D3 N85-16892

SHUTTLE AVIONICS SOFTWARE DEVELOPMENT TRIALS, TRIBULATIONS, AND SUCCESSES: THE BACKUP FLIGHT SYSTEM

Edward S. Chevers
NASA Lyndon B. Johnson Space Center
Houston, Texas 77058

THE BACKUP FLIGHT SYSTEM REQUIREMENT

The initial design of the Orbiter flight control system was limited to the quad-redundant computer complex. Systems management and nonavionic functions were contained in a fifth computer, which was not considered flight critical. This concept was well into development when a blue ribbon panel was asked to review all aspects of the Approach and Landing Test (ALT) phase of the Space Shuttle Program to verify that the design was proper. One of the conclusions reached by the panel was that an unnecessary risk was being taken by not providing a backup flight control system for the first flight. This decision was based on the relative complexity of the computer synchronization scheme being implemented and the lack of a direct manual flight control capability.

It must be remembered that the ALT development occurred during the early seventies when microprocessors as such did not exist, only four-bit arithmetic chips were available, and software consisted mostly of punched cards implemented in batch mode on a ground computer. Of course, man had gone to the Moon in an Apollo spacecraft and used a digital computer for navigation and guidance, but there was also an analog flight control system onboard with both automatic and manual modes available. Beyond that, a manual direct mode enabled bypassing all the electronics and powering the reaction jets directly. Thus, the step forward to a total computer-controlled flight system was very significant and risky according to the blue ribbon panel. Therefore, they recommended a backup mode for ALT.

That was the background behind the decision to add the backup flight system (BFS). Initially, it was to be a very simple system installed for ALT only and deleted once confidence had been developed in the primary flight system. The word simple is very important because one of the main concerns was the NASA capability to properly verify the large software programs being developed for the Orbiter. The development of the primary flight system software is addressed in a companion paper; therefore, that area is not discussed further here.

THE BFS IN THE APPROACH AND LANDING TEST PHASE

The approach taken for the BFS was to develop a very simple and straightforward software program and then test it in every conceivable manner. The result was a program that contained approximately 12 000 full words including ground checkout and the built-in test program for the computer. The flight control portion of the software consisted of approximately 6000 words. The remainder was for the systems management functions, which still had to be performed in the fifth computer along with the backup autopilot functions. The BFS ALT configuration is shown in figure 1.

Because of the relatively simple program involved, several decisions were made that had major impact later in the Space Shuttle Program. First, Rockwell was given the contract for the BFS and the contract was an amendment to the vehicle contract. Technically, this arrangement meant that the BFS was delivered to NASA as part of the vehicle and not as an independent entity like the primary flight system software. This difference was not significant in ALT since the BFS had a unique set of hardware, the software program was small, and much of the testing was done on the vehicle. During the Orbital Flight Test (OFT) phase, when the BFS software grew to almost 100 000 words, it became very difficult to equate the primary and backup software verification efforts and to break out the BFS software as an independent deliverable product. Second, and even more significant with time, was the decision to not provide a BFS software development and verification facility at Rockwell. Here again, it was a logical and proper choice for the ALT BFS but unsatisfactory from the standpoint of long-term programmatic effects. This subject is addressed later in discussions concerning the BFS in the OFT phase of the program.

To perform verification, a series of tests was defined using the actual flight-type hardware and simulated flight conditions. Then, literally hundreds of simulated flights were flown and detailed performance analysis was conducted. The intent of most BFS tests was to demonstrate that a stable flightpath could be obtained after engagement from an anomalous initial condition. In fact, the early ground rule for BFS was to stabilize the vehicle long enough for the crew to bail out. Later, the importance of completing the mission was stressed. Since the tests did not have to run from separation to touchdown every time and the ALT flights lasted less than 30 minutes, quite often it was

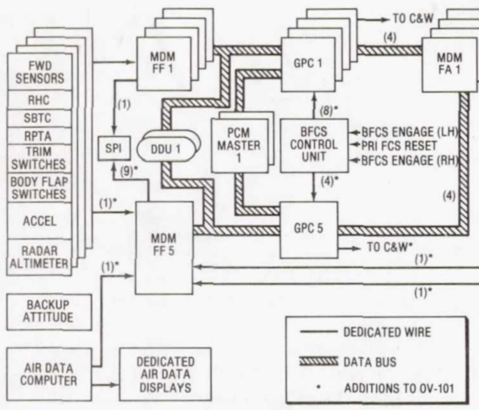

FIGURE 1.- BACKUP FLIGHT SYSTEM CONFIGURATION.

