```
        z6.d    ]

SUMMER [ z6.d , z4.d ; ;
                              z7.d    ]

HPF [ z7.d ;
        0.1 ,
        5.0  ;
        u.dot.comp  ]

COMP [ u.dot.avrg , u.dot.comp ;
                    0.926   ;
              flag.acc.dif    ]
```

When Fig. 1 is examined, it is seen that the specification is written by simply replacing the blocks with their keywords and their input-output signals. These signals are "declared" at the top of the specification. Assertions have also been introduced at various points that use the '/$' facility; for example, the variable 'a4' is declared to be less than 488.3.

Conclusions

In this paper, the problem of automatically generating simplified models or benchmarks of DFCS modules has been addressed. A specification language has been proposed which makes it easy for the system designer to describe flight software. This language specifies the DFCS in a form that is machine translatable. It also embodies constructs that enable the designer to express facts about system performance that can later be employed to set up executable assertions for software verification. The language can be readily updated to express new features of DFCS by the addition of keywords. The salient features that the language translators should have are described and the language is applied to a typical DFCS module.

References

1) Ludewig, Jochen, "ESPRESO: A System for Process Control Software Specification," IEEE Trans. on Software Engg., Vol. SE-9, July 1983, pp. 427-436.

2) Beichter, Friedrich W., Herzog, Otthein, Petzsch, Heiko, "SLAN 4, Software Specification and Design Language," IEEE Trans. on Software Engg., Vol. SE-10, March 1984, pp. 155-161.

3) Rajan, N., de Feo, P. V., Saito, J., "Stress Testing of Digital Flight Control System Software," IEEE/AIAA 5th Digital Avionics Systems Conference, Seattle, Wash., Nov. 1983.

4) Andrews, Dorothy M. and Benson, Jeoffrey P., "An Automated Program Testing Methodology and Its Implementation." Proceedings, 5th International Conference on Software Engineering, 1981, pp. 254-261. Also reprinted in "Tutorial: Software Testing and Validation Techniques," 2nd edition, 1981, pp. 349-356.

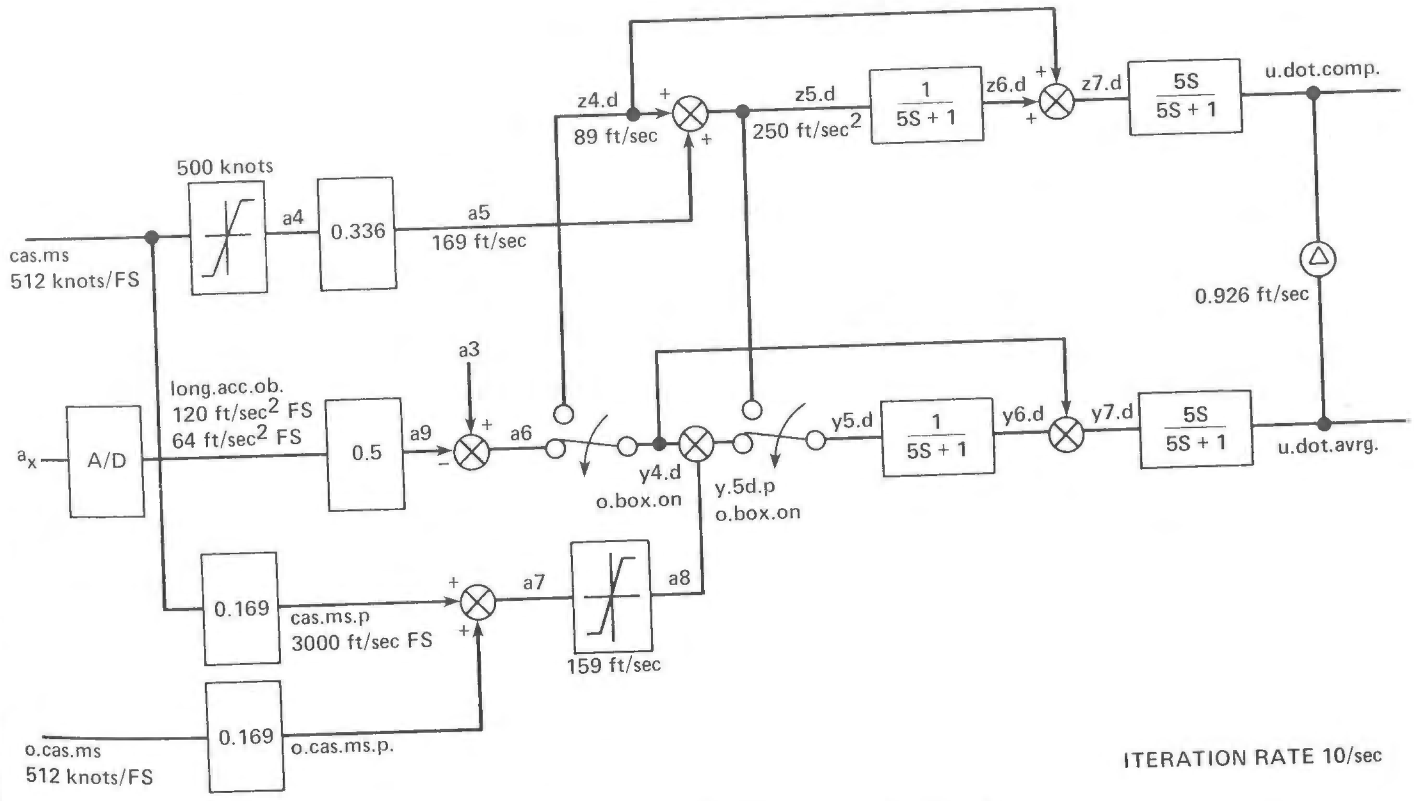
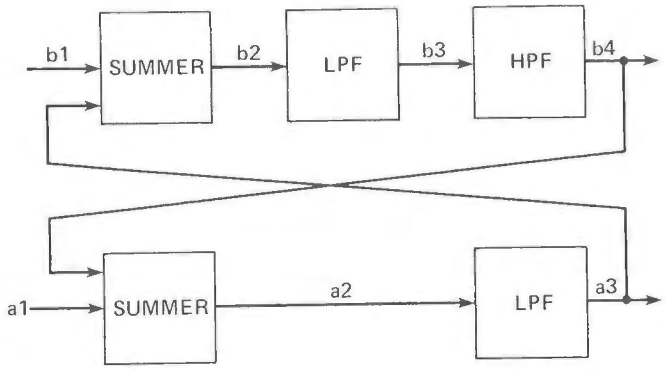
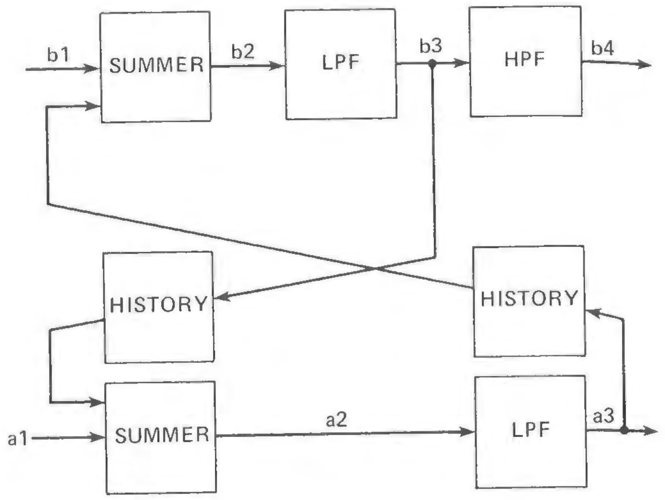


Fig. 1 Block diagram for the speed-comp module.

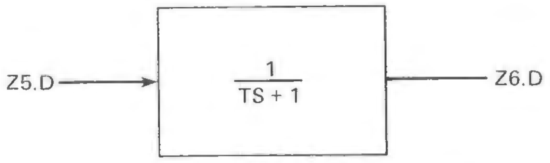




a) Without history keywords.

b) With history keywords introduced.

Fig. 2 Cross connection of keywords.



LOW PASS FILTER REPRESENTATION

DIFFERENCE EQN.:

$$\text{PRESENT OUTPUT} = \frac{\text{TIME-CONSTANT} - \text{SAMPLING TIME/2}}{\text{TIME-CONSTANT} + \text{SAMPLING TIME/2}} * \text{PAST OUTPUT}$$

$$+ \frac{\text{SAMPLING TIME/2}}{\text{TIME-CONSTANT} + \text{SAMPLING TIME/2}} * (\text{PRESENT INPUT} + \text{OLD INPUT})$$

Fig. 3 Representation of a low-pass filter.

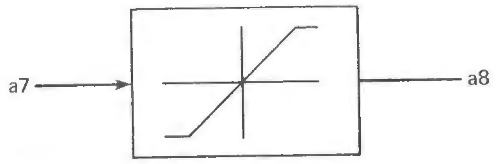


Fig. 4 Representation of a limiter.

AUTOMATED SOFTWARE TEST SYSTEM FOR THE 737-300 FLIGHT MANAGEMENT COMPUTER

84-2665

Steven C. Runo*

Senior Engineer
The Boeing Commercial Airplane Company
Seattle, Washington

Abstract

Traditional manual approaches to software validation could not produce the required testing rigor within the inflexible 737-300 Flight Management Computer System program schedule. In response, an integrated and highly automated validation test system was developed based on experience gained from previous validation efforts. The "user-based" system consists of three basic elements: 1) a specialized simulation for generating expected results, 2) an automated test bench that interrogates the operational flight program "in situ", and 3) a program to document and compare test results to expected results. The entire procedure is automated from test case design through final analysis of the test results. The system has proved to be an efficient and rigorous validation of the flight software within a tight time schedule and limited budget. Further, the user is released from tedious laboratory testing and allowed to concentrate on test analysis.

Introduction

In recent years, automated software testing methods have been suggested as a means to provide rigorous verification and validation of operational flight programs without the drain on time and manpower that manual techniques normally require. Suggested methods include: specialized simulations, theoretical "best" approaches to test design, result comparators and automated test benches. However, in many cases these automated methods reduce the effort required in a specific area only to increase the overall testing effort by expanding the number of test conditions measured or the required degree of analysis.

The time and budget constraints of the 737-300 Flight Management Computer System (FMCS) project would not permit manual testing methods or inefficient automated methods. For example, approximately one third of the validation program for the 737-300 Flight Management Computer was testing of the performance functions. This was initially expected to require at least 10,000 separate test points and hundreds of laboratory test hours for each new software release. As a result, validation of the performance functions was considered "risky" and it was determined that the required testing could only be accomplished with an integrated, automated system for preparing expected results, conducting the laboratory testing and finally, analyzing the results.

The system that was developed is largely the result of several years of experience in similar testing efforts. Appropriately therefore, this paper begins with a discussion of the previous Boeing Co. validation programs that significantly shaped the 737-300 Flight Management Computer System validation effort. Next is a detailed discussion of the specific elements of the performance function validation testing, including: the plan of test, the simulation software for generating expected results, the automated laboratory test system and the methods used to compare and archive the test results. The paper concludes with a brief description of the experience-to-date utilizing this approach.

History

Performance Data Computer System (PDCS). The Performance Data Computer was originally developed in the mid-seventies in response to rapidly increasing fuel costs. The system was designed by Boeing and the hardware / software vendor, Lear Siegler, Inc. of Grand Rapids, Michigan, to optimize the performance of 727 and 737 aircraft via a mixture of stored and computed speed schedules and throttle setting targets based on current flight conditions.

