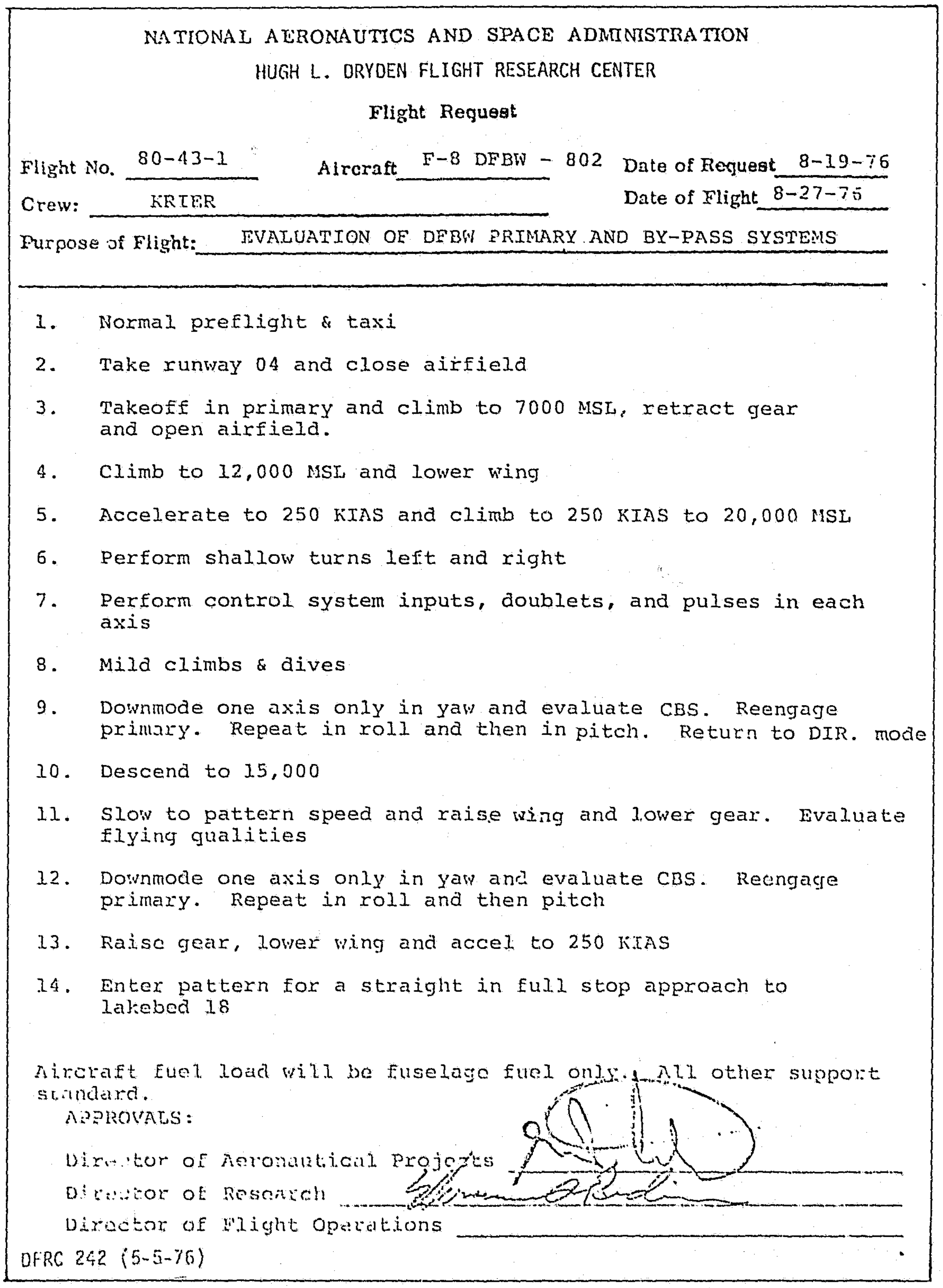
REQUIREMENTS WILL INCLUDE FUSELAGE FUEL ONLY IN THE F-8, TM VAN NEAR THE ACTIVE RUNWAY, NORMAL FIRE-TRUCK SUPPORT, AND ONE NASA MOBILE TRUCK, IN ADDITION TO THE CONTROL ROOM

Figure 5-46. F-8 DFBW taxi test plan.

The channel failure was found to be due to inadvertent crew operation of one of the built-in fault-injection circuits on the ground-test cart. Post-test computer dumps showed the system operation to be nominal, including the response to the injected fault. All aircraft electrical and avionics systems were cycled during the taxi back to DFRC facility. No EMI effects on the DFBW system were noted. On the basis of these results, testing proceeded to the flight phase with no system changes required.

5.7.3 First Flight

The first flight was accomplished successfully on 27 August 1976. The general flight plan is reproduced in Figure 5-47. One significant



NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

HUGH L. DRYDEN FLIGHT RESEARCH CENTER

Flight Request

Flight No. 80-43-1	Aircraft F-8 DFBW - 802		Date of Request 8-19-76
Crew: KRIER			Date of Flight 8-27-76

Purpose of Flight: EVALUATION OF DFBW PRIMARY AND BY-PASS SYSTEMS

1. Normal preflight & taxi
2. Take runway 04 and close airfield
3. Takeoff in primary and climb to 7000 MSL, retract gear and open airfield.
4. Climb to 12,000 MSL and lower wing
5. Accelerate to 250 KIAS and climb to 250 KIAS to 20,000 MSL
6. Perform shallow turns left and right
7. Perform control system inputs, doublets, and pulses in each axis
8. Mild climbs & dives
9. Downmode one axis only in yaw and evaluate CBS. Reengage primary. Repeat in roll and then in pitch. Return to DIR. mode
10. Descend to 15,000
11. Slow to pattern speed and raise wing and lower gear. Evaluate flying qualities
12. Downmode one axis only in yaw and evaluate CBS. Reengage primary. Repeat in roll and then pitch
13. Raise gear, lower wing and accel to 250 KIAS
14. Enter pattern for a straight in full stop approach to lakebed 18

Aircraft fuel load will be fuselage fuel only. All other support standard.

APPROVALS:

Director of Aeronautical Projects [illegible]

Director of Research [illegible]

Director of Flight Operations

DFRC 242 (5-5-76)

Figure 5-47. NASA DFRC flight plan.

point is that the primary digital system was used for takeoff and landing. During the 40-minute flight, the primary DIRECT and bypass system modes were evaluated in low-speed flight. In the crawl-before-walk philosophy used, the augmented modes were not engaged until the fourth flight. The results of this flight can be summarized as follows:

(1) Handling qualities in the primary and bypass modes were satisfactory.

(2) Moding transients were small.

(3) The triplex wing-up discrete set was declared failed by the RM software due to mechanical flexing of the wing during the takeoff.

(4) The heading gyro set was declared failed.

Item number (3) was a surprise. Even with the liberal 300-millisecond tolerance on discrete failure declarations, the switch contact skew was sufficient to cause the set to be failed. A mechanical adjustment was made to alleviate this problem.

A bad heading gyro caused a valid miscompare leading to item (4). It was replaced.

5.7.4 In-Flight System Performance Summary

The performance experience of the DFBW system covered 50 flights and approximately 55 flight hours. It is of special interest to relate this experience to the ground-test program, in order to assess the reasonableness and thoroughness of the verification and validation approach taken on the F-8 system. Figure 5-48 summarizes the in-flight system experience for the first 50 flights. The following sections will discuss these items in more detail.

(1) ALL TAKEOFFS AND LANDINGS HAVE BEEN IN THE PRIMARY DIGITAL SYSTEM.
(2) REVERSION TO THE ANALOG BYPASS SYSTEM HAS NOT BEEN REQUIRED.
(3) SINGLE-CHANNEL HARD FAILURES HAVE OCCURRED ON THREE FLIGHTS.
(4) CHANNEL FAILURES WERE ALL DUE TO HARDWARE FAULTS.
(5) NO HARD OVER OR OTHERWISE HAZARDOUS SURFACE COMMANDS HAVE BEEN GENERATED.
(6) ONE TRANSIENT CHANNEL FAULT HAS OCCURRED.
(7) SOFTWARE-DESIGN ERRORS HAVE BEEN DISCOVERED IN POSTFLIGHT ANALYSES.
(8) SEVERAL CONTROL LAW REFINEMENTS WERE NECESSARY.
(9) NO FLIGHT WAS ABORTED DUE TO A SOFTWARE FAULT.
(10) NO SOFTWARE FAULT AFFECTED FLIGHT PERFORMANCE.
(11) FEW SENSOR NUISANCE FAULTS OR ACTUAL FAULTS WERE NOTED.

Figure 5-48. In-flight system performance summary.

5.7.5 In-Flight Computer/IFU Fault Experience

Table 5-14 describes the in-flight computer and interface-unit failure experience. The computer faults on Flights 2 and 3 provided fail-operational experience earlier than expected. The system FDIR response was nearly identical to that observed during ground testing. In both cases, the two remaining channels declared the offending channel hard-failed, and operation continued with no degradation in flying qualities. Precautionary but routine landings were made on the remaining two channels. Postflight analysis of the fault on Flight 3 did disclose a software design error, which is discussed under software experience.

Table 5-14. In-flight computer and IFU failure experience.

Flight	Failure	Comments
2	Computer memory fail	● Memory parity error unrecoverable ● Multiple restarts could not restore operation
3	Computer memory fail	● Channel declared failed by partners ● No surface transients ● Pilot did not observe performance change ● Routine landing on two channels
17	Interface unit I/O (transient fault, failed hard on ground)	● Restart restored normal operation ● No surface transient ● Flight continued per plan
19	Computer stopped execution	● Partners assumed it was off ● No restarts ● Routine landing on two channels
22	IFU Component	● Channel failed by partners/ self ● Routine landing on two channels

The computer outputs and airplane response during the computer failure on the second flight are shown in Figure 5-49. Multiple restarts occurred between the time the memory parity error occurred, and the point at which a hard fault was declared. During this time, no surface deviations occurred. The vehicle responded normally on two channels after Channel A had been declared hard-failed.

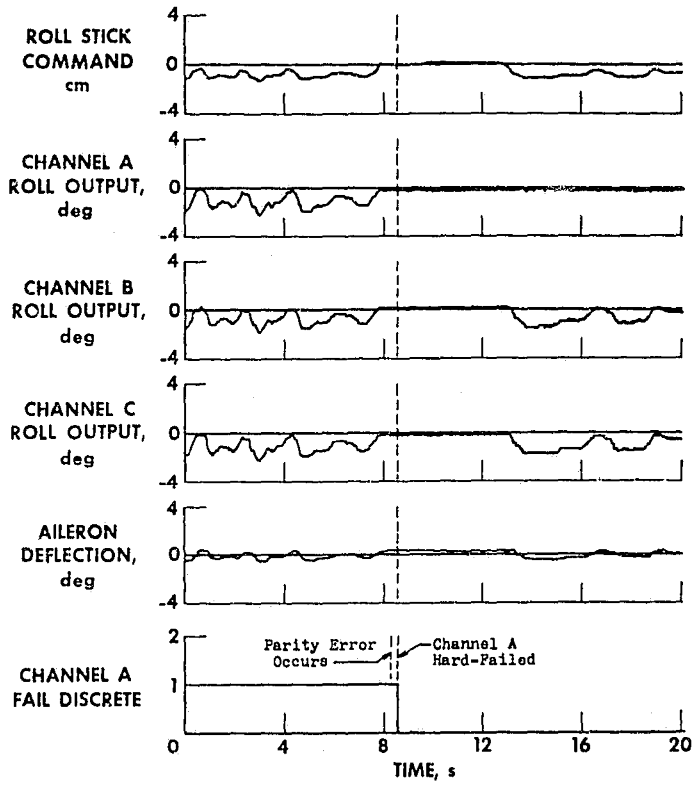


Figure 5-49. In-flight computer failure.