- **CERTIFICATION PLAN**
- **SCHEDULE**
- **CSDL TESTING**
  - ***STATIC***
  - ***DYNAMIC***
- **ADL TESTING**
  - **OFP CHECKOUT**
  - **SOFTWARE/HARDWARE INTEGRATION**
  - **PARALLEL SYSTEM OPERATION**
  - **SYSTEM DEMONSTRATION**
- **SAIL TESTING**
  - **SOFTWARE/HARDWARE INTEGRATION**
  - **SOFTWARE VALIDATION**
  - ***SYSTEM VERIFICATION***
- **OV-101**
  - **GROUND**
  - **TAXI**
  - **FLIGHT**

FIGURE 2.- BACKUP FLIGHT SYSTEM TESTING.

possible to run five or six tests per hour by merely changing the initial conditions. Figure 2 contains an overview of the tests performed on the BFS.

With a well-documented set of requirements, a software program that was easy to understand, a dedicated set of hardware for performing tests, and a combined Rockwell/NASA team of less than 50 compatible people, the ALT BFS could be called "the ideal program." Any proposed changes from either Rockwell or the NASA Lyndon B. Johnson Space Center (JSC) had to be justified. Changes were not allowed just because they made things easier or took less code or ran faster in the computer. A significant improvement in system capability and a requirement for the improvement had to be demonstrated before a modification was allowed.

One change that was made before the first ALT flight was the technique for acceptance testing of the software. As stated earlier, the BFS was delivered as part of the vehicle, but it was always recognized that separation of the software from the total system was desirable from a contractual standpoint. To accomplish this separation, Rockwell combined with Intermetrics in development of a FORTRAN simulation of the BFS that would run on a ground-based computer. The program emulated the AP-101 flight computer and executed the programs one minor cycle at a time. The input data being used in the emulator and output commands computed from these data were collected each minor cycle along with cycle counts, error logs, status words, and other pertinent internal computer parameters. These data were then fed into buffers in multiplexer-demultiplexer (MDM) simulators in a Shuttle Avionics Test Set (SATS), which could be connected directly to a flight computer. The flight computer then accessed the data buffers by using actual MDM address codes and did computations based on the data acquired. The general-purpose computer (GPC) outputs were collected each minor cycle and stored for comparison with the precalculated results from the Intermetrics emulation program.

Many will recognize this technique as being the equivalent of the flight equipment interface device (FEID) used with the flight computers in the JSC Software Development Laboratory (SDL). The primary difference was that the BFS technique was used for final tape verification rather than for software development. Every parameter in the emulation program and in the calculated answers was carried out to the full 32-bit word size in the ground processing computer. The flight computer results were also dumped as 32-bit words, and complete bit-for-bit comparison was made during postprocessing. Only the last 2 bits of each 32-bit word were allowed to be different before a failure was noted. This technique was the precursor to the captive simulation (CAPSIM) procedure used by the BFS for verification during the OFT flight phase.