Validation testing of the PDCS was one of the first efforts of its type and thus has played a significant role in shaping the development of later validation efforts. The earliest test cases were designed to test each of the functions of the PDCS at conditions that were likely to be encountered in normal

*Member AIAA

operations. All early testing of the PDCS was accomplished via manual entries through the Cockpit Display Unit (CDU) keypad or through test bench discrete switches and variable potentiometers. Results were manually recorded from the CDU responses of the PDCS. This test approach was highly time-consuming and thus, necessarily limited the scope and detail of practical testing to about 1000 total test points.

Late in the PDCS development program an automated test bench was developed. The automated test system would enter Cockpit Display Unit and aircraft system inputs, wait an appropriate length of time, read the CDU responses and compare the test results to pre-stored expected results. Because this system was somewhat inflexible and required a great deal of pre-test effort to convert the manual test cases to the automated format, it was primarily used to test only the uniformly-formatted propulsion test cases. However, the system did provide valuable experience for development of the 737-300 test system.

Performance Navigation Computer System (PNCS). The Performance Navigation Computer System was developed as one of the earliest "flight management"-type systems. The original intent was to integrate PDCS performance information with navigation and guidance capability in a single computer for 737-200 aircraft. Limited sales interest and technical problems forced cancellation of PNCS development prior to the anticipated system certification. Again, the accumulated experience would prove to be valuable.

This system introduced the complexity of navigation and guidance computations overlayed on the basic performance information. Using a "flight plan buffer dump" developed by Lear Siegler for the PNCS program, much of the performance information could be captured at each waypoint in the predicted flight plan. The data could then be analyzed to determine if aircraft performance was being computed correctly. However, if an error was found, it was often difficult to trace it to its origin because the flight plan buffer dump could not include all of the performance variables and their intermediate values. The PNCS performance plan of test did not actually increase the total number of test points acquired, partly because of confidence in the previous Performance Data Computer aircraft and engine models and partly because the higher-order functions of the PNCS required significantly greater test and analysis time. The experience of the PNCS test program suggested an entirely new approach to validation testing of flight computer software was required.

757/767 Flight Management System (FMS). When the new generation airliner programs were launched in the late seventies, it was recognized that testing of the 757 and 767 would be the most rigorous ever attempted. This especially applied to the new "glass" cockpits and flight management systems. In response, major projects were undertaken to develop automated testing techniques for validating the performance functions of the 757/767 Flight Management System. The first major project was the creation of a series of programs to generate flight management system expected results. These series of programs became the Boeing Standard Programs (BSP) and are detailed below. Other test tool projects included the development of an automated test bench and a test report comparator program. The latter is also detailed in the discussion below.

The Performance Algorithm Test System (PATS) was designed to set test conditions and extract results from the operational flight program (OFP). The OFP source code is first prepared by adding input and output routines to translate between the unique FMS software structure and the PATS driver. This modified code is then loaded and executed in the Flight Management Computer hardware according to commands given in the PATS driver file. Test results are recorded in a file formatted identically to the expected results generated by the Boeing Standard Programs simulation software. The 757/767 work was a major influence on the development of the 737-300 Flight Management Computer performance function test program and many of the features of the system described below were originally developed for the 757/767 FMS test program.

737-300 Validation Testing

Development of the 737-300 Flight Management Computer performance function test program began with the design of an overall plan of test and the development of requirements for the individual elements of the test system. The three major elements required under the plan of test were: 1) the Boeing Standard Programs (BSP) expected results generator, 2) the Performance Validation Test System (PVTS) automated test bench and 3) the COMPEX test results comparator / report generator. The BSP simulation software would require major revisions to reflect unique 737-300 Flight Management Computer design requirements and the 737-300 airframe / engine combination. The Performance Validation Test System would have to be developed from the "ground up" to interact with the new flight computer hardware. However, the COMPEX report generator developed as part of the 757/767 BSP project would only require user experience.

Performance Plan of Test. The performance plan of test was developed given two major considerations: 1) the experience gained in the PDCS, PNCS and 757/767 FMS test programs, and 2) the timing of required software deliveries from Lear Siegler. The plan of test calls for a "bottom-up" approach starting with module-level validation of the aerodynamic, propulsion and atmospheric data bases. The "flight phase" testing level is designed to validate the integration of data basc information. The final "mission"-level testing is directed at validating the the complete performance function system. Each subsequent level of testing is dependent on the successful validation of the lower levels of testing. This scheme not only simplifies pinpointing errors, but also blends well with 737-300 program schedules that called for the delivery of increasingly capable software in distinct packages. The plan of test is depicted graphically in Figure 1.

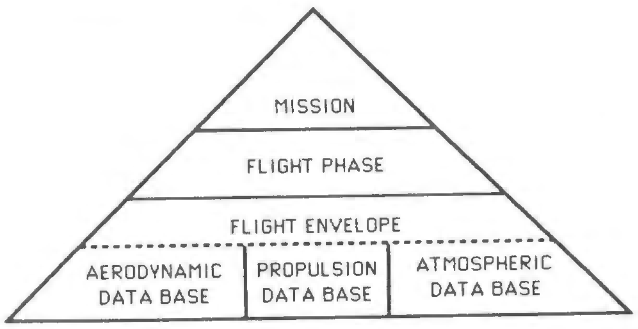


Figure 1 Performance Plan of Test

The data base level is designed to check the individual software modules by varying each of the input parameters over, and outside, the full range of possible values, with particular attention to regions where errors are likely to occur. Testing tolerances are limited to computer "round-off" errors since the polynomial nature of the data bases eliminates inexact interpolated or approximate extractions. This level also verifies the correct computation of the complete flight envelope defined by the combination of aerodynamic, propulsion and atmospheric data bases.

The flight phase level of testing examines the computation of each separate mode of flight, e.g., maximum gradient climb, long range cruise, economy descent). Intermediate results are compared at each performance integration step, rather than simply at each navigation-defined waypoint. Further, the values of intermediate parameters, such as thrust and drag, are checked against expected results derived from the Boeing Standard Programs. Tolerances at this level are a combination of the database and FMCS requirements-specified tolerances.

The mission level of performance function testing is designed to validate the complete integration of a flight plan or "mission". This "top-level" testing also includes examination of performance function interactions with the other Flight Management System components. However, the scope of these tests are limited by separately defined "flight scenario" tests that are designed to check the complete range of Flight Management System component interactions. The intermediate results examined in the flight phase tests are also available on this level, but testing tolerances are now based only on tolerances specified in FMCS requirements.

Boeing Standard Programs. The Boeing Standard Programs (BSPs) are a set of mainframe computer programs originally developed to validate the performance management functions of the 757/767 Flight Management Computer (FMC) system. The BSPs consist of nine separate, but inter-related programs that generate expected results in a format permitting automated comparsion with test results. Previous Boeing performance programs were generalized to support a wide variety of uses and not specifically tailored to the requirements of flight management software validation. The programs emulate the nine functions shown in Table 1.

PROGRAM	DESCRIPTION
A523 AEROEX	Aerodynamic data base
A524 PROPEX	Propulsion data base
A525 ATMOSX	Atmosphere and wind prediction
A526 FNLIMX	Speed envelope calculation
A527 ALTEX	Altitude limits calculation
A528 SPEEDX	Speed generation
A529 LEGEX	Leg integration calculation
A530 STEPEX	Step-cruise optimization
A531 BSPEX	Full flight path trajectory

Table 1 Boeing Standard Programs

The BSPs are designed to share common modules with each other. This is best illustrated in Figure 2. The solid-lined boxes represent a related set of routines; the solid-lined boxes inside the larger dash-lined box (e.g. speed generators, leg integrators, etc.) are termed "functional" modules and represent sets of routines used by more than one BSP. The solid-lined boxes outside the dash-lined box are termed "driver" modules and represent routines which are used by only one BSP (e.g. SPEEDX, LEGEX, etc.). Arrows point from calling routines to the routines called.

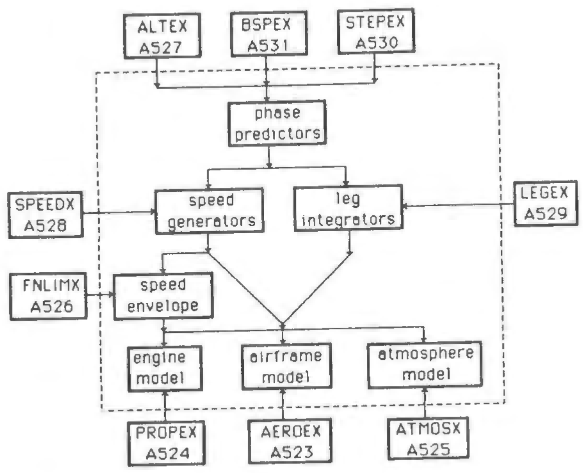


Figure 2 BSP System Architecture

Execution of the BSPs requires files containing the necessary aerodynamic and propulsion coefficients, and a performance input file that selects the particular test conditions and functions to be executed. As flight testing of the 737-300 updated the available performance data, this new information could be periodically added to the propulsion and aerodynamic data bases by editing the coefficient files. Updated expected results could then be generated without redesigning the test cases or recoding the BSP routines.

For the 737-300 FMCS performance validation effort, the BSPs required a new engine model, an enhanced aerodynamic model, polynomial speed generators and additional logic in the phase predictors, and leg integrators for the specific requirements of the 737-300 FMCS. Since each version of the BSPs is based on a single engine model, a new BSP version was created for the 737-300 FMCS program. However, the other changes were added to the existing BSPs as selectable options available to all versions of the BSPs. For example, all versions of the BSPs currently allow the user to select speed generation based on 757/767 table look-up methods, 737-300 polynomial equations or generalized computational methods.

Despite these relatively major changes, the BSPs were ready for use in a short period of time because of the extensive 757/767 development work that had already been completed and validated. In addition, a generalized version of the BSPs was also available that premitted preliminary generation of expected results and even development of the performance data base equations.

<u>Performance Validation Test System</u>. The Performance Validation Test System (PVTS) is an automated test bench designed to accept Boeing Standard Program format inputs, execute the corresponding software modules in the Flight Management Computer, and record the results in a format suitable for automated comparsion with BSP-derived expected results. The system is unique in several respects. First, the vendor-supplied operational flight program (OFP) is interrogated without modification while it is resident in the vendor-delivered hardware. This is significant because it ensures testing of the flight software in nearly actual operational form. Second, PVTS utilizes the same inputs to drive the execution of the OFP that were used to execute the BSP simulation software. This allows the test engineer to rapidly redesign a given test case to meet changing requirements and obtain both the test results and expected results without having to formulate a separate, and possibly different, test condition file. Finally, PVTS allows testing of a single function or a complete series of functions without user interface except at startup. This feature relieves the tester from monotonous and error-prone manual testing, while producing repeatable results much faster than manually possible.

The PVTS hardware consists of a VAX 11/780 computer connected to a Lear Siegler, Inc.-supplied Computer Control Unit (CCU) via an eight-bit parallel bus. The Lear Siegler-developed interface system allows the VAX to set and examine variables internal to the Flight Management Computer and execute the operational flight program (OFP) from breakpoint to breakpoint. Lear Siegler "symbolic debug" software looks up the memory addresses of variables allowing the variables to be accessed by name. The system is shown in Figure 3.