5.7.6 Software Anomaly Experience

Software anomalies have occurred during the period of system operation since the first flight. This represents approximately 1750 hours of operating time. Table 5-15 categorizes the software anomalies encountered during this period of time. As the figure shows, no software errors have been found that would invalidate the fail-operational requirements.

C-3

Table 5-15. Software anomaly experience.

Software Subsystem	Types of Anomalies	How Found	Comments
System redundancy management	Fault-detection logic	Analysis of flight-computer fault	● Design error ● Noncritical
	Deficiences in fault recovery logic	Analysis of ground-test data	● Could prevent recovery from multiple transient faults
	Sensor fault logic errors	Ground operation	● Minor system effects
Control laws	Restart initial-ization	Ground operation	● Minor system effects
	Miscellaneous flag handling	Ground operation	● Minor system effects
Input/ output	Incorrect inter-nal interrupt handling	Ground-test data analysis	● Resulted in occasional restarts on ground and one in flight ● Noncritical
Ground-test programs (loader, displays, preflight test)	Nuisance-type faults	Ground operation	● Noncritical ● Minor effects

The transient I/O fault on Flight 17, due to a failing component in the IFU, was serious enough to cause a restart which restored operation. The only indication to the pilot was a resettable light.

The execution suspension on Flight 19 appeared to the other two channels as a power down. No restarts occurred, and an uneventful landing was made on the remaining two channels. The faulty computer ran when powered up later and achieved normal sync with its partners. The IFU fault on Flight 22 was handled routinely by the system FDIR routines and looked identical to the pilot as the previous channel failures. These in-flight faults increased the confidence in the fail-operational capability of the system.

Table 5-16 summarizes the in-flight internal performance of the computer IFU redundancy management software. Other than the restarts which occurred concurrently with permanent hardware faults, only one other restart has occurred in flight, that due to the I/O problem on Flight 17.

Table 5-16. In-flight computer redundancy management software performance.

Fault	Quantity	Comment
Restarts*	1	Flight 17 transient I/O failure
I/O errors*	8	Estimated 10^{10} I/O operations
False fault declarations	None	
Alarms indicating serious abnormality*	None	

* Excluding those accompanying hard failures.

Eight detected I/O errors have occurred during the estimated 10^{10} I/O operations which took place in flight. There have been no channel faults declared in flight where a real fault did not occur. Finally, there have been no alarms logged which would indicate any serious abnormality in the executive, self-test, or crosslink communication operations. These results indicate an extremely high level of false-alarm immunity in the FDIR software.

Performance of the flight software during the 20 months and 1750 hours of operation is also excellent. Figure 5-50 summarizes ground experience during this period. No software fault has been found during this 20-month period which would have invalidated the system flight qualification had it been known prior to first flight.

One final software fault experience is germane to this subject. An error in the manufacturer-supplied support software for the computer resulted in an erroneous load tape for a new software release. This error was not detectable by ordinary means and was only discovered

- ALL KNOWN FAULTS DETECTED BY SYSTEM
- NO HARDWARE FAULT CAUSED MORE THAN ITS OWN CHANNEL TO FAIL
- NO VERIFIED NUISANCE-CHANNEL FAULT
  - APPROXIMATELY SIX OPEN ANOMALIES TENTATIVELY ATTRIBUTED TO HARDWARE AND PROCEDURAL CAUSES
- NO COMMON MODE SOFTWARE FAULT HAS OCCURRED WHICH WOULD DISABLE PRIMARY SYSTEM
- NO AUTOMATIC REAL OR NUISANCE TRANSFERS TO THE BYPASS SYSTEM

Figure 5-50. Ground experience with flight software (1750 hours).

in ground testing because it was in executing code. This error was in an area of the code which had not been modified in the new release, and which ordinarily would not have been reverified. This emphasizes the point that software faults can be introduced in a very subtle manner when code is modified after qualification. This potential error source was eliminated by a specially developed procedure applied to the tape manufacture step.

5.7.7 Software Modification Experience

In the course of the F-8 program, 10 software releases have been flown which implemented both minor software fixes as well as major new features. The reverification record has been perfect. No known error has survived the reverification process. This is an implicit indication of the degree to which modularity actually exists in the software. In addition, several overlay tapes have been made to change gain constants in the control-law-refinement task. No erroneous number has survived to the flight tape. This is due to virtually foolproof automatic bookkeeping and verification methods which are available.

The rules governing software modification on the F-8 program are:

(1) All changes must be positively verified and validated on the iron bird.

(2) All changes must be evaluated by a pilot in real-time simulation on the iron bird.

(3) A change to the system redundancy management software requires all FMET steps applicable to that subsystem be repeated.

Software changes involve more risk than the generation of the original software because they are exposed to fewer hours of operation prior to use. Thus, new software does not receive the implicit verification that often exposes errors.

Some additional observations can be made based on the minor and major modifications made to the F-8 DFBW software during the course of the program.

(1) For some straightforward types of changes, the fact that it was a software change meant that implementation could be accomplished in significantly less time than if it had been made in hardware. This includes verification and validation time as well.

(2) For changes affecting more than a small isolated area of software, or affecting more complex functions, implementation effort seemed to be comparable to hardware changes of the same magnitude. One advantage of the software implementation was that the hardware did not need to be requalified, and the existing system could be used until the new software was ready.

(3) The careful reverification and revalidation of new software to the original standards on an iron-bird facility, appears to be an acceptable method of qualifying that software.

(4) Changes in gain constants, filters, and gain schedules could be made routinely in one day with a 100-percent verification process

(5) Where flow charts were not available, changes seemed to incur more interference problems, and reverification was more likely to turn up problems.

5.7.8 Software Reliability

A measure of the reliability of software may be deduced from the "failure" history of the program. Software failures are those deviations in the operation of the digital program which result in violating the performance specification. Recent work in attempting to quantify software reliability has involved the analysis of software errors found during the verification, validation, and operational use of the digital program. During the F-8 DFBW flight-control system software development, records were kept for all anomalies which were found in officially released software. These records did not include errors found in the debug process preceding the production of an executable program.

Anomalous software operation was arbitrarily classified as serious or minor. In the first category were all anomalies which could lead to a false fault declaration, reduced controllability of the aircraft, or could prevent the detection of a valid fault. Figure 5-51 shows the F-8 DFBW flight software-anomaly experience since release of the multicomputer software program. Also displayed in this figure is the total system operating time over the same period of time. Edited from the total anomaly figures are those anomalies associated with the implementation of new functions and those resulting from procedural errors.

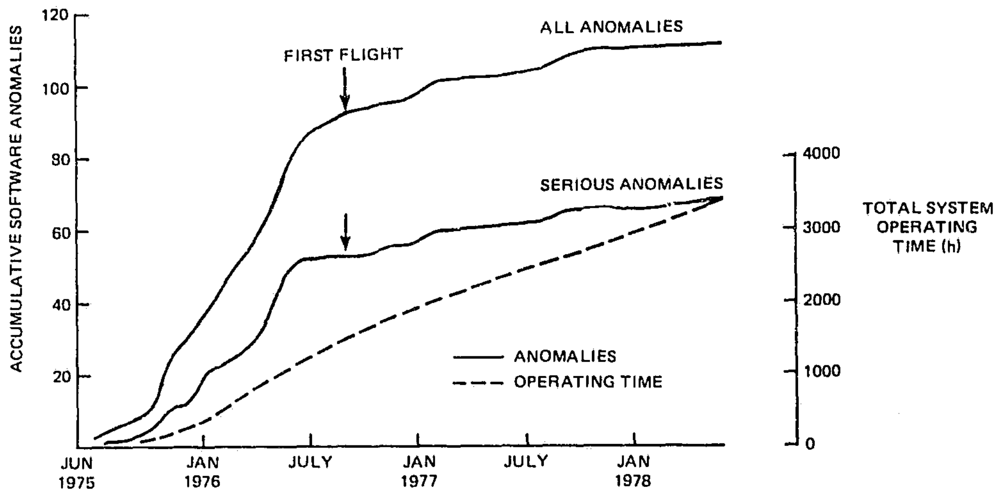


Figure 5-51. Software-anomaly experience.

Both serious and total anomaly histories follow the same trend over what was a constant operating rate. Obviously, most of the anomalies are exposed in the detailed verification and validation tests. It should be noted, however, that the nature of the research program resulted in considerable reverification work as various new functions were added to the program. Thus, the system software has been under almost constant testing over the period of time shown in the figure. Some observations can be made from the anomaly data and from an examination of the particular anomalies found.

(1) A period of approximately 290 hours of operation without a software anomaly preceded the first flight.

(2) Serious software deficiencies were found after the first flight as a result of intense analysis of actual in-flight and ground hardware failures.

(3) The "serious" software deficiencies found after the first flight were "potentially" serious in that the conditions under which the problem would manifest itself did not actually occur.

(4) The "failure" rate of the qualified flight program is not zero.

(5) The software anomalies are not randomly distributed over all modules.

(6) No common mode catastrophic faults have been found in the flight-qualified software.

It is apparent that serious faults may be exposed in a flight critical control system which are not necessarily catastrophic. This test philosophy, however, is dependent on having an alternate control mode (bypass control system). The alternate control mode could reasonably be comprised of resident independent software in the primary digital system. The "failure" rate of the flight-qualified software does yield a qualitative degree of confidence in the integrity of the software. Quantitative measures may be derived provided very detailed records are kept of software faults in a format amenable to analysis. A standard recording format may be useful in this regard. The fact that the software anomalies were not randomly distributed over all software modules or functions clearly indicates that it is not possible to generalize from the data of Figure 5-51 and apply the failure rate trends to a different software program. The data does reveal much about the specific program being developed, however.

5.7.9 Sensor Redundancy Management Experience

The F-8 DFBW experience with sensor redundancy management has been very good. Flight-test data were used to refine the fault thresh-holds for the dynamic motion sensors. Table 5-17 illustrates the margin for sensor fault-declaration false alarms. Cross plots of each sensor pair of each triad and duplex set were examined to determine the maximum differences in moderate turbulence. These differences are all well below the actual fault-tolerance levels.

Table 5-17. Flight II sensor tracking in moderate turbulence.

Sensor Set	Tolerance	Max. Deviation
Pitch rate (deg/s)	4	0.68
Roll rate (deg/s)	10	1.036
Yaw rate (deg/s)	4	0.22
Axial accelerometer (g)	0.1	0.066
Lateral accelerometer (g)	0.1	0.061
Normal accelerometer (g)	0.5	0.060
Pitch C stick (cm)	1.0	0.086
Roll C stick (cm)	1.0	0.092
Rudder pedals (cm)	0.5	0.03
Angle of attack (deg)	2.0	1.148
Left secondary actuator (cm)	3.5	0.167
Right secondary actuator (cm)	3.5	0.209
Pitch attitude (deg)	15	1.85
Roll attitude (deg)	15	2.81
Heading (deg)	15	3.52
Mach number	0.05	0.036
Altitude (m)	150	6.7

Table 5-18 lists the in-flight sensor failure experience to date. As can be seen, very few problems have been encountered. Of the three hard failures which have occurred, two have been actual faults. Ground experience has been about the same. It is also significant to note that the sensor RM algorithms in the F-8 software are very simple. No known sensor fault has gone undetected.