THE BFS IN THE ORBITAL FLIGHT TEST PHASE

As the ALT phase of the program neared completion and attention was turned to the OFT phase, several conditions became apparent. First, a large portion of the primary flight system software development for ALT was unique to that portion of the Space Shuttle Program and could not be used during OFT. Second, a backup flight system was going to be required and it would have to operate during both the ascent and the descent portions of the mission. Since some abort options required a once around the Earth abort or an abort to orbit before reentry, navigation and guidance would have to be added to the BFS. Suddenly, the BFS had matured to a full guidance, navigation, and control (GN&C) system and some of the early ALT decisions became very important. As indicated by figure 3, the BFS no longer is limited to a small subset of dedicated hardware. The BFS GPC is in parallel with

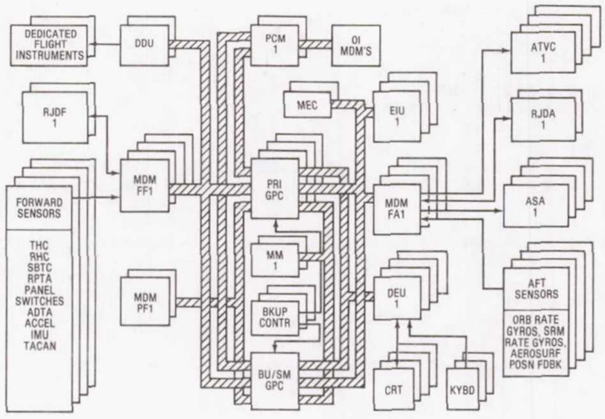

FIGURE 3.- ORBITER 102 PRIMARY/BACKUP FLIGHT SYSTEM.

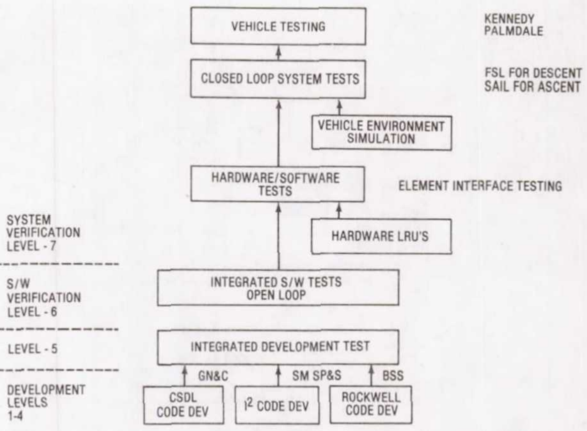
VEHICLE TESTING
KENNEDY PALMDALE
CLOSED LOOP SYSTEM TESTS
FSL FOR DESCENT SAIL FOR ASCENT
VEHICLE ENVIRONMENT SIMULATION
HARDWARE/SOFTWARE TESTS
ELEMENT INTERFACE TESTING
SYSTEM VERIFICATION LEVEL - 7
HARDWARE LRU'S
S/W VERIFICATION LEVEL - 6
INTEGRATED S/W TESTS OPEN LOOP
LEVEL - 5
INTEGRATED DEVELOPMENT TEST
GN&C
SM SP&S
BSS
DEVELOPMENT LEVELS 1-4
CSDL CODE DEV
I2 CODE DEV
ROCKWELL CODE DEV

FIGURE 4.- BFS DEVELOPMENT AND VERIFICATION TEST LEVELS.

the primary flight system computers and has full access to all sensor inputs and effector output changes after engagement.

As mentioned previously, there was an early decision by both Rockwell and NASA program management to not provide a software development facility at Rockwell for the BFS. When couched in the context of BFS being used only for ALT and never again, that decision was certainly proper. However, for OFT, there was a requirement to develop a full-blown BFS with GN&C, systems management (SM), sequencing, and display capability. Experience with the primary flight system during ALT had shown that developing the primary flight system software and the SDL simultaneously was almost an unmanageable effort. Each task was major in regard to the number of highly trained specialists necessary, and the level of management ability needed was not available at the beginning of the ALT phase.