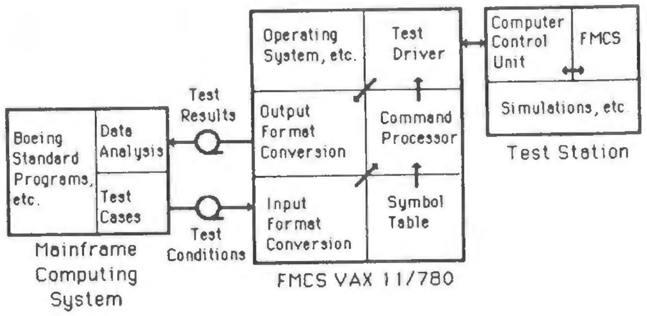


Figure 3 PVTS Overall System Diagram

Actual operation of the Performance Validation Test System is in two steps. The first stage reads the Boeing Standard Program-formatted input files and produces a command file that describes the test conditions in a language familiar to the second stage. A PVTS data base maintains information on the name and numerical conversions required to translate the inputs. This first step is typically accomplished off the test bench prior to the actual test session.

The second PVTS stage is executed on one of two 737-300 FMCS test benches using the previously-generated command file. Computer Control Unit (CCU) commands set the appropriate program counters, breakpoints and test conditions in the Flight Management Computer for each test procedure. The OFP is then executed and the test results recorded. The memory locations of FMC variables and modules are determined from the Lear Siegler symbol table routines mentioned previously.

The development of the Performance Validation Test System (PVTS) was in three phases to support the three phases of the performance function plan of test. Phase I development was designed to execute individual data base modules and therefore is specialized for each test procedure that tests a separate operational flight program (OFP) computation. Phase II of the PVTS development concentrated on the the flight phase test cases where a single module calls the execution of several other modules. This required additional PVTS logic to capture and interpret the integration-by-integration results of the OFP. The final phase of PVTS work required a separate Cockpit Display Unit (CDU) keypush driver. The keypush driver enters the required flight plan information via the CDU and the PVTS captures the mission information as it is built in the OFP.

Comparsion of Test and Expected Results. After the test results are obtained from the Performance Validation Test System, they are transferred via tape back to the mainframe computing system where the expected results are stored after being generated by the Boeing Standard Programs. The test results are compared to the BSP expected results by COMPEX. COMPEX is a point-by-point comparator and report writer program developed as part of the 757/767 BSP project. On a database level, the comparsions are one-to-one and required to be exact (allowing only for computer round-off errors). For the flight phase and mission test cases, COMPEX interpolates the more dense expected result file to compare the test and expected results at the same test conditions. Tolerances can be defined as either a percentage or an absolute value. For each test run a complete set of comparsion statistics is generated and test points failing the tolerance criteria are highlighted. In addition, COMPEX produces a file containing the test conditions, the respective results and the amount out-of-tolerance in the same format as the test results and expected results. This file format is unique because it allows nearly immediate graphical analysis of the data at Interactive Graphics and Data Analysis sites located throughout the company.

The test case inputs, BSP execution output, expected results, test results and comparsions are archived in summary hardcopies, and in detail on microfiche and magnetic tape. All previous test points can be accessed and reviewed to analyze software changes or to re-create previous test conditions. The basic flow of data through the performance validation system is shown in Figure 4. With the exception of operator commands to initiate the individual steps, the procedure is fully automated from entry of the BSP-format inputs into the mainframe computing system until the test report is ready for final analysis.

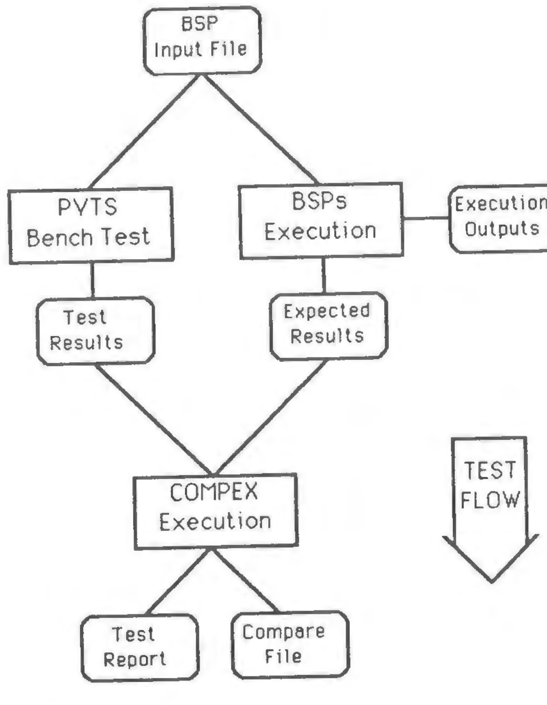


Figure 4 Test Procedure Flow

Experience-to-Date

Validation testing of the 737-300 Flight Management Computer has been at least a six-day-a-week, sixteen-hour-a-day effort since early in 1984.

Nearly one hundred performance function tests, each with about ten separate cases and over one hundred test points, have been designed, executed, and analyzed in this period. The sheer volume of testing accomplished to date clearly demonstrates the capabilities of the system. But this level of detail and test rigor has not come at the expense of scarce engineering or laboratory test bench time. While nearly a third of the planned 737-300 Flight Management Computer testing is in the performance functions, less than a tenth of the scheduled test bench time has been assigned to performance function testing. Using the Performance Validation Test System, the complete Flight Management Computer performance data base (including all aerodynamics, propulsion, and atmospheric data) can be interrogated in less than eight hours. No user interaction is required to execute the over fifty complete and separate tests designed to rigorously exercise the operational flight program performance data base software. Often this testing is completely unattended and is accomplished between the hours of midnight and 8:00 A.M., while the test bench would normally be unused.

Test analysis has kept pace with the rapid rate and high volume of test results. Test reports are nearly all complete the week after a series of twenty or more tests are run. Further, the test documentation is far more detailed and complete than has been possible on previous programs. This is expected to have a large payoff in the future in update or modification validation efforts.

Conclusions

The above described procedures and testing tools have been highly successful in the 737-300 Flight Management Computer validation program. By taking a "user-based" and integrated approach to the test system design, validation of the performance functions has been transformed from the "most risky" area of the 737-300 FMCS testing to one of the "least risky." Further, testing rigor and documentation have been enhanced while the tester is freed from the laboratory to concentrate on test analysis. More traditional approaches to software validation could not have suceeded under the time and budget constraints of the 737-300 program. Even without those constraints, it is unlikely that manual testing methods could have provided even a small fraction of the results already produced. The above described approach has proved to be a very efficient and effective technique for the validation testing of the 737-300 Flight Management Computer.

Acknowledgments

While it is not possible to thank everyone involved in this project, the efforts of B. Blair in developing the Boeing Standard Programs, and the efforts of K. Singh and J. Weston in Performance Validation Test System development must be recognized. The author is grateful to them and the scores of others that contributed to the success of this program. The author would also like to thank the FMC vendor, Lear Siegler, Inc., for their outstanding support.

84-2664

A METHOD FOR TESTING A DIGITAL FLIGHT CONTROL SYSTEM WITHOUT THE USE OF GROUND SUPPORT EQUIPMENT

BY

H. E. HANSEN, LEAD ENGINEER, ELECTRONICS
MCDONNELL DOUGLAS CORP.
ST. LOUIS, MO.

ABSTRACT

On 26 May 1983, a McDonnell Douglas F-15 Eagle fighter became the first to fly with a Digital Flight Control System (DFCS) programmed in a higher order language. The topic of this paper is the testing of the F-15 DFCS prior to that historic flight. Laboratory testing procedures and test equipment are described. Testing at the airplane is the major emphasis, describing the equipment resident on the airplane and the software, dubbed Maintenance BIT, that was used to transform the airplane into its own self-tester. Also covered are actual examples of how Maintenance BIT was used to solve problems at the airplane that might have been difficult or impossible with special test equipment. Other accomplishments resulting from this approach are also enumerated.

Introduction

Recognizing a number of technological advancements were occurring which could importantly affect the direction of digital flight control system (DFCS) design in the near future, the McDonnell Aircraft Company initiated an Independent Research and Development (IRAD) program in the summer of 1981 designed to assess advancements in the following technology areas: microprocessors, higher order languages (HOL's), floating point arithmetic, and parallel processing. [1]

The particular hardware configuration that served as the supporting system for technology evaluation is built around the present F-15 Eagle Dual Control Augmentation System (CAS). This was done for two reasons: the electronic portion of the CAS could be modified at minimum cost/time to provide all of the required digital features; and, the system could be flight tested with minimum aircraft modification.

The software testing of the DFCS included static gain checks, frequency responses, integration in the F-15 hydraulics lab and evaluation by experienced F-15 test pilots in a simulator. This latter test, depicted in Figure 1, allows the pilot - should he suspect an anomaly in the DFCS - to immediately make a comparison with the production flight control system. These tests demonstrated that an F-15 with the DFCS would perform exactly like production F-15's that have analog Flight Control Computers, if the DFCS performed in the aircraft as it had during testing. This could happen only if the more than 100 discrete and analog signals were brought into the DFCS in the memory locations expected by the software.

Traditionally, special support equipment has been used to check for proper operation at the aircraft. Developing such equipment for this limited research project would have been too costly and time consuming. Nevertheless, it was extremely important to verify proper integration of the DFCS and the input signals.

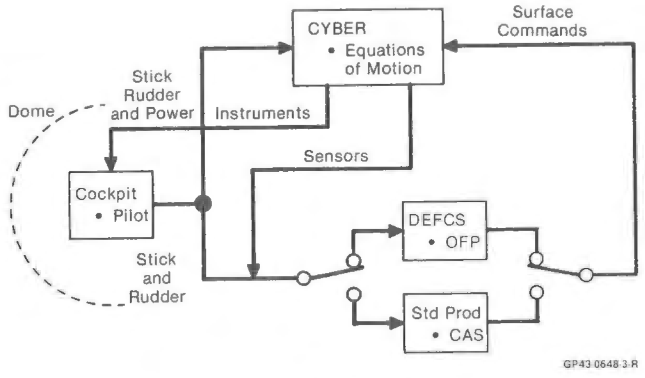


Figure 1
Block Diagram Man-In-Loop Simulator

During lab testing, a Digital Development Set (DDS) was used to interrogate memory to check the input signals. The DDS consisted of a computer (PDP 11/34), a Keyboard to input commands, software (operating system) to process those commands, an output device (CRT) to observe the contents of memory and a link (data bus) between the computer and the DFCS. The F-15 test bed had a computer the Central Computer (CC). Functionally, the DDS could have been used for DFCS checkout at the aircraft. However, the DDS consisted basically of commercial equipment designed for a laboratory environment. To ruggedize this equipment for flight line use would have been too costly and time consuming. Consequently it was necessary to use a different approach.

It was observed that the standard F-15 equipment could be adapted to serve as the controls and displays to a self-contained DDS; the Navigation Control Indicator (NCI) panel could serve as a keyboard and the Vertical Situation Display (VSD) could serve as an output device, while the MIL-STD-1553A multiplex bus was the data link between the CC and the DFCS. The only missing element was the software which could use this equipment to verify the integration of the DFCS with the aircraft. This paper describes how, through use of standard F-15 equipment and the addition of the proper software, the F-15 test aircrft was transformed into its own self-tester. The software that uses the resident equipment to convert the airplane into a self-tester is referred to as "Maintenance BIT" and "Preflight BIT". That software will be described in detail.

System Description

The flight control hardware configuration that served as the supporting system for this project is built around the present F-15 Eagle Dual Control Augmentation System (CAS). The present production F-15 control system is shown in Figure 2. The CAS Computers have been modified by replacing the existing dual analog computers in each Line Replaceable Unit (LRU) with two digital processors (Z8002 microprocessors)[2], associated memory (24K) of PROM, 2K of RAM and 1K of Non-Volatile Memory (NVM)) and converters. The modifications were carried out by the Astronics Division of Lear Siegler Corporation, Santa Monica, and are described in detail in Reference (3).