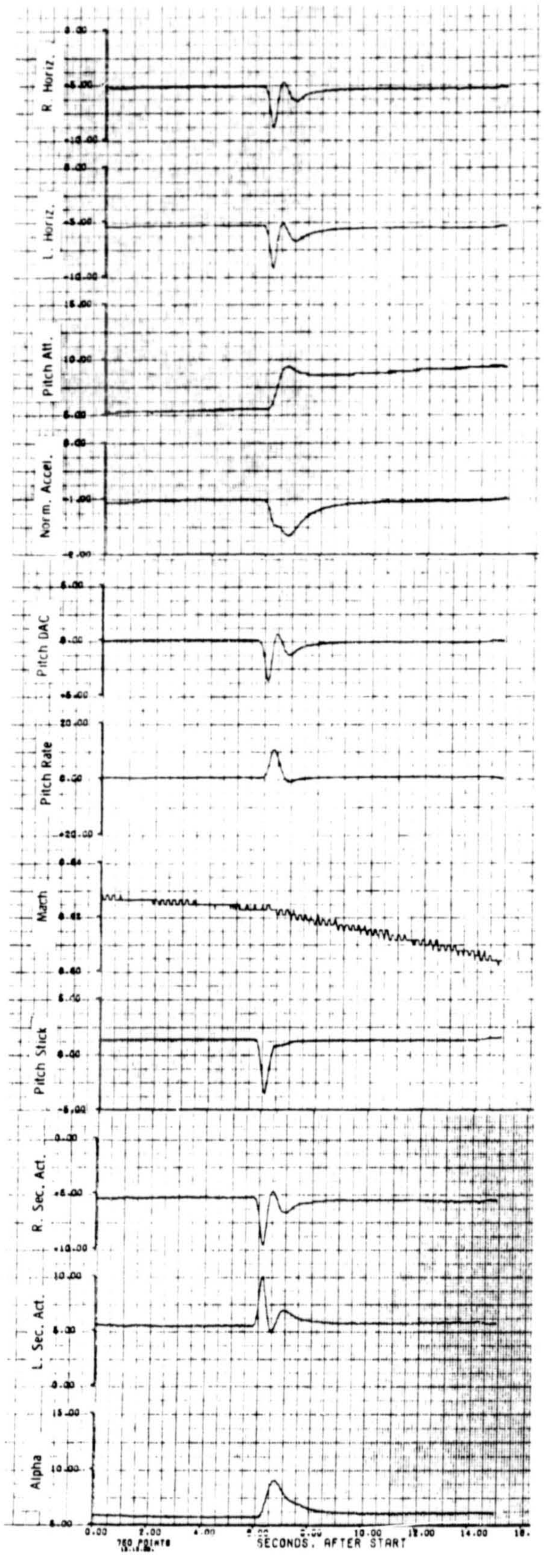


Figure 5-52. Flight-test control-system pitch SAS response (Flight 5).

Table 5-18. F-8 DFBW in-flight sensor failure experience.

Sensor	Number of Hard Failures	Cause
Lateral Acceleration	1	False Alarm
Heading Gyro	2	1 - Bad Gyro 1 - False Alarm
Wing Discrete	1 (entire set)	Switch contact skew and wing mount flexing

5.7.10 Control-System Performance

The performance of the control system has been very good. All control modes have been flight tested. The general observations of the control system are:

(1) Closed-loop airplane responses closely match the analytic predictions as well as the responses obtained on the iron-bird simulator.

(2) The control-surface responses and the airplane responses are indistinguishable from those of a single-channel control system.

(3) No adverse effects of sample rate or quantization have been observed.

Figure 5-52 illustrates the type of airplane and control-system responses obtained during the flight-test program. The control-surface motions are smooth and regular, and do not differ from what would be expected in a single-channel system. The airplane response in this, the pitch SAS mode, is well damped and regular.

The active flap modes have been evaluated for handling qualities effects, but all performance testing is not yet complete. Figure 5-53 illustrates the operation of the maneuver-flap mode. With the flap mode active, nearly all the increased load factor in this mild turn is produced by the flap deflection, with almost no increase in angle-of attack. In maneuvering flight, this translates into lower lift-induced drag, and hence higher performance. The ride-smoothing system

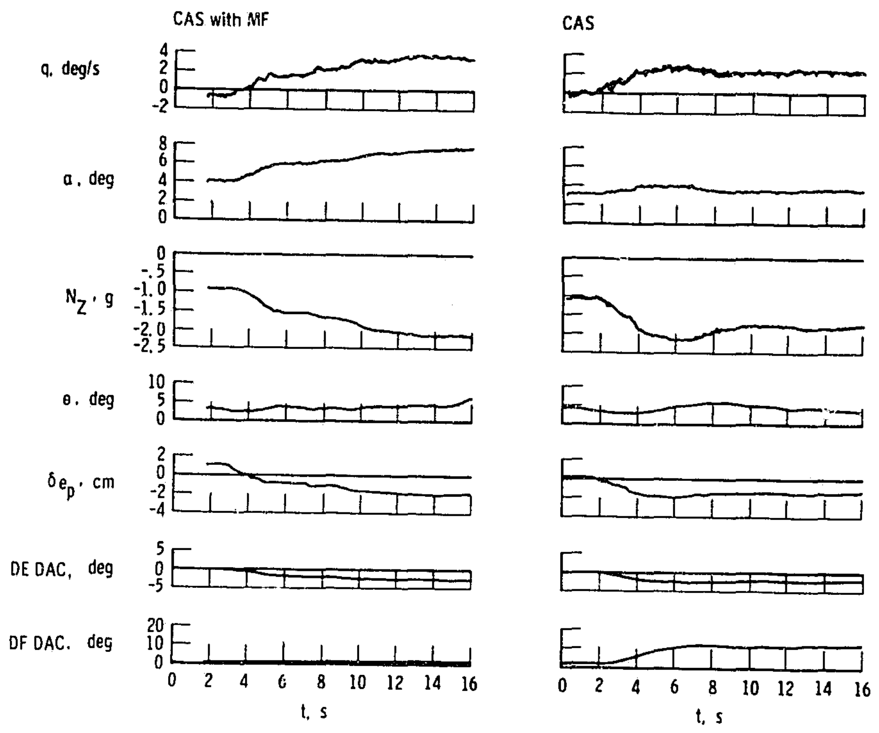


Figure 5-53. Maneuver flap (MF) performance.

has been evaluated at reduced gains in moderate turbulence and at design gains in smooth air to verify closed-loop operation. Flights in moderate turbulence with the full design gains are planned for the near future.

The angle-of-attack limiter has been evaluated and has been found to produce a hard limit. It is not possible for the pilot to increase the angle of attack above the reference value as long as control power is available. Figure 5-54 shows the operation of the limiter during one of the flight-test maneuvers. Here, the reference angle of attack was set at 9 degrees for test purposes. The pilot moves the stick full aft while the control system maintains the vehicle at 9 degrees for test purposes. The pilot moves the stick full aft while the control system maintains the vehicle at 9-degree angle of attack. There are no discontinuities as the system limits the aircraft motion and state.

Autopilot performance has been good. Attitude hold is reasonably tight, and the outer-loop functions operate within fairly narrow boundaries. Figure 5-55 shows the operation of the altitude-hold mode

ORIGINAL PAGE IS OF POOR QUALITY

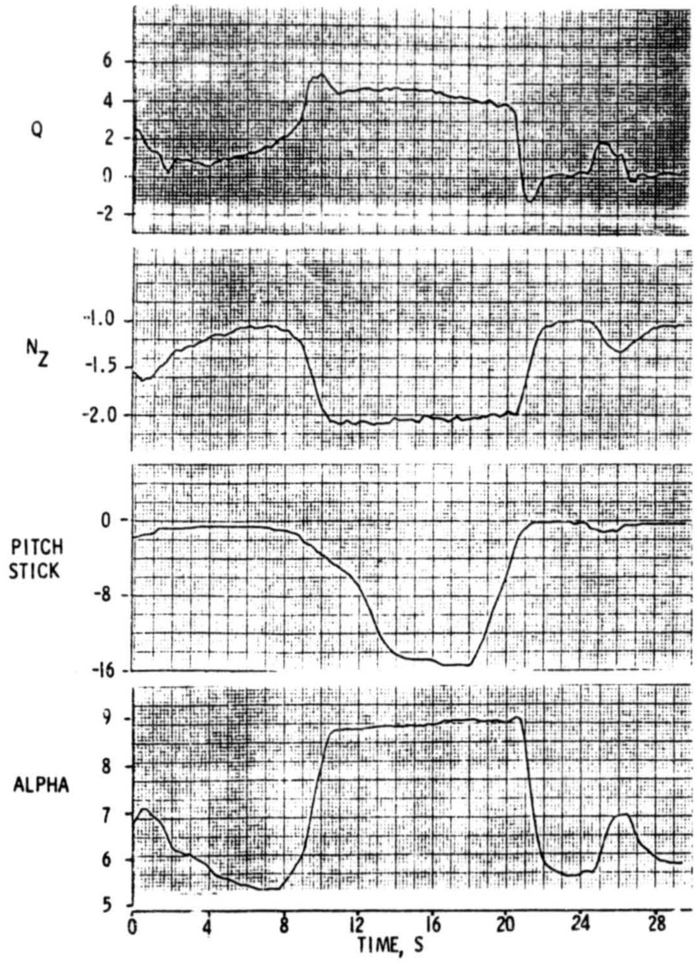


Figure 5-54. Alpha limiter test (Flight 20).

in ascents and descents, with the hold function engaged from a steady climb or dive. In general, the aircraft returns to the reference altitude without oscillations, and achieves a steady-state error less than 20 meters.

Closed-loop damping in the lateral-directional axes is good, with the response being nonoscillatory almost everywhere in the flight envelope.

Simultaneous operation of active control modes was demonstrated with a pull-up in the control-stick steering mode deliberately made into the alpha limiter mode with both the ride-smoothing and maneuver-flap modes active. The maneuver was uneventful.

