NASA Technical Memorandum 86807

NASA-TM-86807 19860009854

Flight Test of a Resident Backup Software System

Dwain A. Deets, Wilton P. Lock, and Vincent A. Megna

January 1986

National Aeronautics and
Space Administration

NASA Technical Memorandum 86807

Flight Test of a Resident Backup Software System

Dwain A Deets and Wilton P Lock
Ames Research Center, Dryden Flight Research Facility, Edwards, California
Vincent A Megna
Charles Stark Draper Laboratory, Cambridge, Massachusetts

1986

National Aeronautics and
Space Administration
Ames Research Center
Dryden Flight Research Facility
Edwards, California 93523

FLIGHT TEST OF A RESIDENT BACKUP SOFTWARE SYSTEM

Dwain A. Deets and Wilton P. Lock
NASA Ames Research Center
Dryden Flight Research Facility
P.O. Box 273
Edwards, California 93523-5000
U.S.A.

and

Vincent A. Megna
Charles Stark Draper Laboratory
Cambridge, Massachusets 02139
U.S.A.

SUMMARY

A new fault-tolerant system software concept employing the primary digital computers as host for the backup software portion has been implemented and flight tested in the F-8 digital fly-by-wire airplane. The system was implemented in such a way that essentially no transients occurred in transferring from primary to backup software. This was accomplished without a significant increase in the complexity of the backup software. The primary digital system was frame synchronized, which provided several advantages in implementing the resident backup software system. Since the time of the flight tests, two other flight vehicle programs have made a commitment to incorporate resident backup software similar in nature to the system described in this paper.

NOMENCLATURE

ADFRF	Ames-Dryden Flight Research Facility	IFU	interface unit
CIP	computer input panel software	I/O	input/output
CL1	control law software (includes output commands)	LED	leading edge down
		LEU	leading edge up
CSDL	Charles Stark Draper Laboratory	N_z	normal acceleration, g
DFBW	digital fly-by-wire	REBUS	resident backup software
DOWNLINK	data recording software	RM1	redundancy management software (part 1)
FAIL1	location of first failure insertion point	RM2	redundancy management software (part 2)
FAIL2	location of second failure insertion point	SYNCH XLINK	synchronization and cross-link software
FRR	flight readiness review		

INTRODUCTION

Digital flight control system designers have been concerned with protecting against "generic" software errors for as long as digital flight control has been in existence. The standard approach has been to provide a dissimilar hardware backup control system, usually analog in implementation. Although a dissimilar hardware backup provides a way of maintaining control of the aircraft in the event of generic software errors, it is a costly solution that fails to take advantage of the unfailed primary system hardware assets. It also requires a redundant switching mechanism to accept system commands from the active system and pass these commands to the redundant actuators. In the case of analog backups, the disadvantages can be even greater because of the need for sensor inputs that are compatible with both digital and analog systems. This usually means a separate set of sensors are needed for each system.

Several approaches have been advocated for handling software errors without requiring independent backup hardware. One approach is to utilize dissimilar software in each of the redundant hardware channels. This approach has been used with success on the A310 airbus (Ref. 1). Although successful for that application, this approach results in a system with wider tolerances on the tracking between channels that may have disadvantages in other applications. A second approach is to implement identical dissimilar software sets within each of the redundant hardware channels. This has been referred to by various names, the most common being "recovery blocks." The resident backup software (REBUS) described

in this paper is of this second type. It has the advantage of minimizing hardware and retains the advantages of relatively tight tracking between the unfailed hardware channels.

Considerable debate has gone on concerning the amount of dissimilarity necessary to assure coverage for the generic software error. In this debate, a distinction is made between several classes of software errors and the amount of dissimilarity necessary for protection for each of these error classes.

One class of generic error is the design error, in which the specified design is not adequate to handle the environment encountered in the actual application. As an example, the coupling between a structural mode and the aerodynamics is sufficiently different from that simulated in ground tests that the resulting closed loop through the control system becomes unstable.