Now, the BFS was faced with the same dilemma: how to bring together the massive resources required to develop some of the most sophisticated software ever attempted while imposing the most stringent safety and reliability specifications ever conceived. Although the decision may have been based more on ALT problems than on pure logical reasoning, it was decided that having one contractor do all programing was not the best choice. Therefore, C. S. Draper Laboratories (CSDL) was selected to program the GN&C, Intermetrics would program the SM, sequencing, uplink, and ground checkout functions, and Rockwell would program the operating system, the displays, and the overall integration. As prime contractor, Rockwell was also responsible for total system verification. The various levels of integration and verification required by the BFS are shown in figure 4. With the software tasks broken into three approximately equal sized portions, everything should have gone along in parallel until it came together again at the end in a nice, neat package. The first indication of trouble appeared when CSDL had to spend a considerable amount of time developing their best guess of how the backup system software (BSS) would work. This research was necessary since much of the GN&C is input/output (I/O) oriented and depends on the timing established by the BSS. Intermetrics developed preliminary sequencing codes but could not perform any dynamic verification because most of the sequences were dependent on guidance and navigation events for which programing had not been done by CSDL. Rockwell could not spend full time on the BSS because most of their time was spent resolving integration problems among all three sections of the software. In addition, JSC began asking Rockwell for their system integration and verification plan and the issue of software validation was drawing increased attention.

It was in this time frame that the CAPSIM technique became a predominant feature in BFS verification. The technique was similar to the emulation program used in ALT but was revised to use actual code to generate the output data for the comparison program. Precalculated inputs were used as drivers and placed in the input compools or buffer areas of the computer. The GN&C and sequencing code accessed these inputs and performed their minor cycle computations. Output results including actuator commands and display signals were then put into output compools, where the data were collected and stored on tape. All of this was done at CSDL using actual GN&C, sequencing, and, later, SM flight code, but still with the CSDL version of the operating system. The other limitation was in use of the CSDL statement level simulator (SLS), which is a non-real-time development tool and thus would not detect marginal timing conditions or dynamic interactive timing problems between different software modules. The CAPSIM data generator procedure is depicted in schematic form in figure 5.