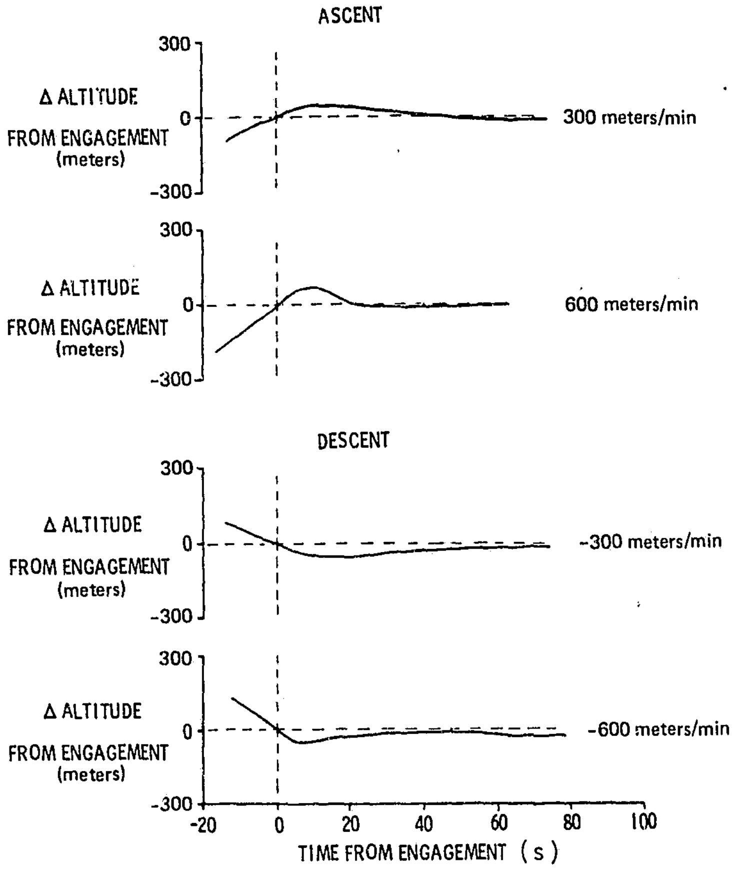


Figure 5-55. Autopilot performance—altitude hold.

5.7.11 Flying Qualities

The handling qualities of the F-8 DFBW have been found to be generally good to very good. Some problems have been encountered in the roll axis, however, with oversensitivity being apparent in close-formation flight. This was alleviated by shallowing the parabolic shaping sufficiently to increase the force-per-roll-rate/roll-acceleration ratio. General pilot observations concerning the airplane flying qualities are:

(1) The airplane is very good in the takeoff and the approach and landing configuration.

(2) Control is very smooth for large maneuvers.

(3) Gunsight tracking is comparable to in-service aircraft of various types.

(4) Formation flight is not as good as in the best airplanes, but is very satisfactory.

(5) Pitch control improves with the active flap modes operating.

(6) The aileron-to-rudder interconnect needs some tuning at specific points in the flight envelope.

(7) There is nothing in the airplane response to indicate that the control system is multichannel or digital.

Cooper-Harper pilot ratings are generally all better than 4. Some flying-quality tuning is still planned in order to improve some minor deficiencies.

5.7.12 Control-System Problems

Some problems have been encountered in the flight program. Although these problems did not occur because the system was digital, or fly-by-wire, it is of interest to note their occurrence.

(1) The wing flexure of the aircraft was sufficient to cause the triplex position switches to open, making it appear to the software that the wing was up. The software selected wing-up loop gains, resulting in a small amplitude limit cycle in the roll axis.

(2) Investigation of wing bending and shear moments with the maneuver-flap mode active showed that although the lateral center of pressure shifted inboard with the flap deflecting downward, the wing bending and shear loads increased about 15 percent. This was due to the adverse pitching moment of the flap, and the resultant increase in wing lift required to restore lift lost by the downward-loaded horizontal tail surface.

(3) The rudder response due to the aileron-to-rudder interconnect term was too abrupt, and could be detected by the pilot as a directional lurch. This required the activation of a low-pass filter in this path, which was not indicated in the design.

(4) The pitch CAS mode was found to be poor for wing-down landing approaches in the 200 KIAS regime. The gain was determined to be too low at the low end of the speed range for precision maneuvering. A modified gain schedule improved the CAS mode to be equal or better than the SAS mode at the same conditions.

(5) It was discovered that gear strut compression could trigger the rate check module in the pitch-augmented modes due to high-frequency low-amplitude oscillations on the resultant pitch-rate signal. The rate check threshhold was increased to alleviate this possibility.

5.7.13 Miscellaneous Hardware Experience

No in-flight faults have occurred in the computer bypass system or in any of the secondary actuators. One common mode fault did occur in the digital mode and gain panel during a preflight test. One of the DIRECT push buttons mechanically jammed in the depressed position, making it appear that the pilot was requesting that mode. Selection of other modes would result in a mode transfer for as long as the pilot held the second mode button in. After releasing the button, the mode reverted to DIRECT. This problem was traced to a manufacturing defect.

5.7.14 Preflight Test Program Experience

Because of the experimental nature of this system, a hangar preflight test is accomplished prior to each flight. The tests performed are essentially those performed on the ramp by the pilot prior to flight. The flight-line preflight takes approximately 40 minutes, including all aircraft checks. In all fairness, however, it must be noted that no attempt was made to minimize this portion of testing as if it were an operational aircraft. The digital systems tests consume approximately 5 minutes of flight-line test time.

There have been no problems associated with the computer load process. In the F-8 program, the memory is loaded prior to each flight, not because of memory alteration problems, but because of crew interrogation and testing after flight.

The philosophy on this experimental program was to overtest rather than undertest. Experience has shown that the system is overtested prior to flight. The only flight-line abort due to a preflight test was due to a fault in the analog self-test circuitry itself. No part of the preflight test checks software functions. Only sensor and controller interfaces are checked. The computer load process is a 100-percent verified process to assure that the software has been loaded correctly. From that point on, the assumption is made that the software will work as validated. Table 5-19 lists the preflight test tolerances on several of the parameters examined during the automatic test sequence.

Table 5-19. Preflight test program tolerances.

Parameter	Tolerance
Alpha	±2 deg
Mach	±0.05
Altitude	±150 meters
Rate-Gyro Torque	
Pitch/yaw	0.575 ± 0.225 Vdc
Roll	0.175 ± 0.075 Vdc
Accelerometer Torque	
N_x	-1.8 ± 0.5 Vdc
	2.4 ± 0.5 Vdc
N_y	-2.1 ± 0.5 Vdc
	2.1 ± 0.5 Vdc
N_z	-1.7 ± 0.5 Vdc
	2.7 ± 0.5 Vdc
Pitch attitude	7.5 ± 5.0 deg
Roll attitude	0 ± 5.0 deg
DAC outputs and surface position	
Pitch	±2 deg
Roll	±3 deg
Yaw	±2 deg
End-to-end gearing checks	
Pitch	1.7 ± 1 deg/cm
Roll	7.07 ± 1.5 deg/cm
Yaw	2.62 ± 0.75 deg/cm

5.7.15 Flight-Test Summary

Figure 5-56 shows the flight-test history of the F-8 DFBW aircraft in summary fashion. Following the computer fault on the second flight, several computers went through a rework program. There were no flights during this period due to the computer MTBF experience. Since flight testing was resumed in January 1977, the flight program has progressed in a relatively nominal fashion. The shuttle software evaluation flights and the transport delay evaluation flight test series were both in direct support of the shuttle program.

There have been no in-flight faults in the last 28 flights.

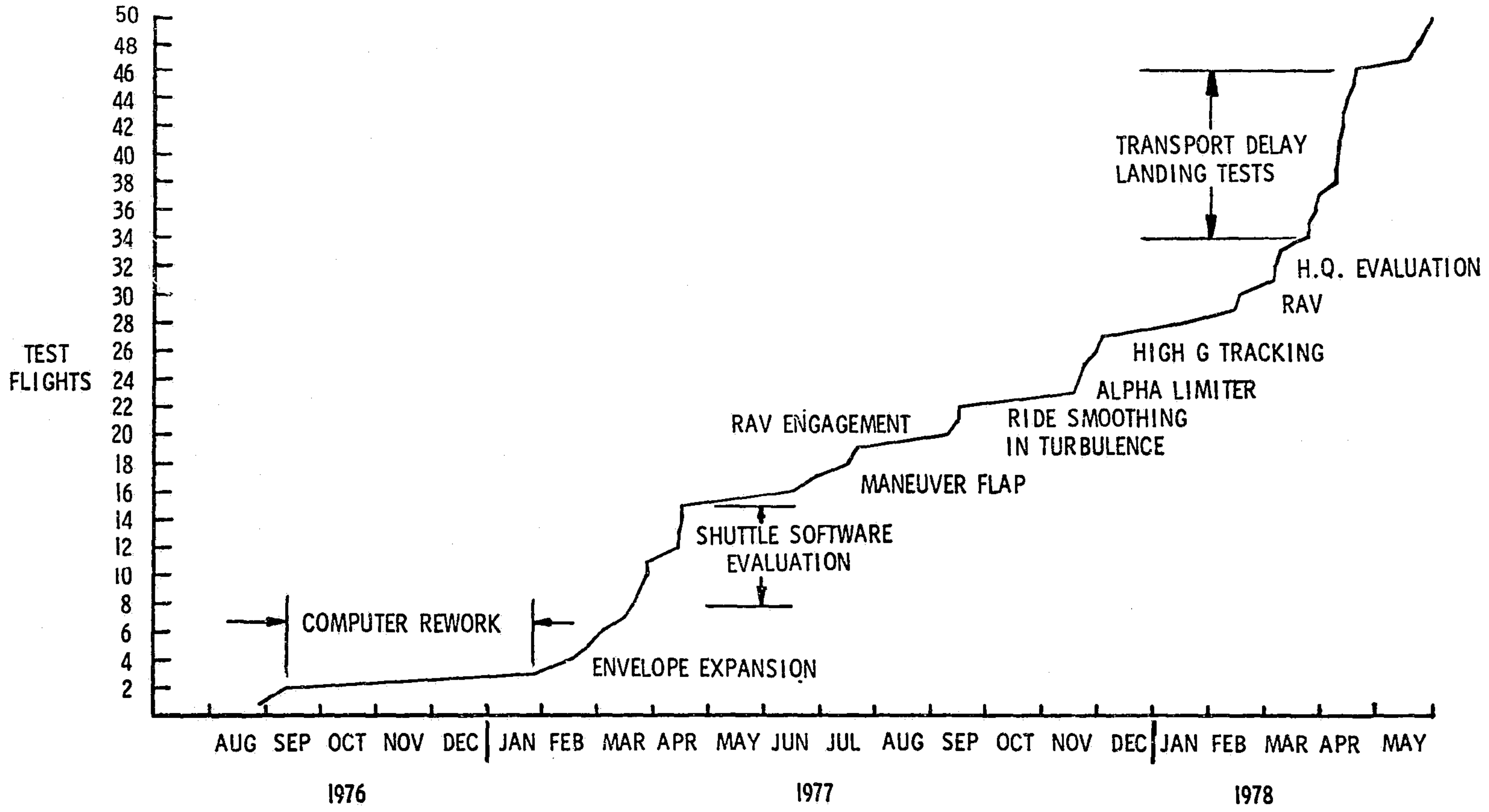


Figure 5-56. Flight-test summary.

SECTION 6

COMMENTS ON VALIDATION METHODS

The purpose of this section is to offer comments on validation methods based on experience on the F-8 DFBW program and on knowledge of other programs and methods that are in use or being developed. It is hoped that these comments will contribute to the body of experience that must be obtained before the policies for the validation and certification of advanced-capability electronic flight-control equipment can be developed. No single organization is in the position of identifying a complete validation program that would be appropriate for all electronic flight-control equipment intended for production civil use. It is not even possible to identify all of the critical questions or all of the areas where further development is necessary. Acceptable validation procedures will have to be developed jointly by all elements of the industry and by the appropriate government agencies. This development process will take several years and will be perfected only in an evolutionary manner by the discipline of validating actual production systems as they begin to be introduced. The experience gained with the first truly fly-by-wire system can, however make valuable contributions to this development process.

In this section, a review is made of some of the validation methods used on the F-8 DFBW, where they were most effective, and where improvements would be recommended. Also described are other available techniques that were not used. A special discussion of software is included which is a relatively new area, and an area of special concern. More questions may be raised than answered, but it is hoped that by raising those questions the process of obtaining a solution may be accelerated.