1553A Multiplex Bus is designed so that software in the CC and firmware in the DFCS control the flow of data between the DFCS and the CC. Therefore the DFCS software can communicate with the CC merely by putting data into and removing data from predefined buffers in the DFCS memory.

The Navigation Control Indicator (NCI) Panel is pictured in Figure 4. Information consisting of up to five digit numbers may be entered thru the North (N) and East (E) windows from the NCI Panel into the CC. By selecting the proper Mode, Data Select and Dest Data switches these data may be interpreted as commands for the CC and DFCS much as typical computer operating systems - UNIX, RSX11M, etc. - do. The principal difference is the constraints of the hardware, i.e., the commands in this case must be numbers instead of acronyms. This has not been any real encumbrance in the use of Maintenance BIT. The two NCI windows may be used for commands or for a command and for data read out from the CC. This function will be covered in more detail in the discussion of Maintenance BIT software.

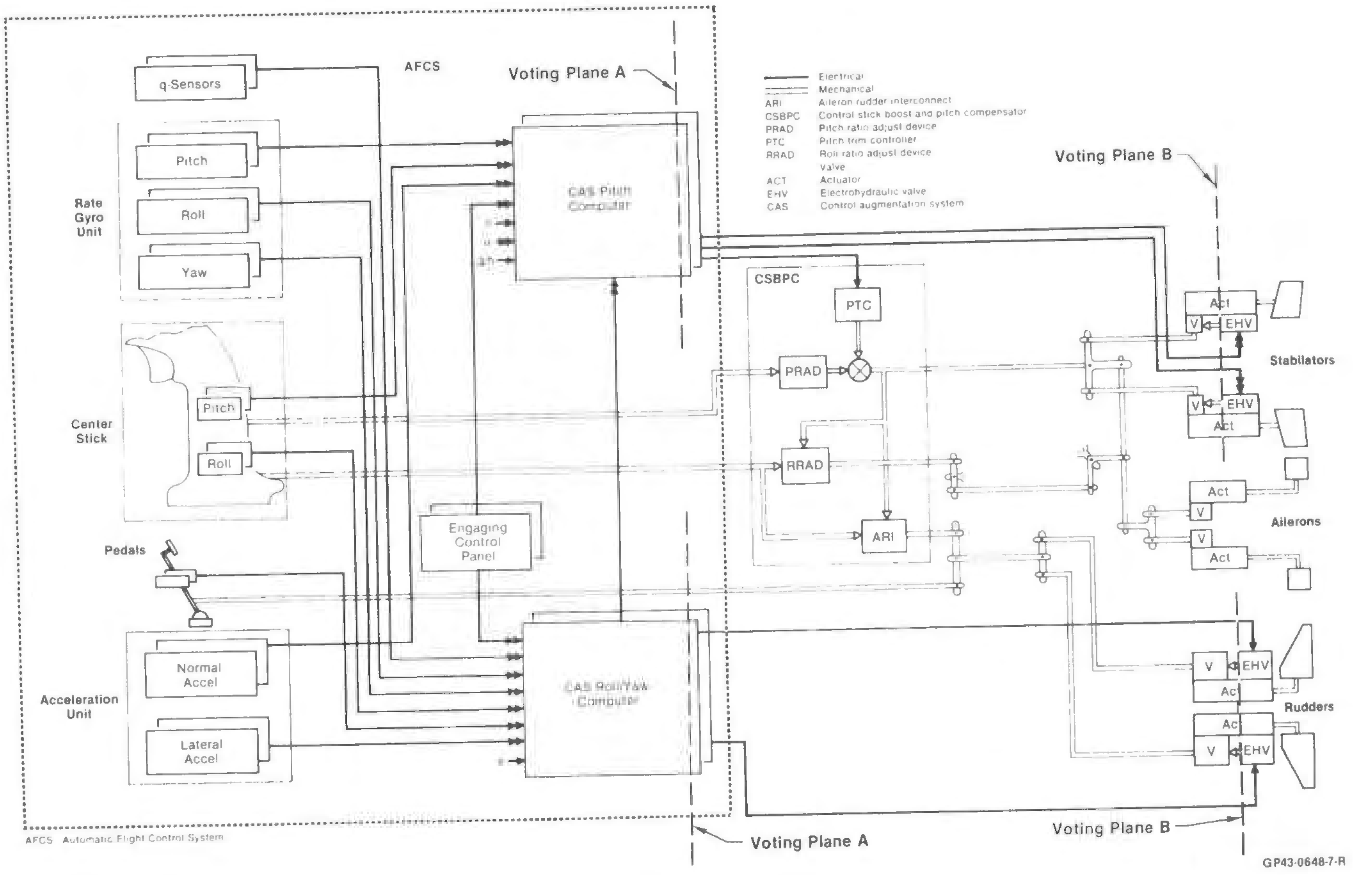


Figure 2
F-15 Present Mechanical and Dual Command Augmentation Flight Control System

The resulting system is a parallel processing system with the Operational Flight Program (OFP) partitioned by control axes[4]. Two processors operate on the pitch axis software and two operate concurrently on the roll/yaw software. The two pitch processors are frame synchronized, as are the roll/yaw processors; the pitch processors, however, are unsynchronized with respect to the roll/yaw processors. The dual processors in the pitch and roll/yaw axes are consistent with the dual (fail-safe) nature of the F-15 production CAS.

The DFCS is connected to the F-15 CC via the 1553A Mux Bus and the NCI and VSD are connected to the CC via the H009 bus as shown in Figure 3. The

The Vertical Situation Display depicted in Figure 5 has an 18 character alpha numeric buffer and serves to read out in edited format information from the DFCS. This along with the NCI (which can only read out unedited data) forms the system's display units.

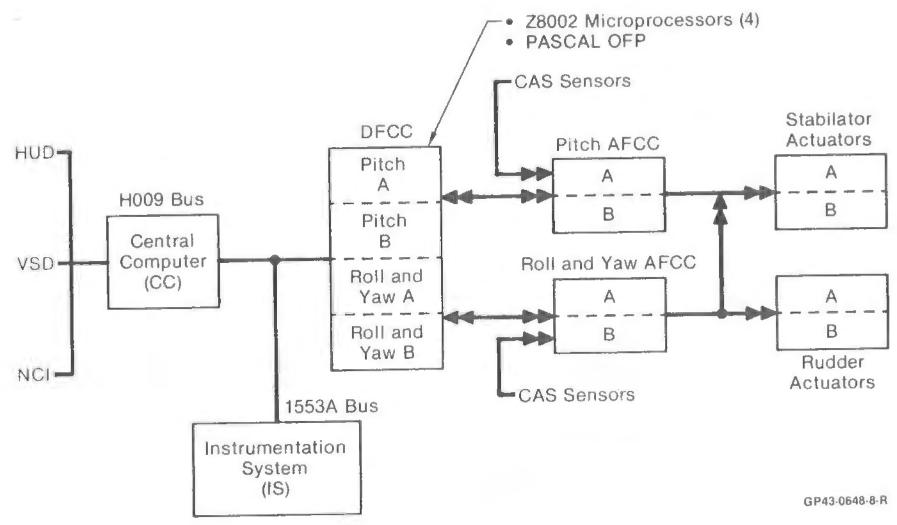
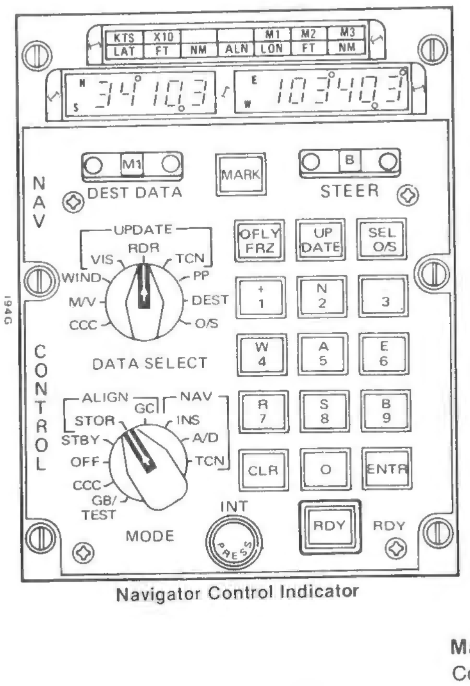


Figure 3

Description of Maintenance BIT

The general features of Maintenance BIT are illustrated in Figure 6. The major features are identified by the following level numbers.

Level 1: This level of maintenance BIT consists only of software changes to the Central Computer (CC). In the normal cycling of the Control Laws (CL) in the Digital Flight Control Computer (DFCC), various measurand data are continuously sent across the MIL-STD-1553A multiplex bus for instrumentation purposes. With this feature of Maintenance BIT, the CC takes selected measurand data and displays it on the Vertical Situation Display (VSD) as shown in Figure 5. The CC receives the data and transforms it into engineering units that are easily read on the VSD. The parameters are divided into several lists. Each list of data can be selected from one of the four DFCS computers. A list has approximately 10 parameters in it. The parameters in each list are displayed successively on the VSD. Each parameter in the list is displayed for about three seconds until the next parameter is cycled onto the VSD. The list repeats itself until the option is disengaged or a different list is called up. To choose a parameter list, a code is entered through the NCI panel. The CC receives the code from the NCI over the H009 Multiplex Bus. The NCI panel with an example code is shown in Figure 4. This feature of BIT doesn't affect the cycling of the control laws in any manner, so it can be used



Navigator Control Indicator

Code			Data Displayed on VSD
5	1	0	Variables From Pitch Channel A
5	2	0	Variables From Pitch Channel B
5	3	0	Variables From Roll/Yaw Channel A
5	4	0	Variables From Roll/Yaw Channel B
5	1	1	Discretes From Pitch Channel A
5	2	1	Discretes From Pitch Channel B
5	3	1	Discretes From Roll/Yaw Channel A
5	4	1	Discretes From Roll/Yaw Channel B
VSD Display	Channel	Menu	

GP43-0648-6-R

Figure 4
Maintenance BIT
Code for Level 1

in-flight as well as on the ground to monitor the flight control parameters in the measurand list.

Level 2: This feature of Maintenance BIT allows the operator to position aircraft control surfaces to selected positions to check the computer/control surface interface. The surface positions can be moved with a resolution of two percent of full scale. Surface position is entered through the NCI. The CC takes the data via the H009 multiplex bus and sends the commanded position over the MIL-STD-1553A multiplex bus to the DFCS. Level 2 can only be initiated on the ground when the aircraft is weight-on-wheels and the BIT consent switch is on.

Level 3: With this feature any memory address in any of the DFCS channels can be read out. The address and desired DFCS channel are entered into the NCI. The CC takes the address and requests the particular DFCS channel to send the contents of the location to the CC over the MIL-STD-1553A multiplex bus. The CC takes the address contents and displays it on the VSD. The NVM contains various test routines and data logs. For quick changes in the NVM this feature of BIT allows the contents of NVM to be changed. Thus, changes in various test routines can be made without having to remove the memory boards to the flight controls lab.

Level 4: This feature allows a memory location to be read in real time. The address of a location in one of the DFCS channels is entered through the NCI. The CC takes the contents of the address location and displays it on the VSD. The data is updated at 20 Hz so that changing numbers can be tracked.