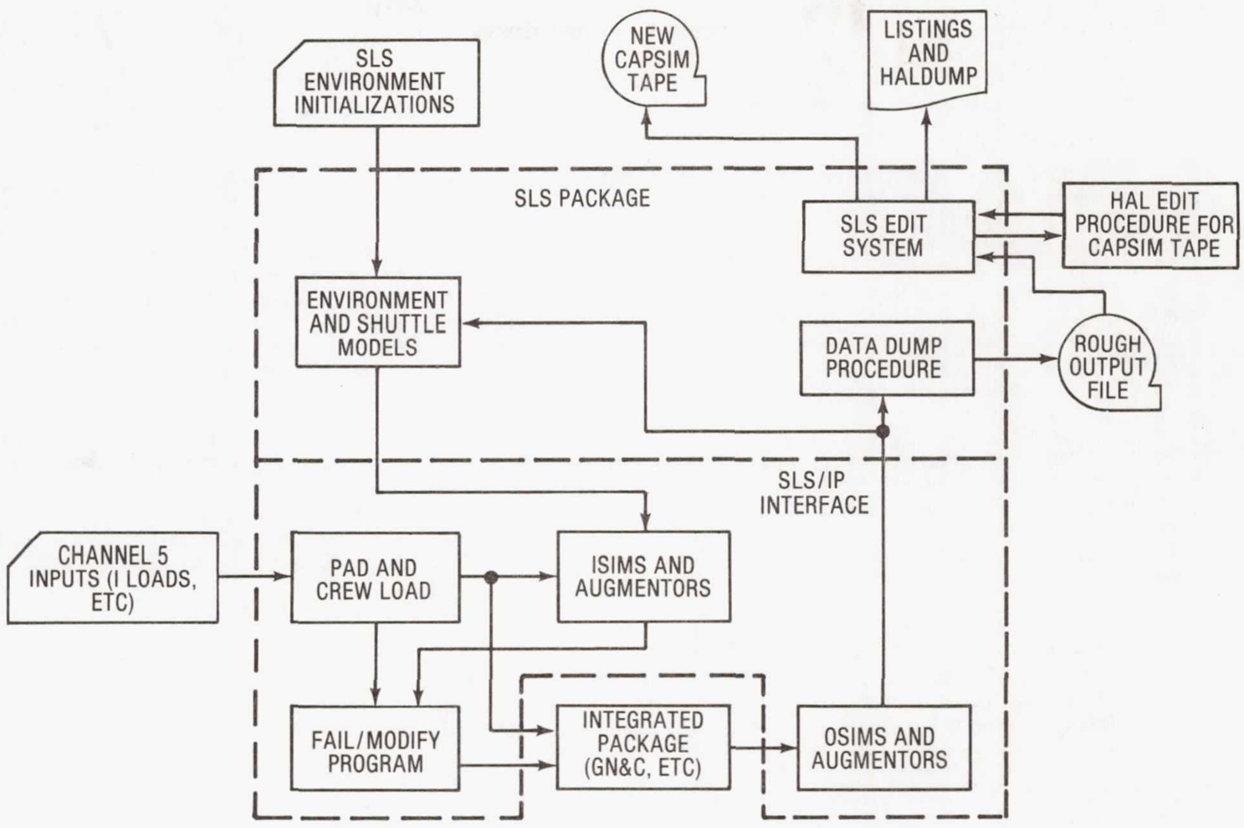

FIGURE 5.- CAPSIM DATA GENERATION.

The next step was to integrate the Draper, Intermetrics, and Rockwell portions of software into a single load tape for verification at Downey. Computer test sets and mass memory units were not available for general-purpose use; therefore, a test set tape was made that could be loaded into the flight computer from a SATS unit. The tape format was tailored to the requirements of the specific computer in the SATS; therefore, the test set tape was different from the final mass memory load tape which would be required at JSC. With the flight computer loaded and cabled to the SATS, the operational BFS was ready for the simulated input data from the CAPSIM program, and, as anticipated, operation was not smooth initially. The evaluation procedure involved collecting the output buffer data each minor cycle from the flight computer and then comparing the tabulated data with equivalent data sets from the CSDL tests.

Approximately 25 test points in ascent and 20 in descent were selected for comparison. Continuous comparison was not possible at first because no automated plotting programs were available. These programs were developed after 6 to 8 months of the effort and later included dual comparison on the same plot. If all of this history seems archaic, remember that the BFS started from scratch and had to compete with the primary system for resources and time on any existing facilities. During the first 6 months of CAPSIM testing, the BFS was a virtual basket case. When output data were compared, there were times when it was impossible to determine whether they were from the same program. One of the main problems consisted of the initial condition values used at CSDL and Rockwell. The guidance and navigation programs were very sensitive to the data-base parameters, and as many as five or six different versions could exist at any given time in the various JSC, CSDL, and Rockwell facilities. The problem is not major in terms of gross performance of the flight system, but when two different facility tests were overlayed and every discrepancy had to be accounted for, a considerable amount of time was wasted in trying to explain away what could be a minor difference. Of course, in the beginning, no one was sure whether a minor difference was insignificant or an indication of a potential major software coding problem.

With time, as everyone began to get a better feeling for the differences in output results and more of the process became automated, dual comparisons between the BFS and the primary flight system also were begun. One of the cost-saving features of BFS verification was the so-called "piggyback" mode of operation, in which the BFS was to use all applicable data from the primary flight system and eliminate duplication. The results of one test case showed very close correlation between the primary and the backup systems and both were found to be wrong. The problem would not have had a major impact on the flight but did serve to show the danger in not performing total independent analysis. Afterwards, the BFS people were careful to spot check more test cases and to perform some independent analysis.

Although the CAPSIM technique was the primary verification tool for the BFS, it did have one serious shortcoming. The test case scenarios were from a cataloged set of runs at CSDL which had been based on primary flight system testing. Again, this procedure was in accordance with direction to control development cost and maximize use of existing primary flight system information. The concern that arose was based on the fact that the BFS would only be engaged after a massive primary flight system failure. Therefore, the initial conditions could quite possibly be at the limits of the normal primary GN&C boundary conditions and very few BFS test runs were made from these very abnormal conditions. Therefore, approximately 100 additional "stress cases" were developed to be run in the Flight Simulation Laboratory (FSL) in Downey and the Shuttle Avionics Integration Laboratory (SAIL) at JSC. By combining multiple objectives into one test run, the final number of individual test runs was refined to 37. Every one of these tests was unique and required special downlist data formats for additional visibility, and all data analysis was manual. Since these tests were to be performed one time as a final safety verification check before the first OFT flight, automated data processing could not be justified. All but a few of these tests were performed successfully, and failure of the few was due to procedural errors or to a misunderstanding of the intent of the stress condition. However, adequate data were collected to give the Shuttle community a high level of confidence in the capability of the BFS to meet its intended objectives for the first orbital flight.