The second class of generic software error is the implementation error, which results in a system that does not meet the specified design. A particular difficulty concerns the latent system or software defect that goes undetected in testing but is exposed under certain environmental conditions experienced in flight. These implementation errors can occur at a number of different levels· in the algorithm chosen by the designer, in the way the software coder writes the source code, or in the compiler, which converts the source code to machine language. In every case, these errors would go undetected throughout ground testing and manifest themselves in an unsafe manner in actual flight. Implementation errors in the compiler are usually considered to be least likely, a compiler receives a considerable amount of implicit testing as a result of all the code that is generated and exposed to a wide range of conditions during the extensive verification and validation that are characteristic of man-rated digital fly-by-wire systems.

Even when care is taken to assure dissimilarity for each level of implementation error, precautions must be taken to assure a successful transfer to the dissimilar software because of the nature of the digital computer and the unforeseeable consequences of a software error. This is most important during initialization of the dissimilar software because vehicle state data are needed to establish proper startup conditions. The danger exists that the software error in the primary system has caused contamination of the vehicle state data, thereby raising the possibility of initializing the backup software at an unsafe condition.

Potential problems of this type have been of concern and have been some of the principal reasons that designers have chosen to use the better understood, albeit more expensive, analog backup. The flight experiment described in this paper was done to reduce the risk associated with backup software, with the hope that this more cost-effective method will be suitable for use in future flight control system designs. The experiment utilized the F-8 digital fly-by-wire (DFBW) airplane operated by the NASA Ames Research Center's Dryden Flight Research Facility (ADFRF) at Edwards, California. The system implementation was accomplished by the Charles Stark Draper Laboratory (CSDL).

FLIGHT EXPERIMENT OBJECTIVES

The F-8 DFBW airplane is operated as a flight facility for evaluating new concepts, such as REBUS, in the real flight environment. One key step in preparing for a flight test is the independent review by people with broad operational experience in flight test. At ADFRF, this review process is called the flight readiness review (FRR). The value of evaluating a system or concept in flight is that the system will, by necessity, be subjected to close scrutiny and thorough testing to receive a favorable response from the FRR committee. It will also be subjected to the actual flight environment, which will be different to some degree from that experienced in ground test. Those differences, and their effect on the system behavior, are important in understanding the concept under evaluation.

These considerations contributed to the establishment of the experiment objective· to thoroughly evaluate a high-potential approach to fault-tolerant software utilizing a dissimilar software flight control system concept, subjecting it to all the processes necessary to qualify it for flight and validating those processes through subsequent flight test.

The REBUS concept was selected for evaluation because of its system-wide efficiency gained by utilizing the same computing hardware used by the primary system.

F-8 DIGITAL FLY-BY-WIRE SYSTEM

The F-8 DFBW airplane (Fig. 1) was modified by installing a fail-operate, fail-safe digital fly-by-wire system. (A detailed description of the system is presented in Ref. 2.) It is useful to provide some details in a simplified form to make explanations of the REBUS modifications easier. The DFBW system contains a triplex digital primary system with a triplex analog computer bypass system serving as its backup. Figure 2 illustrates the interface between the two systems. The triplex primary system utilizes redundant digital computers, and CSDL provided interface units (IFUs). Redundant aircraft motion sensors and pilot stick transducers are conneced to the IFUs. Surface commands are passed through an analog midvalue voter, with the midvalue of the three channels passed on to the servo drive electronics. This midvalue voter, being external to the digital computer, has additional analog circuitry that can declare any one of the digital output commands as failed and disable the particular channel output at the midvalue input plane.

Analog Backup System

The analog backup is a direct electrical link between the pilot's stick and the servo drive electronics. The redundant commands are passed through the same midvalue select devices utilized by the primary system. This system, forward of the midvalue voter, is called the computer bypass system.

Primary System

The primary system is frame synchronized. This synchronism facilitates the exchange of data between computers and the input/output (I/O) function that distributes sensor information between computers, such that each receives a copy of all available sensor inputs.

The primary software mechanizes a number of different control laws for the aircraft. They are the direct mode (control stick to surface actuator without any feedback) and stability augmentation system mode for the pitch, roll, and yaw axes. The pitch axis also contains the command augmentation system for further improvement in handling qualities. Finally, there are the standard autopilot modes of aircraft control.

The input sensor set includes not only pilot stick and pedal sensors and vehicle motion sensors, but also information on surface commands and positions. In particular, the left and right elevator positions and the last surface command to all five control surfaces are available in the input sensor buffer located external to the digital computers in the IFU. This control surface information is useful in the initialization of the REBUS software.