Unacceptable or illegal commands present no problems to the user; Maintenance BIT merely waits for an acceptable one. Also it should be pointed out that non-passive commands, such as alter Non-Volatile Memory, require special acceptance procedures thus ensuring that they are not invoked accidentally. Maintenance BIT is very closely associated with and complements Preflight BIT in the flight testing phase of the program. The general features of the Preflight BIT used on the DFCS program are illustrated in Figure 7.

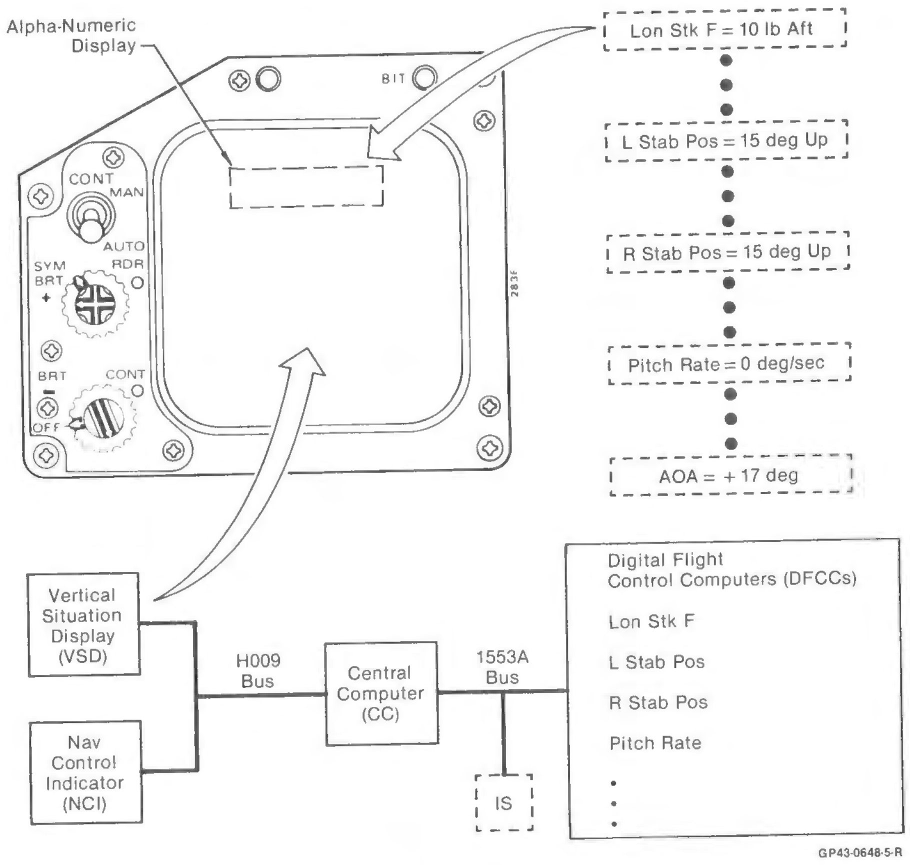


Figure 5
Maintenance BIT
Level 1

Description of Preflight BIT

Preflight BIT is designed to test various safety monitors in the Digital Flight Control System (DFCS). Preflight BIT is implemented interactively by an operator in the cockpit using the CAS Control Panel.

F-15 Flight Control System (FCS) consists of a mechanical system and the CAS. The CAS is fail-safe. Should the CAS shut down, meaning the surface commands are centered, the mechanical system is still in operation. The production CAS is implemented in a dual channels per axis

Level 1

- Allows Selected Instrumentation Parameters Sent on 1553A MUX Bus to be Displayed on Aircraft Vertical Situation Display (VSD)
- Parameter Menus Are Entered Through the Navigation Control Indicator (NCI)
- Functions While Control Laws Are Cycling
- Used on Ground to Check Sensor Inputs and Sensors-to-Control Surface Steady-State Operation and Gains
- Can Be Used in Flight to Monitor Control Parameters

Level 2

- Can Position Control Surfaces by Selected and Controlled Amounts
- Can Fully Exercise the Watchdog Timer
- Initiated Only While Aircraft Is on the Ground
- Entry Made Through NCI
- Used on Ground to Check Computer-to-Surface Actuator Interface

Level 3

- Allows Reading Any Memory Location in the DFCCs
- Allows Changing Values in Non-Volatile Memory (NVM) on the Aircraft
- Allows the History of Airplane Power Fluctuations, CAS Failures/Shut Downs Etc., to Be Sent to the VSD After the Flight.
- Memory Addresses and New contents Are Entered Through the NCI
- Memory Address Contents Are Displayed on the VSD

Level 4

- Allows Continuous Monitoring of Any Memory Location in the DFCCs
  - Memory Addresses Are Entered Through the NCI
  - The Address Contents Are Displayed in Real-Time on the VSD
  - Operates While Control Laws Are Cycling

GP43-0648-1-R

Figure 6
Main Features of Maintenance BIT

configuration. The channels monitor each other continuously at voting planes A in the Control Laws (CL) and voting plane B downstream from the CL in the hardware. These voting planes are illustrated in Figure 2. Should a differential between the channels occur at either of these voting planes, the CAS will be shut down and control of the airplane reverts solely to the mechanical system. Voting planes A and B have been retained in the FCS used in the DFCS. Voting plane B has been completely unaffected by any changes made for the DFCS. However, voting plane A is now mechanized in software.

In addition to the fail-safe features carried over from the standard production CAS, an additional safety feature has been added due to the unique nature of the digital FCS. This is the Watchdog Timer (WT). The WT is a countdown clock in hardware. Should the WT countout it will cause the CAS to be disconnected. To prevent the WT from counting out and disconnecting the CAS, it has to be updated during each cycle through the control laws. Should the normal program cycling stop and the WT not be updated, the WT will quickly countout and shut down the CAS. Examples of failures that could prevent normal program cycling are a memory alteration sending the CPU into an invalid region of memory resulting in the CPU halting or looping infinitely. The CPU could also have a failure which halts the CPU. Here again the WT will quickly shut the CAS down.

There is a known sequence of CAS shutdowns and resets that occur during the running of Preflight BIT. After the last reset the CAS will remain engaged and begin cycling the Control Laws. Preflight BIT indicates a problem if the exact sequence of CAS shutdowns and resets cannot be completed.

Interaction of Preflight BIT and Maintenance BIT

If a problem is discovered by Preflight BIT then Maintenance BIT needs to be invoked and extensive trouble shooting can be performed. An example of such a problem occurred in the lab with the arrival of a new DFCS. On running Preflight BIT, a problem with the Stabilator shutdown in voting plane A was observed. Stabilator fail tests using Maintenance BIT were performed, and it was noted that the left stabilator could go full authority down while the right stabilator remained in neutral and no shutdown occurred. With this analysis in hand the problem was very quickly traced to a grounded wire leading to the Pilot Reset Switch (Figure 8). Thus the CAS was continuously being reset before the disengage logic in the hardware could shutdown the CAS.

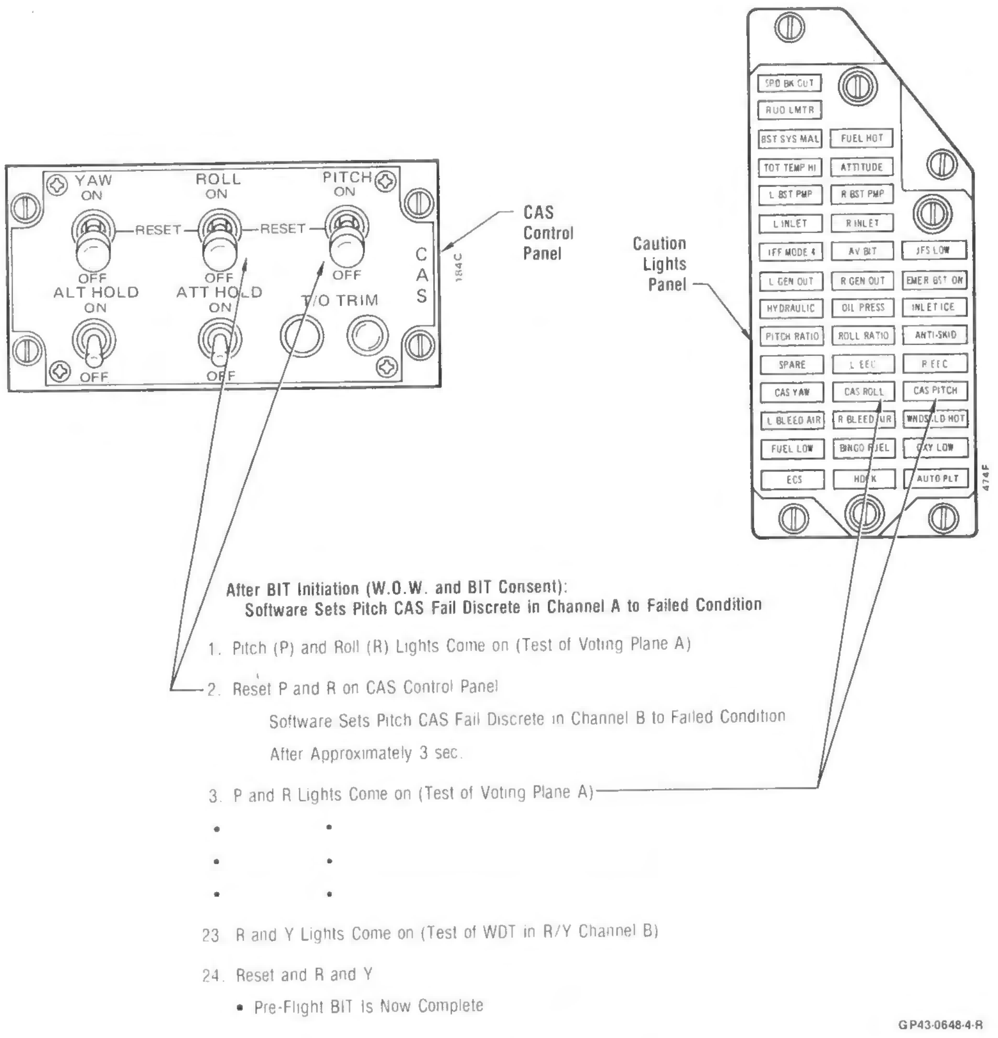
- Test Is Intended Primary for Use by Pilot and/or Ground Crew as a Pre-Flight Check of Proper Operation of Fail-Safe Features
- Tests Hardware Associated With Fail-Safe Monitors in FCS
  - — Checks Circuitry Associated With Control Law CAS Shutdown
  - — Checks Ability of Watchdog Timer to Shut Down CAS
  - — Tests Hardware Monitoring at Actuator Level
- Test Is a Series of Apparent CAS Shutdowns With Associated Resets
- Entire Procedure Takes About 1 min and Uses Only Reset Switches on the CAS Control Panel and Lights on the Warning Light Panel

GP43-0648-2-R

Figure 7
Overview of Pre-Flight BIT

Other Examples of Maintenance BIT

After the installation of the DFCS into the aircraft, but prior to the first flight, a constant rudder oscillation was noticed whenever the DFCS was turned on. The oscillation was judged too serious to fly with. After watching the inputs from various A/D's into the yaw channels, it was surmised that the problem might be caused by noise on an A/D. Since the A/D's were multiplexed, the noise on all yaw inputs should be the same (there were only two bits of noise anyway) - it was suspected that one or more of the paths in the Yaw axis could not tolerate any noise at all. A procedure was set up using Maintenance BIT that would allow selective removal of noise from any or all inputs in any desired combination. By this procedure it was determined that yaw rate was the



After BIT Initiation (W.O.W. and BIT Consent):
Software Sets Pitch CAS Fail Discrete in Channel A to Failed Condition

1. Pitch (P) and Roll (R) Lights Come on (Test of Voting Plane A)
2. Reset P and R on CAS Control Panel

   Software Sets Pitch CAS Fail Discrete in Channel B to Failed Condition

   After Approximately 3 sec.
3. P and R Lights Come on (Test of Voting Plane A)

•   •

•   •

•   •

23. R and Y Lights Come on (Test of WDT in R/Y Channel B)
24. Reset and R and Y
    - Pre-Flight BIT Is Now Complete

Figure 8
Control Panels Used by Pre-Flight BIT

culprit. The solution was to bring yaw rate into the yaw channels via two A/D's, as coarse and fine yaw rate.