Some of this confidence waned on launch day, when the BFS failed to synchronize with the primary flight system a few hours before lift-off. However, the shock to the BFS developers quickly disappeared when it was determined that the problem was due to a timing error in the primary system and that, in fact, the BFS recognized the problem and was trying to inform the launch team that an anomaly existed. When the problem was identified and fully analyzed, it was shown that the mission could have been flown successfully even with the interface timing error. The BFS would have bypassed the data being sent in the wrong time slot and corrected itself after engagement had that been necessary. However, those facts were not available for several days after the launch. A modification was made, the flight was highly successful, and considerable relief was expressed by all involved. The primary flight system people said they knew all along the BFS would not be needed but were glad it was there. The BFS people said they knew they were ready to take over and save the mission but were glad they did not have to do it.

CONCLUDING REMARKS

The BFS has never been required to demonstrate its capability. A proposed on-orbit engagement and orbital maneuvering system (OMS) engine burn has now been deleted from the list of flight test requirements. This change is partly the result of a very full and busy flight plan and partly of the high confidence level which exists in the capability of the BFS to perform when required. Although most of the people who worked on the backup flight system would like to see it used, I consider it a supreme compliment from the Program Office and the Flight Directors to accept the system on the faith of the development and verification program. This acceptance is a tribute to the people and organizations who put so much of themselves into the project for many years.

SUMMARY

In retrospect, several factors or lessons learned in the BFS evolution stand out prominently. Neither the NASA nor anyone else should ever attempt to develop a software program approaching 100 000 words in size without having the development facilities in place, especially when the requirement is for zero-defect software code. In addition to the basic facility, the software analysis and support tools should be available throughout the project, not near the end. It will always be difficult to convince a program manager that he should put a sizable amount of his money into a facility initially for the purpose of saving money several years downstream. But no NASA or contractor program manager should again have to experience the trials and tribulations that existed for the primary flight system in ALT and the backup flight system in OFT. In the end, the only glory is in success.

FLIGHT SOFTWARE FAULT TOLERANCE VIA THE BACKUP FLIGHT SYSTEM

Terry D. Humphrey and Charles R. Price
NASA Lyndon B. Johnson Space Center
Houston, Texas 77058

ABSTRACT

A generic software error in the quadruply redundant primary flight system could result in the catastrophic loss of Space Shuttle vehicle control in the hostile environment of ascent or reentry. The Space Shuttle backup flight system was designed to protect the crew and vehicle in this eventuality. The significant challenges met in the design and development of this state-of-the-art protective system is the subject of this paper.

INTRODUCTION

Implementation of the backup flight system (BFS) for the Space Shuttle is a major advance in state-of-the-art fault-tolerant software in general and in Space Shuttle fault-tolerant flight software in particular. The BFS was chartered to protect against a software fault in the most sophisticated flight software system ever implemented: the Space Shuttle primary flight software (PFS). The PFS is designed to operate a redundant set of general-purpose computers (GPC) to control an innovative, multiple-element, reusable, manned spacecraft through an ensemble of flight regimes never before encountered by a single vehicle. For STS-1, the PFS consisted of more than 400 000 computer words of flight code, a complete test of every possible combination of branch instruction or decision point of which would take more than 10 000 years of computer time on today's fastest computers.