The primary software contains a capability of recovering from transient errors. This capability (known as "restart") initializes the control laws after a potentially resettable failure has been indicated, and it attempts to recompute, anticipating that the failure will not reoccur. The software has been set up to recycle through the restart 10 times. If the failure still persists after 10 cycles, the channel declares a channel failure and is voted out by the remaining good channels. If one channel has already failed, the flight control system would be forced into the computer bypass system.

Of the various conditions that can create a failure condition, two cause immediate failure declarations, and one (designated "execution error") is potentially resettable. These conditions are the following

1. Failure to perform an I/O function within 53 msec of the last I/O (the control law frame time is 20 msec)

2. Failure to update the watchdog timer

3. Execution error for 10 consecutive program iterations or once every 6 iterations for 306 iterations

The reason for attempting 10 restarts before declaring a channel failure is the time duration that can be tolerated without a surface command update. For the F-8 aircraft with stable bare airframe, a 300-msec time period can be tolerated, 10 restarts require between 100 and 300 msec, depending on where the failure indication occurs within the compute cycle.

REBUS SYSTEM DESIGN

Several tradeoffs must be made before specifying a system design. Many are similar to those made traditionally for an independent hardware backup. Issues such as complexity of the backup, criteria for automatic transfer, and interchannel synchronization must be addressed. Other issues that are specific to this experiment, in that an existing system was modified, will not be discussed.

Complexity

Selection of control laws is usually based on the minimum necessary to provide return to the base and safe landing capability. In the case of the F-8 DFBW aircraft, an unaugmented backup would be adequate to provide the necessary functionality. For this experiment, the decision was made to include slightly more capability than the bare minimum to better represent modern airplanes, which generally require augmentation to some degree. Three-axis fixed-gain rate damping was selected. Figure 3 illustrates, in the form of a block diagram, the pitch-axis stability augmentation system. As can be seen, there are several nonlinear functions, such as stick shaping and dead band. The only deviation from fixed gain is the selection of a landing approach gain set when the wing is put in the up position and the control surface authorities are modified. (The standard F-8 aircraft provides for two wing-incidence positions, the wing-up position providing improved pilot visability during the landing approach.) The other axes are similar in complexity.

The control law computation cycle could be mechanized quite simply by using a straight in-line code and very few branches. Many of the self-check functions, such as sensor redundancy management, could also be eliminated to reduce complexity. The overall complexity was reduced considerably relative to the primary system. For example, the REBUS software required less than one-tenth the memory that the primary software required. The compute cycle was selected to be 50 samples per second, the same as that of the

primary system (for convenience, in that it reduced the amount of separate stability analysis and validation necessary).

Transfer Criterion

A criterion was needed for automatic transfer to backup, assuming that a generic software error could cause a situation requiring action sooner than the pilot can react. The appropriate solution was to use the channel failure declaration, which is based on repeated self-execution error indication. Failures to execute while in the control law code are representative of the failure mode for which the backup software was intended, in that the system behaves in the way that a generic software failure would be expected to manifest itself.

It is desirable to minimize transients that occur upon transfer to backup if this does not contribute to increased complexity or other undesirable effects. Given that the system failure has occurred because of a generic software error, one must also assume that there is a finite probability that the record of the aircraft state as maintained in the memory of the system computers has been degraded by the failure. Therefore, one must look elsewhere for the information required to initialize the backup software.

The only information that the backup software needs to take control of the aircraft without introducing an objectionable transient in aircraft state is the current position of the aircraft control surfaces. This allows for some small transients that result from reinitialization of any active filters and changes in loop gains from state-dependent to fixed values.

Three assumptions were made relative to initialization of the REBUS software.

1. Primary software failure occurs simultaneously in at least two channels of the system.

2. The trigger mechanism generates simultaneous pulses for at least two channels of the system.

3. At least two channels simultaneously execute the input sensor I/O function.

Given these assumptions, the backup software can be initialized to the aircraft's existing control surface commands, and thus it generates no transient other than that due to gain changes. This is possible because the simultaneous performance of the initial input sensor I/O function by the computers enables them to receive a full complement of sensor signals, that is, not only their own dedicated set but those dedicated to other computers.