Aliasing Problem

It is always desirable from a computer usage perspective to compute the control laws at the lowest iteration rate possible so more functions may be included. Preliminary analysis indicated that the yaw control laws could be iterated at 40 hertz instead of 80 hertz at which the pitch and roll control laws are iterated. But confirmation by the simulator or actual test flight was highly desirable. So two OFP's, one with yaw iterated at 40 hertz and one with yaw iterated at 80 hertz were prepared for evaluation at the simulator.[5] Comparison of two versions of the OFP at the simulator requires only replacing the memory cards in the Roll-Yaw channels of the DFCS. Both versions were flown. Unfortunately, the pilot after flying in the simulator was unable to make a definitive statement as to the viability of the 40 hertz yaw iteration rate. Plans were then made for a test flight using a dual mode, i. e. flying for a while with a 40 hertz iteration rate and switching to an 80 hertz iteration rate during the flight. This could be done using Maintenance BIT. An OFP was developed with both an 80 hertz yaw rate and a 40 hertz yaw rate with the pilot keying thru the NCI panel his choice during the actual flight. This would have given inflight evaluation of two OFP's with different flight characteristics. However, ground vibration testing prior to that flight detected an aliasing problem using the 40 hertz OFP thus obviating the above exercise. Nevertheless the utility of a system such as Maintenance BIT is clearly demonstrated. For otherwise two distinct flights would have had to have been flown. And besides the costs and program delay, parameters - weather conditions, different pilots etc. - change, making multiple flight evaluation less desirable Also one can see the potential for very easily customizing the flight characteristics of the airplane for various pilots, conditions and missions.

Summary and Conclusions

A method for converting the test bed aircraft into a self-tester has been described. It has been shown that special software can be developed that can utilize equipment resident on the aircraft to turn the aircraft into a self-tester.

Advantages of the self-tester approach over special purpose test equipment are:

1) External test equipment does not have the same response time as the real system on the airplane and therefore, due to different sampling rates and inherent lags, gives a false impression of how the flight control system works. Simultaneous surges to both pitch channels induced by test equipment in the lab, for example, may cause a pitch CAS shutdown. This is not the way the system works. Maintenance BIT does not have this problem as it is part of the system.

2) Maintenance BIT is always on the airplane, therefore no time consuming search for and attaching of special ground equipment is needed and after the maintenance, analysis, or checkout is completed there is no concern that damage to the system (bent pins etc.) has occured as there is with special test equipment.

3) Unlike special test equipment, one can perform a test flight without disengaging Maintenance BIT, and continue testing in flight. Also the airplane may be at a remote site, anywhere, at any time, and will have its own test equipment with it.

4) Since only changes to software are required, the self-tester can be easily modified or expanded to meet changing program needs. The success and utility of this method is demonstrated by the fact that it or a variation of it have been or are being used on several subsequent test flight programs with no consideration given to additional test equipment at the airplane.

5) With the trend toward more CRT's in the cockpit, this method will become even more attractive in the future.

REFERENCES

1. T.F. Westermeier, H.E. Hansen, "Recent Digital Technology Advancements and Their Impact on Digital Flight Control Design", NAECON, May 1983.

2. Zilog Inc., "Z8000 CPU Technical Manual", January 1983.

3. Lear Siegler, Inc. (Astronics Division), "F-15 DEFCS Hardware and Interface Description", Report No. ADR-843/1, dated 27 July 1982.

4. T.F. Westermeier, "Parallel Processing Applied to Digital Flight Control Systems; Some Perspectives", NAECON, May 1981.

5. T.F. Westermeier, H.E. Hansen, "The Use of High Order Languages In Digital Flight Control Systems":, IEEE/AIAA 5th Digital Avionics System Conference, November 1983.

84-2666

REAL TIME DATA PROCESSING FOR AVIONICS TESTING ON THE A-6E

Paul T. Richards, P.E.
Manager, Engineering Analysis

James Lehmann
Senior Programmer Analyst

Grumman Data Systems
Calverton, New York 11933

Abstract

Many of the nation's aircraft manufacturers have accelerated their flight development programs by producing flight test results while the aircraft is airborne, rather than waiting for results after the aircraft has landed. This is called real time flight testing. The major portion of real time testing today is limited to airframe testing; avionics testing uses post flight data analysis. Some recent work at Grumman expands real time testing to the avionics development area.

A groundbased computer system is described here. This system receives digital data from the A-6E's onboard computer and processes it to give test results while the aircraft is airborne. Experience with this system indicates that it embodies a viable approach to rapid development of embedded avionics software.

Introduction

Since the early 1970s Grumman Aerospace Corporation has conducted its airframe flight test programs using a computer facility called the Automated Telemetry Station (ATS). The test aircraft is instrumented with a variety of sensing transducers, and signals from these transducers are multiplexed into a Pulse Code Modulation (PCM) serial data stream. The stream is telemetered to a ground receiving site where it is demultiplexed and analyzed using special and general purpose computers (see References). Analytical results are displayed at a work station called the Data Analysis Station (DAS), and a team of DAS analysts communicate with the aircrew through another engineer designated as the Test Conductor. This closed loop scenario proves cost-effective (Figure 1).

Real time testing has been applied to airframe development projects, addressing disciplines of stability, control, loads, flutter, propulsion and the like. Traditionally, flight development of avionics systems has not used the real time approach. Rather, the avionics data, which invariably comes from a computer, is recorded on tape and played back for analysis after the plane has landed. Flight safety is generally not a critical issue in avionics testing, so the impetus behind the real time approach is not as great as that for vehicle testing. It is now economic pressure and the increased availability of onboard computer power that spur the extension of the vehicle real time testing to the avionics area.

This paper describes work done at Grumman on a pilot real time avionics test program using the A-6E all-weather attack aircraft. A 16-bit parallel output from the aircraft's weapon system computer is serialized and transmitted to the ground. Processing algorithms perform engineering units (EU) conversion and limit checking, and present results to DAS engineers in plot and tabulation formats.

The emphasis here is on the computing aspect of the ground processing system. Prefatory material describes the A-6E weapon system and the conventional post-flight data processing. The new real-time approach is then described and conclusions drawn.

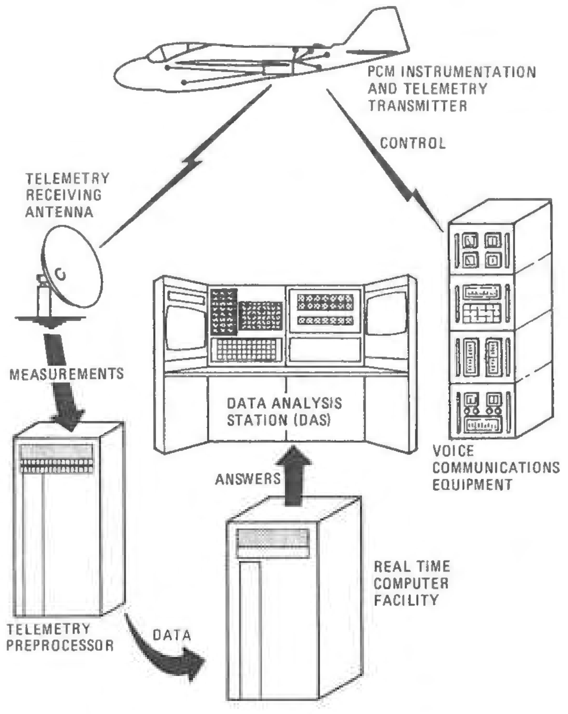

PCM INSTRUMENTATION AND TELEMETRY TRANSMITTER
CONTROL
TELEMETRY RECEIVING ANTENNA
MEASUREMENTS
DATA ANALYSIS STATION (DAS)
VOICE COMMUNICATIONS EQUIPMENT
ANSWERS
REAL TIME COMPUTER FACILITY
TELEMETRY PREPROCESSOR
DATA

Figure 1 Automated Telemetry System

A-6E Weapon System

The A-6E Intruder aircraft, a sophisticated air-to-ground weapon system, uses a general purpose onboard computer (Figure 2). Mission inputs come from three major sources: (1) a track-while-scan, forward looking radar used to designate ground targets, (2) a local level inertial navigator to provide aircraft position, velocity and attitude, and (3) an air data computer to sense properties of the air mass. These inputs are polled by the weapons system computer on an equispaced time schedule. The computer holds numerous software algorithms for weapon delivery, with major outputs of steering signals to the pilot and precise weapon release timing. A third output is a 256 x 16-bit parallel buffer of internal data used for flight test analysis. It is this ancillary output which is of most interest here.

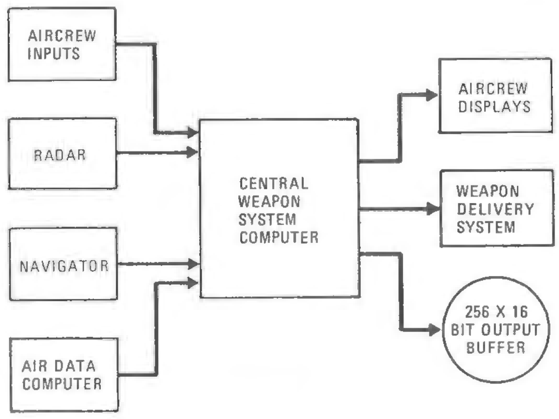


Figure 2 Simplified A-6E System

The 256 x 16-bit ancillary output buffer contains a tightly packed group of engineering data which the A-6E's computer generates as it executes its weapon delivery algorithms. Each piece of data is called an "Engineering Units (EU) parameter," and there are potentially 1500 such EU parameters defined for the current A-6E embedded software code. In order to fit 1500 pieces of information into a relatively small 256 x 16-bit buffer, the software engineers have multiplexed the buffer in both space and time.

Space multiplexing is done by close packing of integer data words. One-third of the EU parameters are single-bit discretes, often corresponding to cockpit switch settings, and are represented by a value of zero or one. Others are short n-bit counters, with n less than the 16-bit word size. About one-half of the parameters are scaled 16-bit integers, which are multiplied by a scale factor to obtain engineering units. By packing these data close together, with few unused bits, a large number of parameters can be squeezed into a single 256 x 16 raw data record. Further, some EU parameters, called calculated parameters, are functions of two or more raw parameters, and only the raw parameters are packed into the buffer; the calculated parameters are computed on the ground. A small number of EU parameters are double precision in that they use 32 bits to describe them.

Data compression is increased by time multiplexing. Certain engineering units parameters are of interest only when the weapon system is operating in certain modes. Therefore, selected 16-bit words in the data buffer are time shared according to the system mode, so that a single raw data word may represent a large number of EU parameter values depending on the slowly changing mode of operation. System mode can always be determined from fixed locations in the raw data buffer, and these bits are combined in boolean expressions to form a time multiplexing indicator called the "branch code."