To protect against a latent programing error (software fault) existing in an untried branch combination that would render the Space Shuttle out of control in a critical flight phase, the BFS was chartered to provide a safety alternative. The BFS is designed to operate in critical flight phases (ascent and descent) by monitoring the activities of the Space Shuttle flight subsystems that are under control of the PFS (e.g., navigation, crew interface, propulsion), then, upon manual command by the flightcrew, to assume control of the Space Shuttle and deliver it to a noncritical flight condition (safe orbit or touchdown). Many technical, managerial, and operational challenges were experienced by the development team of NASA and its contractors in bringing the BFS from a concept to a working operational system. The challenges addressed herein are those associated with the selection of the PFS/BFS system architecture, the internal BFS architecture, the fault-tolerant software mechanisms, and the long-term BFS utility.

CHALLENGE 1: A MANAGEABLE SYSTEM ARCHITECTURE

Central to any concept of a reusable spacecraft is the theme of higher system reliability through redundancy of finite-lived components. In the Space Shuttle avionics, the availability of a properly functioning flight computer is assured through the wiring of five identical GPC's in parallel. Since the memory available for state-of-the-art GPC's at the beginning of the Space Shuttle development cycle was much less than the flight software requirements dictated, an early partitioning scheme was established. For each of three flight phases (ascent, on-orbit, descent), a redundant copy of all critical functions was loaded in each of four GPC's and the fifth GPC was loaded with a complement of useful functions the loss of which could be tolerated. This strategy assured protection from multiple, sequential computer hardware failures, but did not address the possibility of a software fault generic to the set of four redundantly programed GPC's causing loss of control of the Space Shuttle. Concern for such a software fault is valid in that regardless of the number of checks and balances that are put in place to find and eliminate specification and coding errors in major software developments, there can be no 100-percent assurance that latent, potentially dangerous software errors do not exist in the delivered product.

The obvious strategy for increasing the software reliability of the Space Shuttle was through software redundancy, but the challenge of the problem was the form of the redundancy to implement within time and cost constraints. Three redundancy alternatives were available: (1) increasing the internal PFS redundancy; (2) duplicating the PFS in another software version programed by a different set of programers completely isolated from the PFS programer, or (3) implementing a reduced-capability backup system by a semi-isolated set of programers. The first alternative was not pursued because it was felt that every practical internal measure was already being pursued by the PFS designers. The second alternative was considered too costly and fraught with duplication of functions not essential for a secondary system. The third alternative was selected since it afforded an additional measure of protection that was achievable within cost and schedule constraints. The reduced capability was set by a single memory load for both ascent and descent in a single GPC. The BFS also assumed the non-flight-critical functions that had been scheduled for the nonredundant fifth GPC for ascent and descent. The semi-isolation of the programers was achieved by having the BFS programed by a contractor geographically separated from the PFS contractor.

THE BFS FAULT-TOLERANT ARCHITECTURE

Significant challenges were faced by the designers to develop a BFS which would closely track the PFS, protect itself and the PFS from data pollution from each other, and also be ready at any point in the ascent, abort, or descent profile to take over control of the vehicle safely when manually engaged by the crew. To provide this capability, a technique had to be developed that would provide for tight synchronization of the BFS to the PFS in order to prevent divergence, but that would also protect both from inducing faults in the other. A more pollution-protective technique than that used for PFS redundant-set synchronization had to be developed. To protect the PFS from faulty BFS data or timing, this technique would permit no transfer of data from the BFS to the PFS.

An innovative technique for synchronization of the BFS and the PFS was developed using flight-critical data bus input profile tracking of the PFS that involves use of the input/output processor (IOP) input data bus listen capability and the transfer of input profile and minor cycle data from the PFS to the BFS. The BFS protects itself from pollution by erroneous input profile data by voting on the redundant data sent to it by the individual PFS GPC's. In addition, the BFS protects itself from input profile timing faults of the PFS by the use of its own data bus timing window thresholds for each of the individual groups of input profile data.