Return to Primary

A major issue for any backup system designer is whether return to primary should be allowed. If the primary system has suffered a major fault, then it must be assumed that transferring back to that defective system is unsafe. On the other hand, if the backup software is not providing a controllable airplane, the pilot may want to try the primary software in a last-ditch effort. Thus, both sides of the issue could be argued without reaching a clear resolution. For an experimental system such as the F-8 REBUS, it is desirable to transfer into backup even though no serious problem exists with the primary. For test purposes, it is highly desirable to be able to transfer back and forth at will during a given flight. For these reasons, provisions were included in REBUS and the F-8 DFBW primary system for transfer back to primary from REBUS.

Interchannel Synchronization

Since the primary synchronization algorithm is a software function, the REBUS must provide a dissimilar synchronization algorithm if a synchronized backup system is to be established. The dangers associated withnot being able to synchronize upon transfer to REBUS must be considered. Also, the additional complexity associated with synchronization makes it less desirable from a system complexity standpoint. For these reasons, it was decided to establish REBUS as an asynchronous system. This decision has several ramifications (such as, no exchange of data between computers can be performed because of the complexity of adding an asynchronous data transfer capability).

Sensor cross-trapping is used in the primary system to ensure that each channel operates on identical data. With the asynchronous approach in the REBUS, each computer must operate on dedicated sensors that are independent of the operation of the other computers. Because of this interchannel independence, it is of interest in both ground and flight tests to see how much variation develops between each channel.

IMPLEMENTATION

The specific details of the REBUS implementation were strongly dependent on the peculiarities of the F-8 DFBW system. Some functions were included in the total modified DFBW system because it was a flight experiment and it was required that tests be performed easily. These details include the triggering mechanism, the approach to simulating software errors, and the status of the computer bypass system.

REBUS Triggering

The failure detection logic in the primary system causes the generation of a discrete output from the computer channel that is indicating a failure. This discrete output is processed by external circuitry unique to each channel computer. This circuitry sets a channel failure indicator, which is then passed to the other channels. This discrete was convenient for use as the trigger generator, therefore, a circuit that voted the discrete output from each computer was constructed for each channel. If two of three computers would generate a discrete, the trigger pulse would be produced for that channel. Figure 4 illustrates the relative location of the added hardware including a switching card in the IFU and a transfer circuit external to the computer. Also shown is the REBUS memory in the primary digital computer.

This trigger pulse must be introduced into the computer to initiate the execution of the backup software. The pulse could not enter as some external interrupt, for the computer could fail in such a way that all interrupts would be masked. The only "interrupt" that would be recognized by the computer at any time and under any conditions was the system reset.

Each channel has its own trigger circuit. This mechanization provides protection against inadvertent switching of the system from primary software to backup software. Two of three computers must be generating a channel fail to produce the trigger pulse. If a particular trigger circuit fails and generates a pulse when none is commanded, only that channel's computer switches to backup, and any discrepancy between its outputs and the other computers' outputs will be voted out by the external mid-value voters. Failure to generate a pulse when one is commanded will not prevent the other two computers from switching to backup software.

Software Programming

Ideally, it would have been desirable to have people other than those involved in primary software development mechanize the backup software. Also, it would have been advisable to use software development tools different than those used for the primary software. Neither one of these steps for enhanced dissimilarity were used for this experiment because of the added time and cost associated with them. There would have been no way to evaluate the effects of these additional steps within the limited scope of this experiment, so it was not deemed necessary to include this additional expense.

Simulation of Errors

For the purposes of this experiment, a simulated generic software error was needed. Several types of generic software failures were studied as to how they would manifest themselves as apparent hardware failures. It was apparent that a generic software failure that would be of sufficient severity to bring the primary system down would be such that a restart would result. This can be simulated by adding an instruction to write to a protected portion of memory. Segments of memory can be protected from being written to. Thus, the simulation of a generic software error was implemented by including a pilot-selectable branch in software that included a write instruction into a protected portion of memory. Several different points within the control law code were selected for error insertion. The error insertion function was subject to an arming device, also pilot selectable.