6.1 Review of Validation Methods

Many of the validation methods used during the F-8 DFBW program are not basically different from methods already in common use. Most of these methods will continue to be useful in the validation of future systems. There are, however, at least three characteristics of emerging advanced flight-control systems which will require some of these methods to be modified and expanded, and new validation concepts to be developed. The three characteristics identified here are the increased level of system functional reliability required, the increased degree of system complexity, and the more extensive use of software programs. The special characteristics of software will be discussed in Section 6.2.

The increased levels of reliability required are moving systems further and further away from levels that can be analyzed and demonstrated with conventional methods. Individual electronic units with failure rates in the range of 10^{-3} to 10^{-4} per hour can be analyzed by conventional methods such as FMEAs, using established experience for component failure rates. The predictions made by these methods can be rather accurate and can be confirmed by actual service experience. A typical production unit will accumulate hundreds of thousands of operating hours per year and will experience many failures, giving a statistically significant estimate of its actual reliability. However, to achieve the very high reliability required of a flight-critical system, several units have to be integrated into a system which can tolerate faults and still operate. The reliability of this total system has to be assessed theoretically, based on an analysis of the reaction of the system to failures within the individual units. The reliability of individual units is an important input into this system analysis. Thus, conventional unit-reliability analysis will continue to be important. However, the methods used to combine this data to give the total system-failure rate are still being developed. We believe the contribution to the system failure rate due to combinations of random failures of individual components within units can be determined with theoretical (albeit complex) analysis techniques. We do not know of any method for demonstrating the absence of potential common-mode failures or generic design faults to this low level of probability. If a method were proposed, it is obvious that it could not be demonstrated by service experience, since the goal of the design is to produce a low probability of one failure in whole fleets of aircraft over their entire life spans.

Another characteristic of emerging systems is their greatly increased complexity. This complexity is due both to the increased capabilities of the systems and to the necessity for redundancy and the associated redundancy management necessary to meet the reliability requirements. Conventional reliability-assessment methods can usually still be applied to these systems. However, the great increase in the number of different combinations of paths and conditions make the usual detailed fault analysis very complex and difficult to perform. They are also difficult to understand; thus, their usefulness and cost effectiveness is reduced. New validation methods can make a valuable contribution to increasing the confidence that the system actually is able to meet its reliability requirements.

Several reliability-assessment tools have been developed to aid in the analysis of complex systems. These analysis techniques use a variety of analysis methods and simulation procedures, and are normally implemented in large general-purpose computer programs. These tools were not used formally in the F-8 DFBW program. Two of these methods, however, will be described here briefly to give an idea of their purpose and characteristics in order to show what contribution these methods might make in future validation efforts.

The first technique is the Complementary Analytic-Simulation Technique (CAST).(21) CAST allows the best features of both analysis and simulation to be used in analyzing system reliability. Analytic modeling can be very flexible and rapid. However, for the more complex systems, the mathematical model can become very involved and almost unmanageable. Simulation can more easily handle system details but is slow and expensive. These methods are effectively combined in CAST by using an engineering characterization of the computer system to provide input to a fault-driven simulation, which minimizes simulation costs. The simulation produces modeling parameters that are used in the analytic modeling to measure the fault tolerance of the system. This process is shown in Figure 6-1. Results of applying CAST to typical system configurations is shown in Figure 6-2.

Another analysis program is the Computed Aided Redundant System Reliability Analysis (CARSRA).(22) CARSRA is a general-purpose reliability-analysis program that handles modular-redundant reconfigurable systems, taking into account such factors as fault coverage and transient

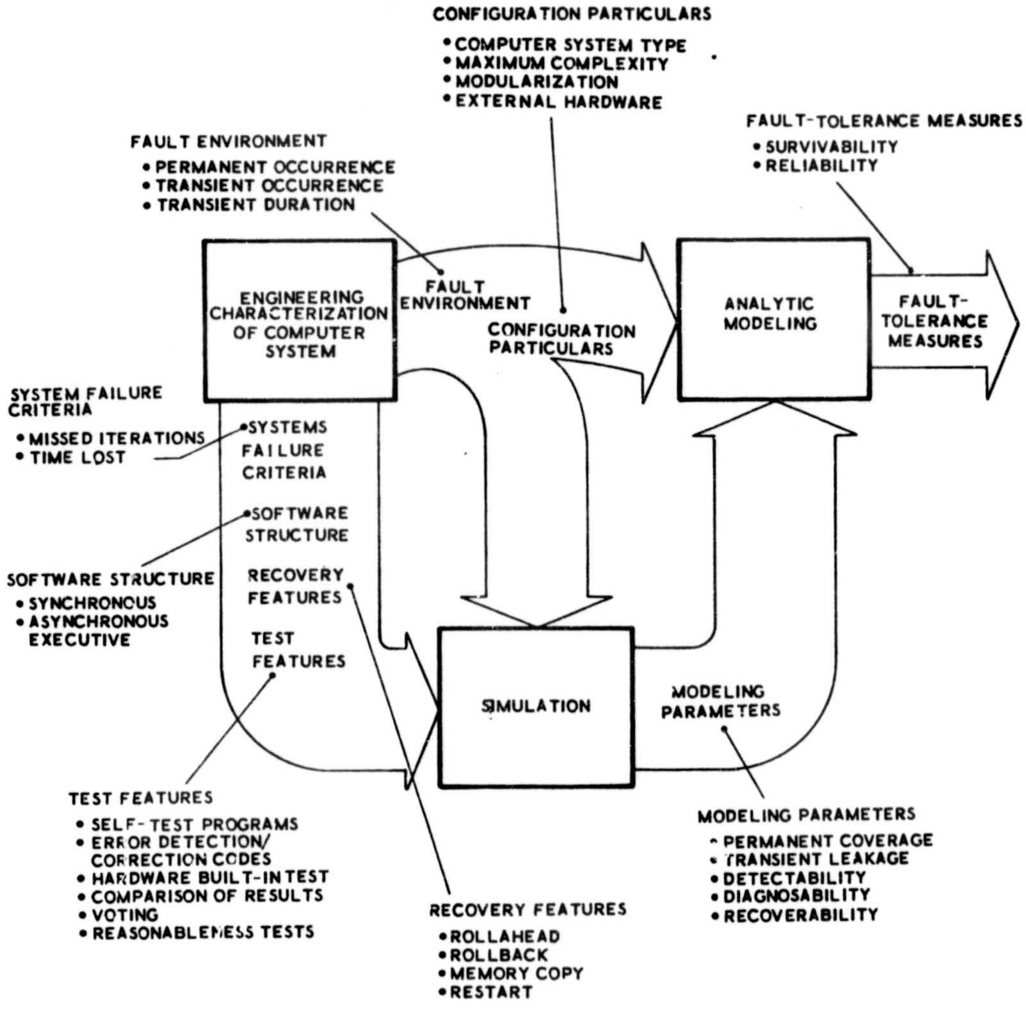


Figure 6-1. CAST activity sequence and information flow.

faults. The complexity of a system is overcome by dividing it into stages; a stage is a set of identical redundant modules. The reliability of each stage is described by a Markov model; a typical Markov model for a triplex stage is shown in Figure 6-3, where the potential states are shown for the stage ending in the states of either detected or undetected failure. The symbol λ represents the rate at which a stage transitions from one state to another. For example λ_{12} is the probability that any one

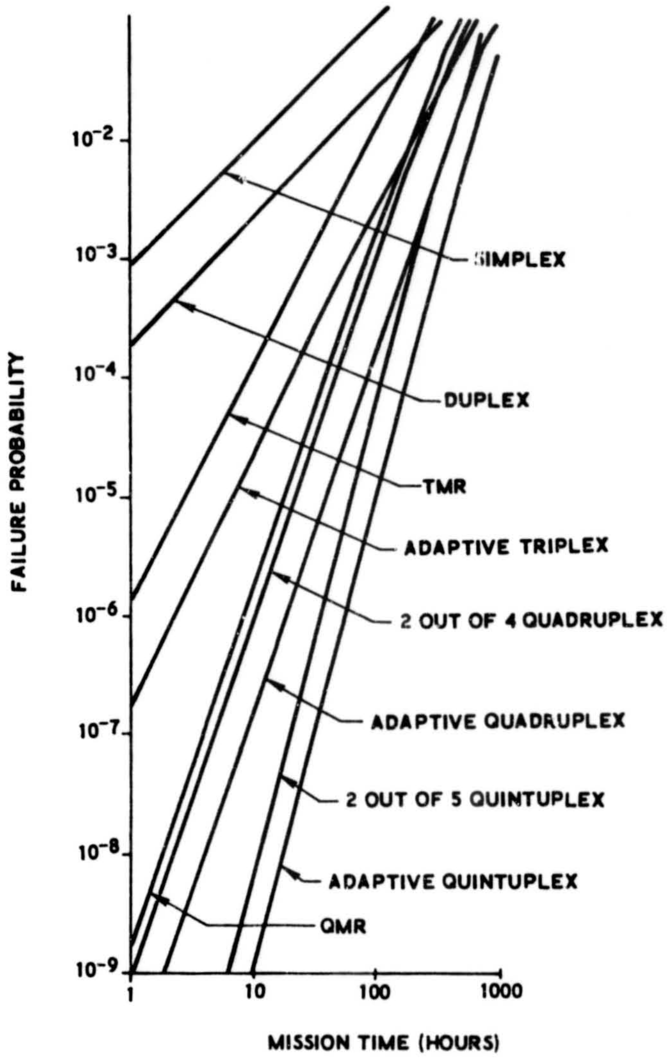


Figure 6-2. Effects of computer system redundancy and adaptability on failure probability.

module fails in the first stage. An assumption made in this particular model is that there is no transistion from state 1 to a failed state. In other words, there is no single-point failure mode. This model shows two possible transitions from the one failure state. In one case, a second failure causes the stage to fail, and in the other, the stage continues to operate on the one remaining good module. The ratio between these two transition rates is a function of how well the system can identify the failed module by self-test or other techniques.

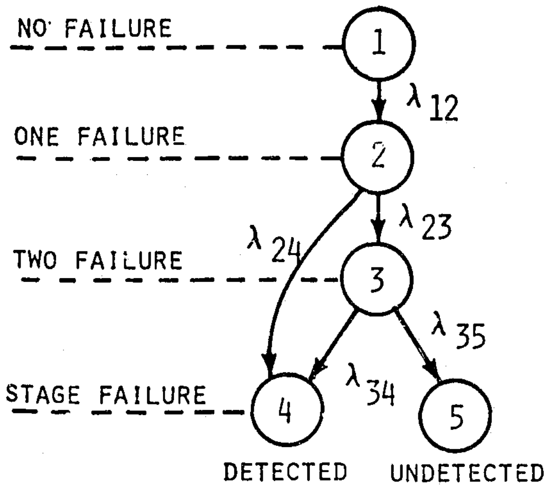


Figure 6-3. Typical Markov stage model.

The Markov models for the individual stages are related by a dependency tree as shown in Figure 6-4. This dependency tree shows how the failure of a module in one stage will cause the failure of modules in other stages. For example, the failure of a multiplex and A/D module will cause the loss of one set of modules of all sensor stages that provide information as analog signals. The numbers in each stage are the levels of redundancy. The circles on the right side indicate functional elements needed for the system to survive. The V indicates voting is used to combine the redundant signals. When the Markov models for each stage, the transition rates, and the dependency are defined as inputs to the CARSRA program, the program computes the functional readiness and failure probabilities for the system.