Current Data Processing

Until recently, all Grumman A-6E flight test data processing was done after the aircraft had landed. During a flight, the 256 x 16-bit data buffer is recorded on a one-half inch computer compatible tape, and this tape is switched on and off by the aircrew at the beginning and end of each data run to conserve valuable recording time. The raw data buffers are written to the tape, one buffer per physical tape record, at fixed sampling intervals.

After the flight, the raw data tape is normally processed using a multiple-step procedure. First, a complete engineering units conversion is performed to produce a fully expanded EU file on disk or tape. Data words in the raw record are shifted, masked and floated to produce up to 1500 EU values at each sample time. The EU data is then scanned by a Quick Look program to determine instrumentation validity, data loss, and similar functions related to the onboard hardware. During that same process, critical features of the recorded data are extracted: time-of-day, data run counts, physical record counts, and selected engineering units values at certain times (e.g., navigation state at time of air-to-ground weapon release).

The Quick Look output is used by analysts to select time slices for further investigation. This data is then passed to various special purpose analytical programs for further processing. The analytic program used most often generates columnar listings of the EU values in chronological order (Figure 3).

The data analyst, then, views the EU data on paper at his desk one or two days after the flight test. Such turnaround time is often acceptable, since a number of flight test activities can occur in parallel. It becomes unacceptable, however, when a problem arises which forces serial testing. A second disadvantage of the post-flight processing method becomes apparent when either: 1) the onboard recording system fails, or 2) a particular test condition is not achieved due to erroneous switch settings or flight conditions. With current data processing methods, both these failures go undetected until the Engineering Units and Quick Look program are run and the results analyzed, and the failures may never be detected if they occur when the data tape is switched off. Flight time thus wasted adversely affects the development schedule by approximately 20 percent.

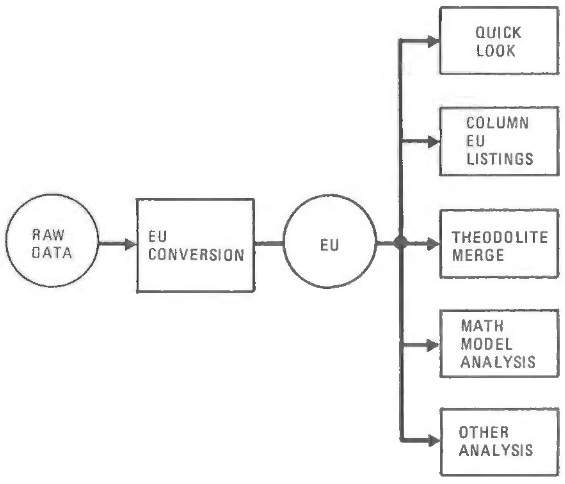


Figure 3 Batch Data Processing

Advantages of Real Time Flight Testing

During real time flight tests, the raw data buffers are telemetered to the ground and the EU information is presented to the data analyst as it occurs during flight. Time wasted due to faulty instrumentation and erroneous flight conditions is therefore eliminated and tape recording capacity no longer shortens flight times (especially at the high data rates associated with modern avionics).

Aside from these obvious features, the main advantage of real time flight testing is the increased rigor it brings to the testing problem. With EU data available in real time, the flight crew, the system designers, and the data analysts must be keenly aware of all of the aspects of a given test. This makes test planning and anticipation of test results extremely important. Emphasis on the processing and analysis of EU data is therefore shifted from after-the-flight to before-the-flight, and this invariably results in an early shakeout of inconsistencies and, ultimately, a better weapons system.

A further advantage of real time avionics testing is its early detection of bugs in the embedded code. In practice, this is of marginal value only, since the bugs cannot, in general, be corrected while the aircraft is airborne. A telemetry uplink allows in-flight software changes, and this technique has been used successfully on at least one Grumman weapons program, but not on the A-6E.

The primary disadvantage of real time avionics flight development is that the pace of a real time test is not conducive to careful analysis of complex embedded algorithms. Such analysis must be postponed to the postflight period in a quiet office. Experience shows that for new embedded codes most of the software problems occur in the housekeeping logic, and not in the analytical portions which have been previously validated by computer simulations. Simulation results combined with real time data in an artificial intelligence algorithm promises a solution to this problem.

Description of the Real Time System

The largest part of a real time data analysis system for flight testing is the ground hardware and the operating system software. Although it is ultimately an applications program that presents test answers to the data analyst, a good deal of hardware and software is necessary to accept the raw data from the onboard computer inflight, convert it to a serial telemetry stream, transmit and receive the stream, reconstruct the raw data inside an applications program, and present displays of results to the analyst on the ground. These functions, along with archival and immediate access storage of the raw data, occur within one second of real time, and all are available in the Grumman ATS system (see Reference 1).

The ATTACK Program

Of most interest here is the applications software that processes the 256 x 16 bit data buffers into real time outputs. For the A-6E project, this program is named ATTACK. A raw data buffer is presented to ATTACK in a Fortran common area by calling a single real time data serving routine. Flags and timing information are passed with the raw buffer to prevent the reprocessing of buffers already processed. On the other hand, if the real time buffers are arriving faster than ATTACK can service them, the operating system's deferred processing mode is invoked to queue the data on disk before the applications program receives it. The rate at which ATTACK processes the raw buffers depends on: 1) the sampling interval requested by the DAS data analyst (i.e., processing occurs every nth buffer, where n is established by a keyboard entry during initialization), and 2) the complexity of the analytical algorithms themselves. In practice, the processing rate is adjusted manually to fit the analyst's needs on a test-to-test basis.

The data flow is shown in Figure 4. Real time telemetry data is received in the Advanced Telemetry Pre-processor Computer (a SEL 32/77) and passed to the DAS mainframe (a CDC CYBER 740). The ATTACK application runs in the mainframe to produce plots and data tabulations on the analyst's cathode ray tube. In addition, two non-real-time paths are used to hold non-real-time information. First, the SELECT program runs interactively before the flight to establish initialization quantities. Second, the QLOOK program runs interactively during or after the test to analyze the raw data buffers using remote terminals. Data is not immediately available for analysis of the remote sites. The analysts stationed remotely must wait until a particular time segment is completed before the buffers are made available to them. This time lag is usually 2 or 3 minutes.

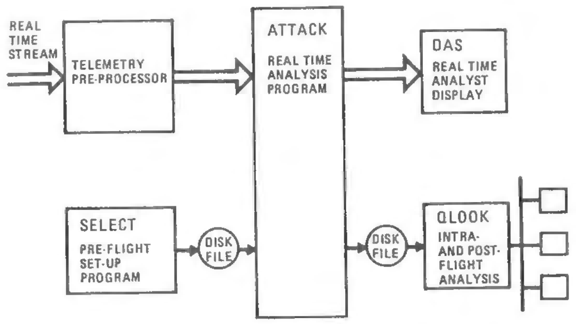

REAL TIME STREAM
TELEMETRY PRE-PROCESSOR
ATTACK
REAL TIME ANALYSIS PROGRAM
DAS
REAL TIME ANALYST DISPLAY
SELECT
PRE-FLIGHT SET-UP PROGRAM
DISK FILE
DISK FILE
QLOOK
INTRA- AND POST- FLIGHT ANALYSIS

Figure 4 Attack Data Flow

As an initial feasibility study for the A-6E aircraft, the real time ATTACK code does little more than convert the raw buffer to engineering units and present these results to the analyst in various formats suitable for immediate interpretation.

The EU conversion algorithm uses a series of indirect addresses to select bit positions, scaling, and alphanumeric labels for each of the up to 1500 EU parameters, as shown in Figure 5. One EU value is calculated at each call to the algorithm, so that only those parameters requested by the DAS analyst at any given time are actually computed. This design gives a central processor time advantage over the postflight approach of computing all EU values for each raw data buffer, and it is well suited to real time analysis since the human analyst can assimilate only a small number of values at once.

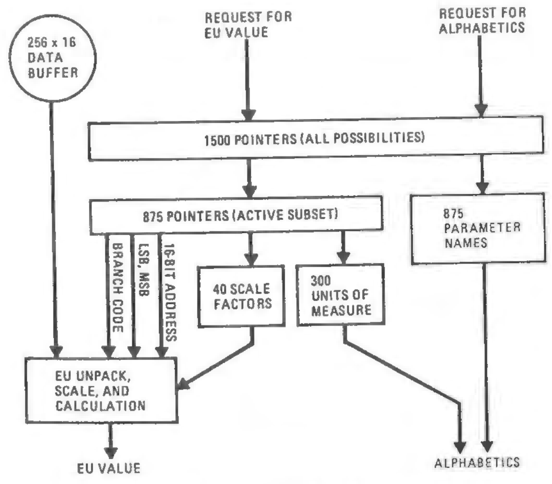


Figure 5 EU Conversion Algorithm

In practice, the analyst does not need to see <u>all</u> the parameters. Rather, he needs to see <u>any</u> parameter at will. In the ATTACK software design, such flexibility is provided as follows (Figure 6):

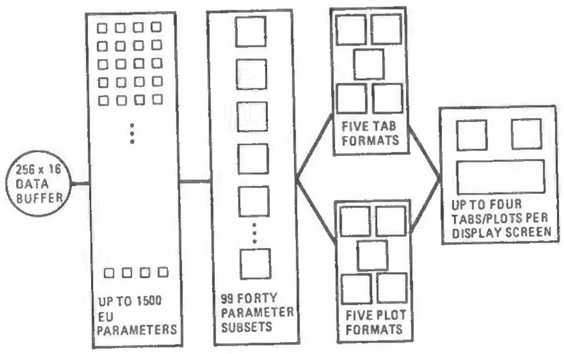


Figure 6 Real Time EU Parameter Selection

1. Convert to engineering units a small subset (40) of the total EU parameter population (1500).
2. Predefine up to 99 such subsets before the flight test.
3. Give the analyst the ability to select one of the 99 subsets for real time processing, and then to change his selection quickly during the test.
4. Distribute the 40 EU parameter values to ten different plot and tabulation formats for display in real time.
5. Present the plots and tabulations on the analyst's screen in various predefined combinations, with up to four presented simultaneously.

The plot and tabulation formats are standard parts of the ATS system. Tabulations are displayed in two different forms: 1) columnar tabulations which build EU values in columns down the display screen as time progresses, and 2) refresh tabulations that use a 10 x 16 matrix to display up to 160 pieces of numeric or alphanumeric data in fixed positions with changing values. Similarly, real time plots are generated in three forms: 1) single variable time histories, 2) dual variable time histories, and 3) dual variable cross plots.

In the current version of the ATTACK code, plot scaling must be defined preflight. This turns out to be a major drawback. Future ATTACK versions will have scales adjustable during real time, but no automatic scaling based on data values will be used. Experience shows that autoscaling is distracting to the analysts, especially when temporary telemetry dropouts cause gross errors in the EU values.

In addition to the flexible EU data display, the ATTACK program also flags values that fall outside predefined upper and lower limits. This straightforward calculation is especially useful for verifying the initial onboard configuration for a specific flight test. Using EU parameter and limit selections set up before the test, the DAS analyst can quickly determine if all of the various flight conditions and switch settings are correct for a given objective.

Note that, even with analyses as simple as EU conversion and limit checking, the design of the ATTACK software emphasizes careful predefinition of the real time data processing, pushing data analytics to the left on the project schedule.