Computer Bypass

Since the unmodified F-8 DFBW already had a backup provided by the computer bypass system, the question arose as to whether the computer bypass function should be removed when the REBUS was added. Retaining the computer bypass would be in effect a backup on a backup. However, a strong argument existed for not removing it because of the reliability record of the specific prototype primary computers available to the F-8 program. With the computer bypass system, the total system hardware reliability is adequate for safe operation. Thus, the computer bypass system was retained for its contribution to hardware reliability.

GROUND TESTS

An important step in evaluating the REBUS system was the F-8 Ironbird simulator, which utilizes a decommissioned F-8 aircraft with a complete DFBW system installed. The aerodynamics are simulated in a general-purpose simulation computer. The specialized hardware necessary for the REBUS was implemented in brassboard versions of the interface units.

Part of the ground tests was the determination of transfer transients for a number of failure insertion points within the control law code. The only significant difference was whether the failure was inserted before or after the actuator command output. Figure 5 illustrates the situation for a typical 20-msec control compute cycle. The end of CL1 (control law software module) represents the point at which the actuator output command occurs. The first failure insertion point is represented by FAIL1 (at the end of RM1, the first part of the sensor redundancy management software module). Other software modules include SYNCH XLINK, which performs the synchronization cross-link, RM2, which is the second part of redundancy management, and DOWNLINK and CIP, which process the data recording and the pilot's computer input panel, respectively. Another failure insertion point was FAIL2, representative of a failure occurring after the surface output commands. As in the ground tests, if the failure occurred prior to the surface command update, such as at FAIL1, the aircraft sustained a longer period of time without

surface commands, and the backup software was initialized with relatively old aircraft state information. Depending on the aircraft motion at the time of the failure, the transient on transfer varied. In no case was the transient considered severe.

The time period with no control, between 240 and 300 msec, is a function of the amount of time the primary system is allowed to attempt to correct itself. Figure 6 shows (in a simplified form) how a fault occurring at the end of DOWNLINK and CIP would be repeated through 10 attempts to restart while still in the primary system. For this case, the transfer time was 294 msec. For the F-8 aircraft, this time period is short enough that it does not cause any control problem. For other airplanes, the allowable time period may be much shorter, in which case a smaller value for the failure counter should be selected, assuring controllability, however, this will diminish the system's tolerance of transient failures that can be withstood without transfer to the backup.

FLIGHT TESTS

At the ADFRF flight readiness review (FRR), most of the details of the design and the results from the ground tests were reviewed without significant comment. There was one major concern raised that may have implications for others. This concerned the plans for enabling the REBUS after safely at altitude. The FRR committee questioned the advisability of taking off the first time with the REBUS enabled. The committee felt that the system should not be enabled until a safe altitude was achieved. The committee also raised a question as to the expected transient if the system was forced to REBUS at liftoff, given that the REBUS was enabled. It was finally determined that the more conservative approach was to wait until a safe altitude was achieved before enabling the system.

Once the FRR committee approved the plans, the flight experiment began. The first flight was on 23 July 1984. A summary of the flights is presented in Table 1. The emphasis of the experiment was on comparing flight with simulation. This included the tracking between channels, transfer transients, susceptibility to unwanted transfers, and general operational factors.

The tracking between channels was very close. At no time during the flight tests did drifts occur between the three channels. This was in agreement with the simulation runs conducted prior to flight.

The transfer transients were negligible, even when occurring during elevated-g maneuvers. In most cases, the transfer to REBUS could not be detected in the control surface traces. Figure 7 shows a typical time response in the pitch axis, which illustrates the excellent performance for a transfer during a 3.5-g turn.

There were no unwanted transfers encountered during the flight tests. This was to be expected because the transfer mechanism was the same as had been used to cause a transfer to computer bypass in all the previous flight tests with the F-8 DFBW airplane. There had never been a transfer to computer bypass over an eight-year period of flight testing.

From an operational standpoint, no significant concerns arose. Evaluation of the handling qualities for the REBUS control laws was included under operational assessment. Two pilots evaluated the system, evaluating it as acceptable for emergency operations and preferable to the computer bypass mode.