General analysis techniques such as CAST and CARSRA often prove to be difficult to apply to actual specific systems. More often than not, there is some system characteristic that is not well modeled by the general method. In most cases, systems are most efficiently analyzed by programs specifically tailored to that particular system. These general techniques, however, do provide insight into the failure characteristics of advanced systems, and provide a baseline from which the detailed analysis of specific systems can be developed.

Theoretical analysis and simulation can only provide an aid in support of the validation process. Theoretical analysis can derive the system failure rates due to the expected failure rates of components.

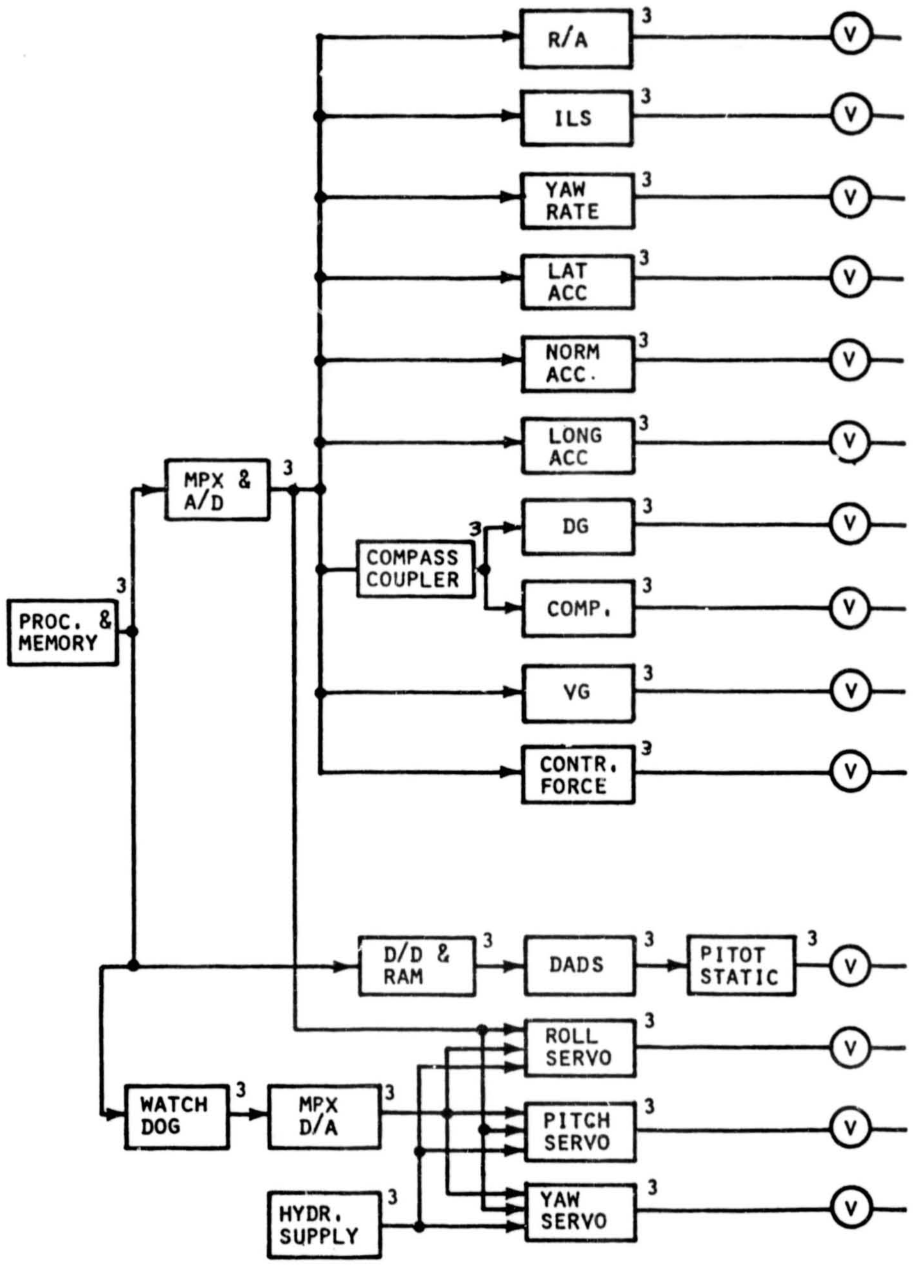


Figure 6-4. Flight-control system dependency tree.

Redundant fault-tolerant systems are purposely configured to account for these faults, and can be designed to achieve an arbitrarily low failure rate due to this source of failure. Although difficult, the problem of determining this system failure rate is a solvable problem, and these techniques help solve the problem. Of more concern are specification faults, design errors, and induced failures. Analysis and simulation methods have very little ability to detect these types of faults.

Present experience indicates that the primary validation method must exercise the actual system, including as many of the real peripheral devices as possible, with simulated stimulus inputs that are as realistic as practical. Every possible fault that can be imagined should be considered, and, where possible, induced during system tests. In order to effectively induce faults, the system must have been designed from the beginning to allow fault introduction. The system must be stressed to the maximum extent possible. The philosophy of the testing should not be to show that the system works properly but to try to make it fail.

This type of testing was the primary validation method used on the F-8 DFBW system. Even more detailed tests are recommended for future systems; one example is the introduction of transient faults. In the F-8 DFBW system tests, some of the simulated hardware failures were injected as "hard failures" only. Furthermore, no attempt was made to synchronize these failures within the control cycle. An expanded test would be made more detailed by

(1) Simulating failures as both "hard" and "transient".

(2) Vary the length and period of transient failures.

(3) Control and vary the relationship of the failure to the control cycle of the system.

6.2 Special Software Considerations

Software is one area where digital systems are unique and different from conventional systems. Thus, it is one of the areas of most uncertainty and concern, particularly for the people responsible for the validation and certification of these systems. In this section, the nature of software and its comparison to hardware characteristics are discussed. An outline is also given on some of the techniques that are being proposed to assure that reliable software is produced and adequately tested.

There seem to be two extremes of opinions on digital systems and software, particularly in what is required for system validation and certification. At one extreme is the opinion that the technology used inside the box should not influence how it is tested or certified. In this view, what is important is to show that the outputs from the system perform the intended function when the necessary inputs are supplied and the system is exposed to the required environmental extremes. At the other extreme is the opinion that software is somehow special and thus requires special consideration during test and certification. The most appropriate opinion would seem to be somewhere between these two, and would depend on the particular design and application. Consderable experience will have to be gained within the industry before the proper perspective can be achieved.

6.2.1 Nature of Software Failures

Software does not fail in the same way as hardware fails. Software cannot wear out or develop a fault as it operates, as does hardware. As a source of system failures, software problems are part of the software from the time it was written. However, the particular unique circumstance that would cause the problem to result in a failure may not appear until the system has been in service for a number of years.

To illustrate the problem of a latent software failure, take the hypothetical situation involving an airline with an inertial navigation system. When the software was written, a mistake was made in how negative latitude was handled. The airplane was used successfully in domestic U.S. service until years later when it was used on a South American charter, and there was a system malfunction when the aircraft crossed the equator. The magnitude of the failure in this hypothetical situation could have been either large or small. A major error may be less hazardous because the condition would be quickly noticed and the emergency dealt with by using an alternate means of navigation. The error could be in a more minor compensation term, which might not be easily noticed. The problem would not be helped by a dual or triple installation. Each system would have the same results. This example shows how an inherent software fault can look to the user like a hardware failure.

Since failures due to software errors are inherent in the design and are not due to wear out or random physical failures, software failures will have characteristics that are different from normal hardware failures. Hardware-failure rates normally follow the classical bathtub curve as shown in Figure 6-5. Hardware-failure rates are normally high at the beginning due to infant mortality and design errors. During useful life, the failure rate is usually constant, and the reliability can be characterized by a constant mature mean time between failure (MTBF). The useful life is ended by an increased failure rate due to wear out. Software, on the other hand, has a failure rate that will continue to drop as errors are discovered and corrected (assuming great care is taken to keep from introducing new errors when corrections are made). Software-failure rate is thus not constant and cannot be characterized by a constant MTBF.

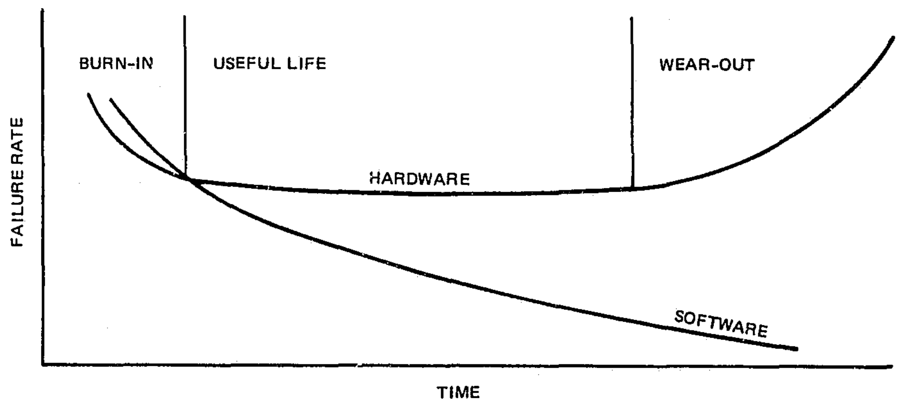


Figure 6-5. Comparison of hardware and software failure rates.

Software errors have the same basic nature as design faults in an analog system. The big difference is in the magnitude of the number of opportunities for errors. Although analog systems can become very complex, the probability that a design error will not be detected during test, verification, and validation is very small. The number of different paths through the system are numerous but more visible and manageable. The possible paths in a large computer program approach infinity. Exhaustive testing becomes impossible. It has been estimated that the execution of all possible paths through the TITAN missile navigation and guidance software would take 60,000 hours or about 7 years.(23) Even if this testing were done, it would still not prove there were no errors. There may be an error in interpretation of requirements or a missing path. A famous quote by Dijkstra(24) is: "Program testing can be used to show the presence of bugs, but never to show their absence!" It is virtually impossible to prove that a practical program is error free.

This possibility of a software error can create a distinction between digital and nondigital systems in some situations. With effective discipline in the preparation of programs, and with a well-constructed test program, the probability of a software fault can be reduced to a low level compared to the probability of a random component failure. In a situation where digital technology is replacing analog technology within a conventional unit, there may be little change because of digital technology. In other situations, a change to digital technology may have significant impact. For example, in a system where redundancy is used for high reliability, the potential for a software error that would be common to all channels is high enough that no one has yet had enough confidence to use digital in place of analog with no dissimilar backup. The problem of a generic software failure and potential solutions will be discussed later in this section.

6.2.2 Review of Software-Validation Techniques

It is worthwhile to review the software-validation techniques used and those not used in order to put the F-8 DFBW effort in perspective and also to make some predictions about future directions. The software-validation effort closely paralleled the hardware validation task. Several techniques were developed to improve the visibility of the software so as to make it more testable. This has, in a sense, made the software appear more like hardware, with the ability to apply test inputs easily and to make measurements on small pieces of the software which perform a limited and well-defined function. These techniques included:

(1) Providing access to individual software functions.