To protect the BFS and PFS from pollution by erroneous data received from one another, an interface design policy was established which allowed no transfer of software-generated data from the BFS to the PFS but did allow data absolutely essential to the proper tracking of the PFS to be transferred and used by the BFS. The absence of data transfer to the PFS prevents any pollution of the redundant PFS by the BFS. The BFS was designed to protect itself from pollution from erroneous PFS data first, by being limited to the use of a small amount of absolutely essential transfer data; second, by performing a vote on the redundant sets of data obtained from the redundant PFS GPC's; and third, by performing reasonableness checks on the voted data.

Another challenge faced in the development of the fault-tolerant BFS was the design of a safe method of taking control of the vehicle at any point in the flight profile without inducing control effector transients which might endanger the crew and the vehicle. The design developed to provide this protection required the input of engage initialization data from the subsystems via the flight-critical data buses immediately after BFS engagement. These data were then used to ensure that subsequent BFS control commands did not overstress or generate significant transients on the control effector subsystem.

One of the foremost innovative techniques used in the BFS fault-tolerant design was developed to provide for recovery from the loss of PFS-generated master events controller (MEC) sequencing as well as for attempting recovery from BFS GPC hardware or software errors. The loss of MEC sequencing commands might occur either as a result of a generic PFS software failure or as a result of the abrupt termination of all PFS-controlled flight-critical data buses at BFS engagement. Recovery from these types of errors is provided by the BFS software restart technique. A software restart is initiated upon BFS engagement, and, in the event that critical MEC command sequences are found not to have been properly performed, the BFS reinitiates the full MEC sequence of commands for the appropriate mission event.

The use of the restart technique to attempt recovery from BFS GPC hardware or software errors was developed to protect the BFS from transient errors and, in the case of hard errors, to continue attempts at recovery in hopes that the error will not persist. This restart recovery involves reinitialization of input/output (I/O) and restarting of BFS application processing at the beginning of a new GPC major processing cycle. The restart recovery technique provides this protection for both the preengaged and engaged modes of BFS operation.

A MOVING TARGET: MAINTAINING TRACK OF SOFTWARE CHANGES

Unlike the Approach and Landing Test (ALT) PFS, the BFS for ALT could not be used as a base upon which to build Orbital Flight Test (OFT) software. The BFS software for ALT was designed and developed by Charles Stark Draper Laboratories and provided backup for flight control functions only, provided no CRT/crew interface, and provided only a very minimal task-list-type executive. Rockwell was selected to develop the BFS for OFT and essentially started anew about 2 years behind the PFS software development. A new operating system had to be designed and developed, all existing PFS requirements and change requests (CR's) had to be reviewed for applicability to the BFS, and an overall BFS software design had to be developed. The BFS not only had to catch up with the PFS level of maturity very quickly, but then had to maintain pace with a very large amount of PFS requirements development and software change activity. A significant amount of effort and manpower was required to accomplish this goal.

POST-OFT UTILITY OF THE BACKUP FLIGHT SYSTEM

The BFS was initially envisioned to be used only through the Shuttle OFT flights. The expectation was that after OFT, the entire Shuttle system design, including PFS software, would be proven safe for operational use and, therefore, the BFS would no longer be needed. Close to the end of OFT, an examination was undertaken to assess the need for continuing the use of the BFS. Assessments of the PFS software discrepancy report (DR) traffic showed it to correlate proportionally to the level of PFS software change traffic but, even in cases in which software change traffic was small, the number of DR's appeared to decay exponentially rather than to drop abruptly. These data indicated that latent software errors had high levels of persistence. This information was used in conjunction with the projections of PFS software change traffic for future flights. It was determined that for a significant time in the future, the PFS software change traffic would continue to remain at significant levels and therefore the risks would remain high for latent PFS software errors. Therefore, it was concluded that for at least a significant time in the future, the BFS would be needed to protect against generic PFS software faults.

BIBLIOGRAPHY

Rockwell International Specification: Backup Flight System Program Requirements Document Overview. Rockwell International MG038100, 1979.

Rockwell International Specification: Primary Avionics Software System (PASS) and Backup Flight System (BFS) Software Interface. Rockwell International ICD-3-0068-03, 1981.

Rockwell International Specification: Backup System Services Program Requirements Document. Rockwell International MG038101, 1982.