EXTENSION OF CONCEPT TO OTHER APPLICATIONS

The experience gained with the F-8 REBUS flight experiment has raised some issues relative to future applications. Many of the recommendations that could be made apply to dissimilar backup systems in general. We will restrict the comments here to issues applicable only to resident software backups.

Much of the REBUS design was dependent on the synchronous nature of the primary system. If the primary system were asynchronous, it would be valuable for initializing the REBUS to have all surface positions available in the input data set.

Concepts similar to REBUS have already been incorporated into the designs for other flight applications. The X-wing rotorcraft program (Ref. 3) utilizes a jam-transfer software backup that is nearly the same as REBUS. The F-16C and D aircraft with digital fly-by-wire systems will have resident software backups in the primary system memories.

CONCLUSIONS

The F-8 resident backup software (REBUS) flight experiment has demonstrated a cost-effective approach to providing dissimilar redundancy to protect against generic software failures. The major findings are as follows

1. Resident backup software that provides protection for primary software errors can be implemented.

2. Transients that occur during transfer from primary to backup can be minimized with little impact on system complexity.

3. Independent reviewers with broad operational background in flight test can be satisfied that resident software backups offer adequate assurance of flight safety.

Recent incorporation of backup software approaches similar to REBUS into the designs of upcoming flight vehicles, coupled with the successful results from this flight experiment, offers evidence that this concept will find industry-wide acceptance as a viable solution to the generic software error problem.

REFERENCES

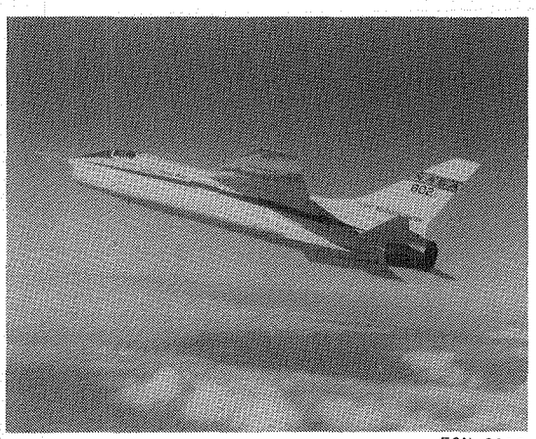
1. Hills, A.D.: Digital Fly-By-Wire Experience. Fault Tolerant Hardware/Software Architecture for Flight Critical Function, AGARD-LS-143, Paper 2, Sept. 1985.

2. Szalai, Kenneth J.; Jarvis, Calvin R.; Krier, Gary E.; Megna, Vincent A.; Brock, Larry D.; and O'Donnell, Robert N.: Digital Fly-By-Wire Flight Control Validation Experience. NASA TM-72860, 1978.

3. Guertin, R.G.: Development Status of the RSRA/X-Wing — Rotor System Research Aircraft. AIAA-85-4008, AIAA Aircraft Design Systems and Operations Meeting, Colorado Springs, Colorado, Oct. 14-16, 1985.

Table 1. REBUS FLIGHT SUMMARY

Number of flights in REBUS	6
Total flight time for these 6 flights	6 h 50 min
Total flight time in REBUS	3 h 54 min
Number of transfers to REBUS	22
Number of transfers to primary	18
Number of transfers at >1 g	6
Highest g level at REBUS transfer	3.5 g
Number of low approaches in REBUS	6
Number of touch and gos in REBUS	10
Number of landings in REBUS	5



ECN 3312

Figure 1. F-8 digital fly-by-wire airplane.