(2) Providing a real-time data-acquisition system with automatic verification plotting output.

(3) Providing internal trace areas and logs.

(4) Construction of special software driver programs for automated verification of complex logic (sensor and discrete redundancy management).

More significant than the test tools and techniques, however, are the design features of the system that make it testable. The technology is available to build parallel-channel synchronized digital flight-control systems. The ability to actually verify, validate, and flight qualify a system in a flight-critical application is very dependent on the experience of the designer, however. In systems that utilize very capable digital processors, the problem will be the maintenance of simplicity. The designer must produce a system design which is elegant because of its simplicity not because of its complexity.

Improvements are being sought in software testing, verification, and validation. Several tools are in various stages of development and use at the present time.(25) These include:

(1) Methods of verifying that all logical branches in code are exercised during the test program.

(2) Methods to automatically analyze a program to determine conformance to programming standards, rules, or good practices. These techniques also have the potential for diagnosing the code for certain classes of errors.

(3) Multicomputer ground-test hardware, which is specifically designed for debug, verification, and validation of real-time multicomputer software.

(4) Methods to automatically verify code against the specification, which is written in a form to be automatically compared with the code itself. The specification takes the form of a software language itself.

(5) Formal proofs of program correctness using mathematical logic. These proofs can become very complex and obscure in proving relatively small pieces of code. It is unlikely that they will provide significant assistance in improving software reliability in large real-time programs in the foreseeable future.

(6) Statistical estimation of software reliability. These estimates have not been well defined, but two techniques have been used.

One method of estimating software reliability is to observe the rate at which errors are found. Early in testing, of course, errors will be found at a rather high rate. As testing progresses, the number of errors will fall off exponentially, as indicated in Figure 6-5. By looking at this early history, some indication can be obtained of the rate at which errors are likely to continue to be found. A plot of this type was made for the F-8 DFBW as shown in Figure 5-51. It would not be possible, however, using this technique, to say how many more errors were present.

A similar method of estimating the number of errors remaining is to purposely embed errors in the program. The completeness of the software testing is then measured by how many embedded errors are found. This method also cannot prove that no other errors remain. The fact that someone recognized that a particular type of error was possible in order to plant it in the program means that he would also be likely to test for it. The most likely errors to remain are those that no one thought of as possible.

The best way to be sure there are no errors in software is not to put any in in the first place. The statement has been made: "the only way to assure that the last bug is found in a program is to never find the first—no matter how much the program is tested and used." A significant quotation from Dijkstra(24) is: ". . .the extent to which the program correctness can be established is not purely a function of the program's external specification and behavior but depends critically upon its internal structure." In other words, the correct operation of a system containing a digital computer—to a much greater degree than an analog system—cannot be assured by external tests but also depends on its original design.

It is, therefore, very important that software programs be written under a discipline that reduces the probability of error to the lowest level possible. Programming procedures are being developed that contribute to providing the necessary discipline. Examples of tools and techniques directed at producing code with fewer errors are:

(1) Standardized high-order languages.

(2) Structured programming (top-down) techniques.

(3) Specification languages that enforce the statement of unambiguous requirements.

(4) Computer architectures and instruction sets that are designed to enhance software testability.

(5) The use of standard routines for functions that are used in many applications (trig functions, control-system functions, etc.).

Even though some kind of programming discipline should be used in every development program, there is no one method that is well enough developed or obviously superior to any other method to justify its institution as a requirement. However, the regulatory authorities must be familiar with the methods being used, and must be able to make the judgment that good practice is being followed. The content of the software and its construction should be made highly visible both to the users of the system and to the certifying agency. The airlines expressed their desire in an ARINC Airline Electronic Engineering Committee meeting by saying:(26)

> The users (by virtue of being the customer, if nothing else!) have established their "need to know" and the need to be "functionally literate" in the area of software.
>
> The airlines made it quite clear to the manufacturers that software must provide "functional visibility" so that they, the airlines, are not left entirely at the mercy of the manufacturers' field service and support people. In colloquial terms, the airlines do not want the software to provide a smoke screen behind which a bearded wizard in tennis shoes will be able to simply hide his sins of omission and commission!

6.2.3 The Implication of Software Errors

The key question which must be addressed is this: can the reliability of the software be established at 1.0 after the application of the best development and testing tools? At the present time the answer is "no". The solution to this problem in a flight-critical control system has been and will continue to be a backup system that uses dissimilar software or a nondigital mechanization. An examination of several flight-critical control systems serves to illustrate this point (Table 6-1). Every primary digital flight-control system in existence or in the planning stage has a dissimilar backup system. As the table shows, this is not true of analog fly-by-wire systems. This points out the widely held opinion that some protection needs

Table 6-1. Flight-critical control-system configurations.

Vehicle	Primary System	Backup System
Lunar Module	Single-channel digital	Analog
F-8 Phase I	Single-channel digital	Triplex analog
F-8 Phase II	Triplex digital	Triplex analog
Shuttle	Quad digital	Simplex digital
YC-14	Triplex digital	Mechanical
F-4 SFCS (Research)	Quad analog	Mechanical; later removed
A-7 Digital (Research)	Dual digital	Mechanical
F-16	Quad analog	None
F-18	Quad digital	Multichannel analog and mechanical (pitch)

to be taken against the probability of a common-mode fault occurring, and that such a fault has a higher probability of occurring in software than in hardware.

It should be noted that no fault (software or hardware) has occurred in the F-8 flight software, which has disabled more than one channel. This is based on approximately 1750 hours of operating time. This does not say that such a latent fault does not exist in the software, however. It is for this fact that the bypass system will not be removed from the F-8, notwithstanding its proven performance.

Recently, there has been a flurry of interest in examining the potential of fault-tolerant software for flight-critical real-time control systems. Work carried out in university and government-sponsored studies has been directed at establishing the hierarchical approach to fault-tolerant software. The basic principle is simple in concept: if a software error is detected, transfer to an alternate algorithm. In actuality, however, the problem takes on many complications. Detection of software errors or common-mode software errors is a complex task. It presumes some system sanity, even to signal that a fault has occurred. Recovery from a common-mode fault in a multicomputer system is not straightforward. Nevertheless, this approach of providing backup software in the primary hardware appears to be a reasonable solution to the common-mode-software problem.

A conceptual design for such a resident backup software system for the F-8 system is shown in Figure 6-6. Existing hardware and software fault-detection alarms would be monitored by relatively simple external hardware devices, which would also contain a timeout function to detect "hung-up" software. It is presumed that a common-mode-software problem would affect all three machines within a relatively narrow window. A hardware monitor that detects coincident fault-detection alarms from all three channels would then trigger an external interrupt to its computer. A computer, upon receiving an external interrupt, would jump to a backup software program, which would provide only flight-critical software. Using its own I/O routines, it would set a discrete indicating it was now running in the backup mode. Assuming that a common-mode fault had actually occurred, each machine would set its corresponding discrete. A machine, upon entering the backup software routine, would check to see that its partners also had recognized the

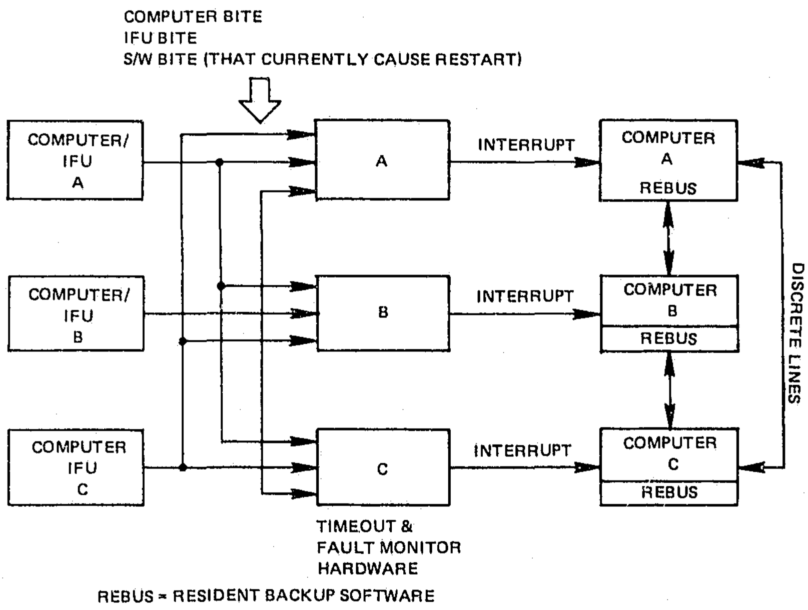


Figure 6-6. Conceptual design of resident backup software system.

fault. If so, each machine would proceed and execute its backup software in an unsynchronized mode of operation requiring no communication with its partners. A machine not finding its partners in the backup software routine would return to the main program and execute a restart.

This, of course, is a simplified description of the process. It illustrates, however, the future directions being examined as alternatives to independent backup hardware.

6.3 Combining Techniques into a Validation Program

An important and integral part of the development program for a new advanced flight-control system will be the validation process. In this subsection. some of the goals and requirements for validation are discussed along with a plan for the formation of a total validation program involving the integration of different methods and tools in order to cover the whole development process from definition of goals to production. This subsection and the report will end with candid remarks on certification by a test pilot who has been intimately involved in the validation process.

6.3.1 Comments on Validation Goals and Requirements

Before a well-designed validation program can be developed, the requirements for validation must be as well understood as possible. The entire validation process will not be, nor should it be, done in response to regulatory requirements. Validation will be performed by those developing the system to demonstrate to themselves and to potential customers, users, and regulators that a good product has been produced—that it performs well, is economical, and, above all, is safe. We are not concerned here with the performance or economic validation, but with the safety validation, which is motivated primarily by certification and product-liability requirements. The establishment of protection from product-liability risks may be the most demanding requirement for validation. However, any discussion of these requirements is beyond the scope and is not the purpose of this report.

The certification requirements for civil aircraft are given in the Federal Aviation Regulations (FAR). The present FARs were not written with some of the advanced flight-control systems concepts in mind. The FAA, under the Advanced Integrated Flight Systems Technology Program, is actively reviewing the regulations to be sure they are adequate to maintain flight safety when new systems are introduced, and that they do not unnecessarily restrain the introduction of valuable new technology.

It does not appear that the regulations will need any fundamental change. The parts of the FAR that are probably the most basic for advanced systems are 25.1301 and 25.1309. The essence of these regulations might be paraphrased by saying that equipment must do what it was intended to do and that it would be extremely improbable that a failure of the equipment would be catastrophic. The fact that these regulations are so broad and straightforward may be a distinct advantage. Real safety may be more enhanced than it would be by a multitude of detailed requirements where much effort was expended in proving these requirements were met rather than making sure that safety is achieved.

What is necessary, beginning from the very inception of a development program, is an understanding between the people developing the system and the people that will be asked to grant the certification on what will be required to show that the regulations are met.

An example of a need for understanding is the interpretation of "extremely improbable", and what would be required to show that it is met. The commonly accepted numerical value for "extremely improbable" is 10^{-9}. There is considerable controversy on the role numerical analysis should play in demonstrating that this requirement is met. In some situations, it appears that numerical analysis can have real significance and make a valid contribution. For example, numerical analysis can be used to compute the probability of system failure in a redundant system due to random-component failure. Random-component failure rates are large enough to be demonstrated in practice. The mathematical techniques for combining these failure rates are also well established. Numerical analysis showing a system failure rate of 10^{-9} per hour can then be believable. The actual value of the number can be significant in this circumstance. A change in this number can change the number of redundant channels required.