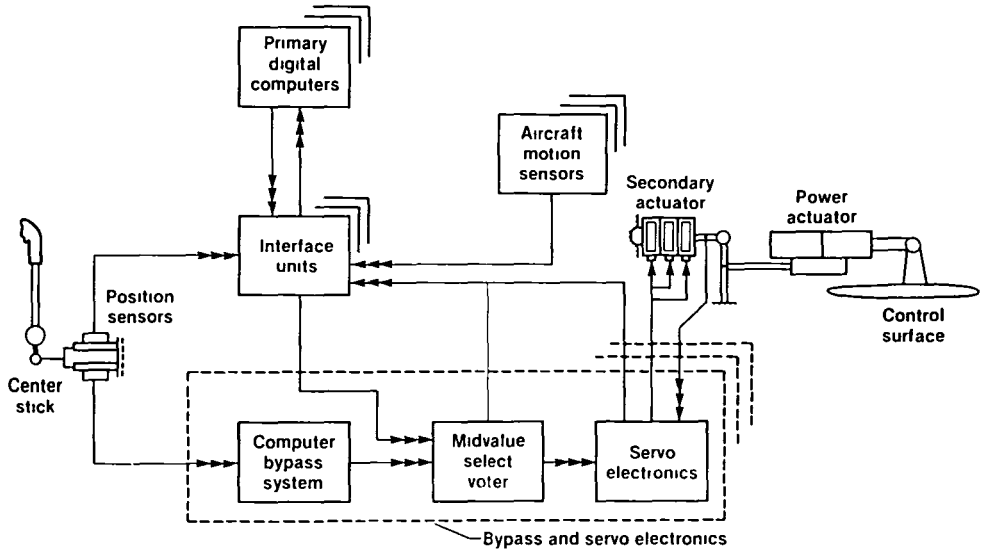


Figure 2. F-8 DFBW simplified control system schematic.

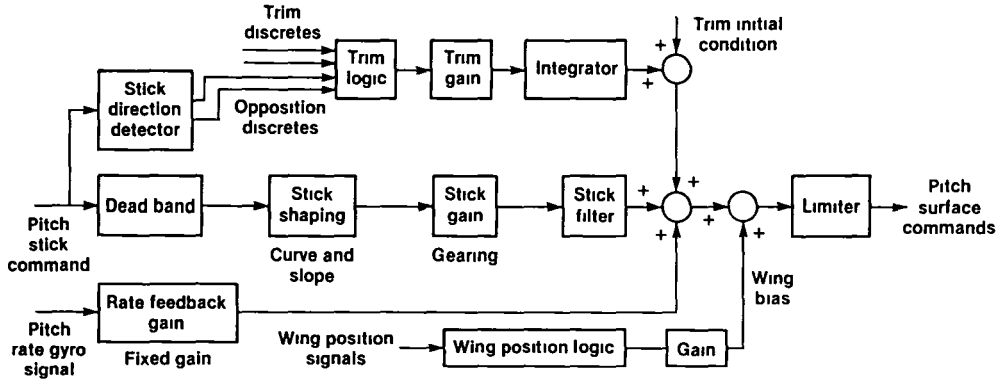


Figure 3. REBUS pitch-axis control law block diagram.

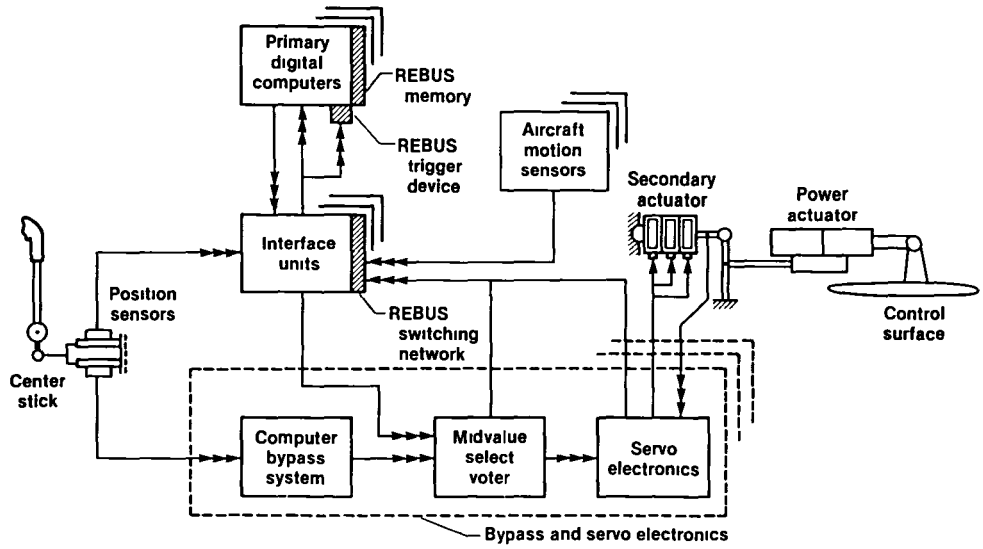


Figure 4. Modified system for REBUS implementation.

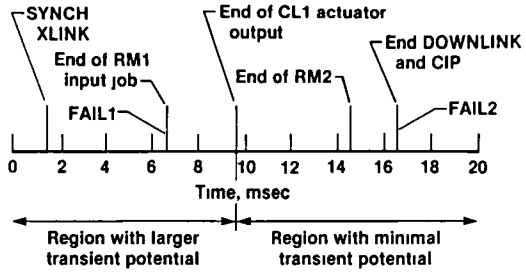


Figure 5. Failure insertion placement and its effect on transfer transient.

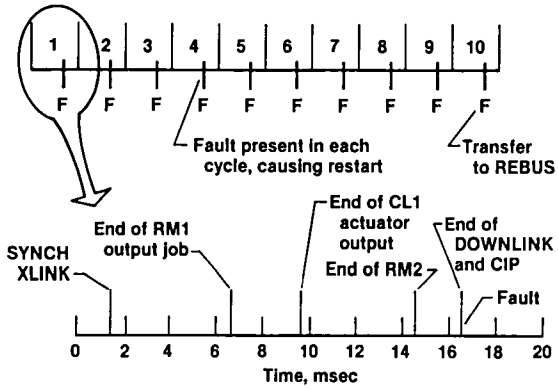


Figure 6. Timing diagram for fault insertion late in minor cycle.

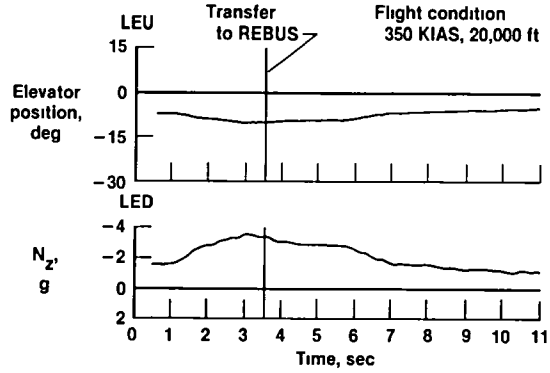


Figure 7. Flight-measured time response for elevated-g transfer to REBUS.

1 Report No	2 Government Accession No	3 Recipient's Catalog No
NASA TM-86807		

4 Title and Subtitle	5 Report Date
FLIGHT TEST OF A RESIDENT BACKUP SOFTWARE SYSTEM	January 1986
	6 Performing Organization Code

7 Author(s)	8 Performing Organization Report No
Dwain A. Deets, Wilton P. Lock, and Vincent A. Megna*	H-1338
	10 Work Unit No
	RTOP 505-66

9 Performing Organization Name and Address	11 Contract or Grant No
NASA Ames Research Center Dryden Flight Research Facility P.O. Box 273 Edwards, CA 93523-5000	
	13 Type of Report and Period Covered
12 Sponsoring Agency Name and Address	Technical Memorandum
National Aeronautics and Space Administration Washington, D.C. 20546	14 Sponsoring Agency Code

15 Supplementary Notes

Prepared for publication in AGARD-AG-289, Fault Tolerant Considerations and Methods for Guidance and Control Systems.

*Charles Stark Draper Laboratory, Cambridge, Massachusetts 02139

16 Abstract

A new fault-tolerant system software concept employing the primary digital computers as host for the backup software portion has been implemented and flight tested in the F-8 digital fly-by-wire airplane. The system was implemented in such a way that essentially no transients occurred in transferring from primary to backup software. This was accomplished without a significant increase in the complexity of the backup software. The primary digital system was frame synchronized, which provided several advantages in implementing the resident backup software system. Since the time of the flight tests, two other flight vehicle programs have made a commitment to incorporate resident backup software similar in nature to the system described in this paper.

17 Key Words (Suggested by Author(s))	18 Distribution Statement
Fault-tolerant computers Flight control Flight test Redundancy management Software	Unclassified – Unlimited STAR category 08

19 Security Classif (of this report)	20 Security Classif (of this page)	21 No of Pages	22 Price*
Unclassified	Unclassified	10	A02

For sale by the National Technical Information Service, Springfield, Virginia 22161.

End of Document