Numerical analysis may have little or no value in proving that the probability of failure is low due to other failures, such design errors, common-mode failures, and generic software errors. These classes of faults may be the most likely. A number like 10^{-9} may not be valuable as a legalistic number that must be "proven" with pounds of paper. It may be valuable as a positive goal toward which everyone strives.

The number 10^{-9} seems to be reasonable. It is likely that if advanced electronic flight-control systems can offer even a part of the advantages claimed for them, they will be used on virtually all aircraft for at least a generation. If it is assumed that an aircraft generation is at least 15 years, and with at least 6×10^6 commercial aircraft flight hours per year in the U.S. alone, a total of at least 10^8 system operating hours can be assumed. The number 10^{-9} thus means that the probability of a catastrophe due to a system failure is 1 in 10. We, as engineers, would not want to have a part in developing a system that was much worse than that. On the other hand, a smaller number may not be productive. Catastrophies due to this source would become so low compared to other sources that the required effort may make a greater contribution to safety in some other area.

A very low probability of failure for inanimate mechanical systems may be required by the general society. There seems to be a greater demand for reliability from mechanical systems than from human operators. It seems people can relate more readily to human failures. They realize that everyone, including themselves, sometimes make mistakes. These mistakes can sometimes be extremely tragic and must be held as low as possible. However, people are still willing to get on airplanes and accept the risk as part of the cost. They seem to be much less tolerant, however, of mechanical failure. There is always the suspicion that the failure was due to some conspiracy in big business and big government that allows inferior equipment to be used to maximize profits. In most cases, mechanical failure is also due to human failure during design, manufacture, or maintenance. The person or persons making the error, however, are not nearly as visible, and the personal consequences are not as high. The pilot usually pays dearly for his mistake. Thus, a relatively greater requirement is imposed on the aircraft and its equipment.

This great reliability for systems does seem to be a realizable goal and can contribute to a net increase in flying safety. We believe that the technology is available to meet this high reliability (although it may be difficult to "prove" that it does). As greater confidence can be placed in automated systems and they take over more of the functions now usually performed by human pilots, the accident rate can be reduced. The probability of an accident during low-visibility landings is now in the 10^{-5} to 10^{-6} range. If the probability of a catastrophe from a failure of an automatic landing system can truly be made to approach 10^{-9}, then accidents due to this cause can be reduced. Automatic systems can also greatly assist in continuing to reliably perform routine tasks when the human crew is distracted by some unusual situation which only a human crew can handle.

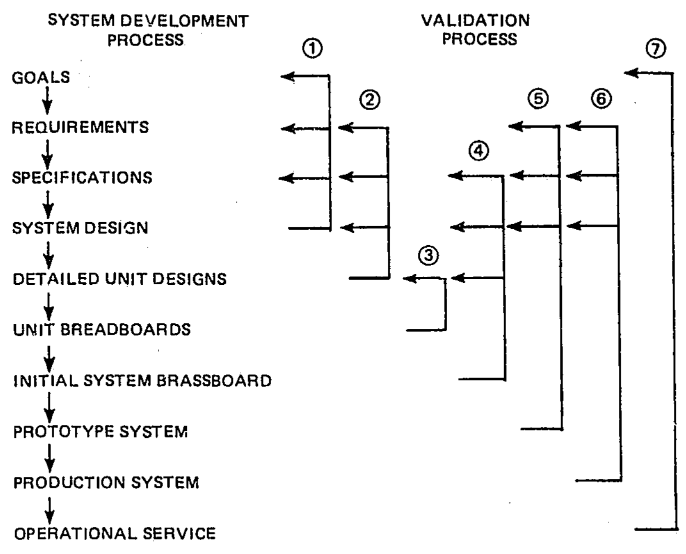
6.3.2 Comments on an Integrated Validation Program

The various validation techniques and tools that have been discussed, and others that have not been mentioned, provide aids in validating various aspects of a system. A set of appropriate methods must be selected and combined into a total validation program that provides complete coverage of all aspects of the system and the development process where errors may have occurred. The total validation will only be as strong as the weakest area.

The requirements necessary to validate the system and obtain a certification should be an integral part of the development process, beginning with the identification of needs, the definition of requirements, and the initial inception of design concepts. The ability of a system to be validated should be included in the basic tradeoff studies. The ease of validation should be weighed against other factors such as performance, cost, reliability, etc. A system design may be very elegant in its approach to fault tolerance, but if that fault tolerance cannot actually be demonstrated, the concept cannot be used.

The validation process will be a much more integral part of the design process for advanced flight-control systems than it has been for many of the conventional systems. In many cases with present systems, equipment has been designed to perform a function and meet a specification. The expected reliability of the system was then calculated after the fact, many times by a separate group within the organization. In advanced fault-tolerant systems, however, a major part of the design effort is involved with determining how the system can fail, and with designing methods to accommodate these failures. Thus, validating the systems becomes very nearly the same thing as confirming the basic design.

A simplified diagram showing an example of a total validation program is shown in Figure 6-7. One possible breakdown for the phases of system development is shown down the left side of the figure. The various stages of the validation process are shown along the right side. The arrows represent how each stage in the design is validated



NOTES:

① PAPER ANALYSIS, ABSTRACT MODELS, HIGH-LEVEL SIMULATION
② FMEA, DETAILED SIMULATION
③ VERIFICATION, FMET
④ HYBRID SIMULATION, FAULT INJECTION, STRESS TESTING
⑤ QUALIFICATION, FLIGHT TESTS, FULL CERTIFICATION
⑥ ACCEPTANCE TESTS, REVALIDATION
⑦ CUSTOMER ACCEPTANCE, STATISTICAL LOGS

Figure 6-7. Diagram of validation process.

against the appropriate previous stages. As the process proceeds, the validation methods become more detailed and systematic, culminating in the exhaustive validation that forms the basis for certification. The final stage is when the customer accepts that the system meets his original goals. Some of the validation methods that might be used at each stage are noted. This example is not intended to be a definitive description of a complete validation process. It is intended rather to illustrate how a total program might be put together and how the different parts interrelate with one another. A detailed program would have to be worked out for each system development effort.

One final comment is made in the form of a proposal, which could reduce the probability of an error source being overlooked and increase the public confidence in advanced systems. This proposal is the use of independent assessment by some competent third party that is not a part of the basic developing organization. This assessment would be of the same nature as an independent financial audit or an Underwriter's Laboratory approval. The entire development and validation process is often performed within one organization with the only review being performed by the certifying authority. The certifying authority will not normally have sufficient manpower to perform a thorough review. With one organization performing the major part of the effort, there may be a predisposition or constraint that would cause a potential error source to be overlooked, in essence a common-mode failure.

The independent assessment would have to be equal in competence to the original design effort. The most competent people in one organization would probably be assigned to the design team in the first place. A potential candidate for independent assessment would be anouther airframe manufacturer that was not a direct competitor. The independent assessment would be submitted along with the original validation data as part of the certification application.

This process might greatly enhance the using public's acceptance of a complex automated system. There would still be the chance that a failure mode would be missed. However, this oversight would be the result of human error of people who could be identified within two competent organizations. In this situation, the risk would be easier to accept as part of the price for progress.

6.3.3 Test Pilot's Thoughts on Fly-By-Wire Certification

The thought of having to step in and certify someone else's fly-by-wire flight-control system is sobering, indeed. So much work will have gone into the system that an outsider would have a very difficult time understanding the system unless it were verbally disassembled before him bit by bit and wire by wire.

The responsibility is awesome—but there is a way. It means getting involved early as a close observer of the development process and this required action may be without precedent. Expect terms like Mean Time Between Failures to have reduced significance. Substitute instead the ability to control the craft after multiple adversities.

Trite as it sounds, let's start at the beginning. The operations research method is not a bad way to start the design of a fly-by-wire system. If it is determined that a fly-by-wire system will be designed, then all the affected parties should meet to sketch out the guidelines for the system. Each group can list and discuss their requirements and modify them to be compatible. This is a pivotal time, because an error or omission at this point is paid for dearly as the system develops. This group should include the systems people, contractors, software/computer people, pilots, airline representatives, and possibly the eventual certifying agency representatives.

There is a team learning concept here. The early involvement of the certifying agency at several levels provides exposure, personnel backup, completeness, and a feeling of openness. The position of the observer must be strictly defined to be an observer, the danger being that of interfering or losing objectivity by subconsciously becoming part of the team.

Simulation, properly used, will eliminate a lot of potential problems. It is both an engineering tool and a demonstrator. Engineers and pilots can learn the system's real characteristics from it. In particular, the system should not be coddled in any way, because it won't be when it is eventually flown. Here is a chance for everyone to exercise the "what if this happens" type of thinking that is ultimately so productive. The more people who fly it the better the end product, for the system will eventually be flown by the good and the not so good.

The question of burn-in hours is bound to come up. How much is enough—and why. Burn-in will be accomplished on the simulator if the simulator has been configured to use the flight electronic hardware (a must!). Mean time between failure now takes on reduced meaning. Obviously one shouldn't go into flight test with a known flaw, but if a sufficient level of protection is available (determined from the simulator) one can fly with an untried system, thus gaining the experience to be able to fix its emerged faults and eventually certify the flight-control system.

The preflight procedure should be short but complete. The purpose here is twofold:

(1) To check the system integrity.

(2) To let the air crew examine the flight-control system and gain their confidence that it is ready to fly.

Actual flight test seems to start a new cycle of problems and decisions. The manufacturers/owner makes the decision about when the flights begin. After that, it would be desirable to have a pilot from the certifying agency participate in some of the flights, to gain exposure and confidence in the system. Problems are certain to develop during flight test, but they will be surmounted—only the question of how to do it is debated.

The handling of emergency procedures must always be in line with a human's natural reaction, because in high-stress quick-response situations, the aircrew will always revert to type—regardless of how long or intensely they have been trained. The system ideally should take each corrective step and advise the pilot by illuminating a warning/advisory lamp. A manual override may be provided to allow the crew to quickly select the most reliable emergency mode at their option.

Who handles the emergency? Simulation will probably answer that. It depends upon the size of the crew, manufacturers' decisions, airline policy, etc., and probably doesn't matter—except that the person should always be in place during flight.

A word about warning lights. They must <u>always</u> say what it is or maybe what to do or what has been done. The message must be clear and unambiguous. There is a switch in one of today's fighters which is labeled "Flight Control Disconnect". Thank God it does not disconnect the flight controls! This mistake should not be made with the lights. Remember, the crew may be fortunate enough not to see these warning lights very often and they shouldn't be confused.

It is my personal opinion that simulation will reveal which lights will ultimately be required. I also think the user should have the final say about lights, hopefully by explaining why they are/are not needed.

The airplane will eventually fly just the way the project team wants it to fly. It will undoubtedly appear very conventional to the passenger but will reduce the workload of the crew.

Should the airplane have a dissimilar backup flight-control system? A difficult question. For a passenger-carrying aircraft I would say yes. Many years of experience will be needed to bring out all the faults in the system and an analog "get home" backup may save the reputations of the manufacturer, the airline, and the certifying agency.

Lightning is a serious unknown. I think the test airplane will have to seek out lightning and take many strikes. I think this should be a mandatory part of the certification procedure. It shouldn't be left for the airline pilot making an approach to JFK with your mother-in-law on board. Or should it?