Flight Procedures

During development of the A-6E weapons system, the flight tests are subdivided into short time segments called runs. The aircrew is briefed on the initial conditions and sequence of events for each run. The DAS analysts set up their EU parameter and limit selections for the same runs, and store these selections on disk for access later. (The SELECT code is used by the analyst for this purpose and is described later.) Once airborne, the crew is in contact with the Test Conductor engineer who coordinates all flight activities, including DAS analysis. At the start of each run, the ground based analysts verify proper initial conditions, usually using a refresh tabulation (Figure 7), and then track the progress through the run using columnar tabulations and plots. The real time software does not contain logic to validate the sequence of events through the run, nor does it contain any statistical analyses of the data generated during the run. Such features have been incorporated into other real time flight development programs at Grumman, but are not slated for the A-6E until 1986.

The SELECT Program

The SELECT program works together with ATTACK to accomplish the real time flight development objective. SELECT runs preflight, ATTACK runs during the flight. SELECT established up to 99 different subsets of 40 EU parameters that are used by ATTACK in real time. Characteristics of the parameters, such as label, units, default values, scale factors, offsets, limits, and plot scales, are added or modified. After entering up to 40 parameter selections interactively, the analyst writes this information as a record on a random access file called the select file. This file is then used by the real time software to extract the EU parameter for the raw data record and to flag out-of-limits conditions.

The SELECT code converses with the user. After displaying a prompt, SELECT waits until an entry is completed and then processes it by checking for syntax errors and limit conditions (such as a maximum of 40 parameters in the current record). There is a teach mode in the program to help the user learn the commands and their syntax, and this mode can be entered at any time.

In practice, the user opens an old select file, reads a specific record, modifies its contents, and then writes the modified record to the file with the same or a different record number. Upon completion of this record processing, the user catalogs the file as a new file or replaces the file with the newer version.

```
M228FLT11          A6E            HJ=10712          VA=653          11/0 4/83  ET  15.43.32.5   FLT002
PSI=262   SAC=16415            Q80=0    A6TST1A4          REC=01       BR=040  ST  15.43.36.6
Q41    QRMD           HJ          NQUAN    SAC                          EVENT STARTED
```

LABEL	UNITS	VALUE	OOL	LIM 1	LIM 2
Q21	IN OPR	1.00000000		1.00000000	1.00000000
Q41	TGT 1		0.00000000	1.00000000	1.00000000
Q69	RKT RP	1.00000000		−111111.00	111111.00.
Q70	TRN RM	1.00000000		−111111.00	111111.000
Q85	SP AT	1.00000000		1.00000000	1.00000000
QRMD	4PI TR		1.00000000	0.00000000	0.00000000
CRTGT	FEET	1111.11111		−111111.00	111111.000
HJ	FEET		10712.000	7700.00000	8300.00000
NQUAN	DIM-LES		0.00000000	6.00000000	6.00000000
THETA1	RADIANS	−0.0012463		−3.0000000	5.00000000
TRKCOD	DIM-LES		0.00000000	33.000000	33.000000
VAPRIM	FT/SEC		653.62500	375.00000	450.00000
VXJ	FT/SEC	−87.12500		−111111.00	111111.000
VYJ	FT/SEC	−599.37500		−111111.00	111111.000
WEAPCD	DIM-LES		0.00000000	60.00000	60.00000

Figure 7 Refresh Tabulation Used For Checking Initial Run Conditions

The following is a list of SELECT's record processing commands:

CREATE	Writes a record to the current select file. The user can specify the record's number (position) in the file or let SELECT position it as the last record. An optional title may be supplied by the user.
RECORD	Reads a record from the select file. This record becomes the current record (i.e., the subject of all further record commands).
DISCARD	Clears the current record (i.e., all parameter data is removed from the record).
SELECT	Adds EU parameters to the current record by alphanumeric labels or ID numbers. This command must be issued before any other command to process a parameter.
DELETE	Removes a parameter previously selected from the record.
DEFAULT	Supplies a default value for a specific parameter. The EU value assumes the default value when it is not specified in the raw data record due to time multiplexing.
LIMIT	Sets upper and lower limits for a selected parameter. These limits become the default plot scales.
SCALE	Stores a multiplicative scale factor for an EU parameter (e.g., to convert from radians to degrees).
OFFSET	Sets an additional offset value for a selected parameter.
PLIMIT	Establishes upper and lower plot scales.
PLT1--PLT5	Stores format indicators for plots presented in real time.
TAB1--TAB5	Stores format indicators for tabulations presented in real time.
SORT	Sorts parameter information in the current record according to label, ID, tab, or plot.
DLABEL	Displays the labels and IDs of all of the parameters as positioned in the current record.

In addition, the following commands are used to manipulate SELECT files:

ATTACH	Copies a permanent SELECT file for user modification. This is called a local file.
CATALOG	Makes the local file permanent on the disk. No other changes can be made to the permanent file unless it is attached as a local file.
PURGE	Removes a permanent file that has been cataloged.
REPLACE	Replaces the current permanent file with the current local file. The local file is given the same name as the permanent file it replaces.
PFILE	Prints the contents of the local file on a line-printer.

The QLOOK Program

QLOOK is an interactive analysis program designed to be used for A-6E weapons system development during or after a real time flight test. It uses a standard graphics terminal (Tektronixs 4014) to analyze raw 256 x 16 data buffers stored on a disk file by the real time ATTACK software. Since the ATS system software allows sharing of read-only files by multiple interactive users, any number of analysts may use multiple copies of the QLOOK code, located at any number of remote sites, to work on the same raw dataset. Files cannot be shared, however, by real time and interactive users, so the QLOOK analysts must wait until ATTACK analysts have completed their work before starting on the most recent dataset. This time lag is usually equivalent to the length of a test run, about 2 or 3 minutes.

The following is a list of QLOOK's interactive commands:

SELECT	Specifies, by name, one to forty parameters for analysis, out of a possible total of 1500. QLOOK keeps a cumulative list of the selected parameters so that additional parameters may be selected during QLOOK's execution.
DISCARD	Removes all previously selected parameters from the list of selected parameters.
LIST	Displays all currently selected parameters and corresponding units.
RECORDS	Provides the start record number, the last record number, and the record increment for time history analysis. The command, RECORDS 10 100 10 tells QLOOK that records are to be searched from record number 10 up to and including record number 100 and only for every 10th record in this interval shall data be retrieved.

SCAN	Reads rapidly through the data file to determine the total number of records in the file. This enables the analyst to properly define a desired record range.
EUVALS	Displays numeric data values of all selected parameters with their corresponding labels, units, record numbers and time tag.
PLOT	Generates graphics plots of up to two EU parameters, either time histories or cross plot.
STATISTICS	Obtains the average value, standard deviation, minimum and maximum value for the selected parameters in the records as specified.

Future Projects

Many other weapons systems development projects are using real time flight data analysis similar to that described here for the A-6E. The original Grumman investment in the Automated Telemetry Station was made 15 years ago, and major upgrade programs are in progress to make the switch from vehicle to avionics testing. The upgrades of necessity address four new areas:

- o Encrypted telemetry and a secure ground station, generally not required for vehicle testing, are generally required for avionics testing.
- o Data rates are higher for avionics than for (at least most) vehicle testing.
- o EU conversion and time tagging algorithms are significantly more complicated for avionics data, which usually come from computers and from high speed data busses, than for vehicle data, which usually come from analog transducers.
- o Avionics simulators, rather than flight simulators, must be used on the ground for crew and analyst training, and for comprehensive real time analysis and performance evaluation.

Grumman is addressing each area with priority indicated by the above list.

Conclusions

Flight data analysis using a telemetry downlink and a real time ground processing station is currently in use in the avionics weapons system development area, after having proven effective in the flight vehicle area for over 15 years. Aside from the obvious advantage of reduced flight test time due to rapid data validation and interpretation, the real time approach fosters detailed analytical planning early, and thereby produces more efficient and smoother development. Analytics as straightforward as EU conversion and limit checking have proven effective on a number of Grumman weapons system development projects, notably the A-6E attack aircraft.

References

(1) Schiano, C., and Simpson, E.M., "Grumman's Automated Telemetry System, Past, Present, and Future," AIAA-83-2760 presented at AIAA 2nd Flight Testing Conference, Las Vegas, Nevada, November 1983

(2) Dickhudt, J., "Scientific Data Processor: Hardware Pre-processing in the Real Time Flight Test Environment," AIAA-83-2755, Ibid.

(3) Meyer, R.R. Jr., and Schneider, E.T., "Real Time Pilot Guidance System for Improved Flight Test Maneuvers," AIAA-83-2747, Ibid.

(4) Perangelo, H.J., and Milordi, F.W., "Flight Flutter Testing Technology at Grumman," presented at the NASA Symposium on Flutter Testing Techniques, Flight Research Center, Edwards AFB, California, October 1975

84-2670

SAFETY-OF-FLIGHT AND QUALIFICATION TESTING FOR AVIONIC SYSTEMS

W. J. Hall, Jr.

Systems Engineer
General Dynamics, Fort Worth Division
Fort Worth, Texas

Abstract

The diverse environmental conditions which avionics equipment is subject to imposes a need for equipment that is capable of surviving in many different environments. Historically, design analysis has not been sufficient to determine if a piece of avionics equipment will operate properly under varied environmental extremes, therefore, it has become necessary to demonstrate that equipment will survive under these extremes. This paper deals with the use of qualification and safety-of-flight tests to demonstrate that the design of the equipment is sufficient to ensure the equipment operates under these extremes. Because of the cost of testing under environment, exhaustive, one-time, and worse case testing is used. Test criteria is selected that encompasses the conditions under which the equipment must operate, i.e., maximum or minimum temperatures, worse case power conditions, and explosive to caustic foreign materials. Exhaustive tests are performed for environmental conditions which have a cumulative adverse effect (moisture, vibration, etc.) and worse case testing is used for conditions which do not have a cumulative effect (power transients, etc.). Tests are developed to fit the actual usage of the equipment with MIL-STD-810 used as a guideline for both test criteria and test methodology.

Purpose

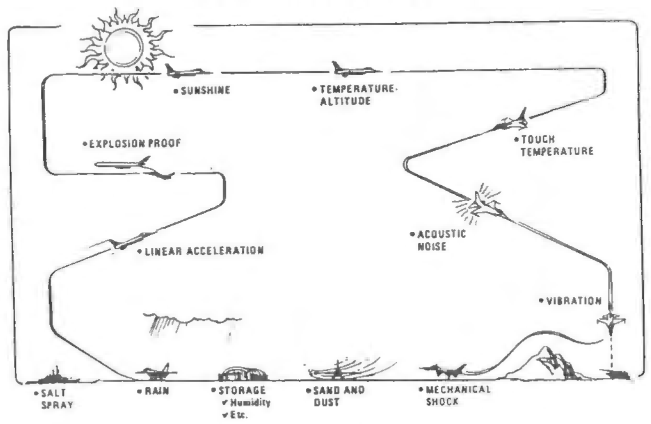
The purpose of this paper is to provide insight into Safety-of-Flight and Qualification test. A rationale and description of each test are provided. No attempt is made here to give detailed test methodology but only to provide a basic understanding as to reasons these tests are performed.

Objectives of Tests

Qual Tests

Qual testing is performed to verify that the equipment design is adequate to ensure that the air vehicle, applicable military standards, and the equipment procurement specification have been met. The intent is to determine (1) that under worst case environmental, and (2) throughout a lifetime of exposure to expected environmental conditions the equipment performs as specified.

AIRBORNE EQUIPMENT MUST BE DESIGNED TO WITHSTAND SEVERE ENVIRONMENTS



Safety-of-Flight (SOF)

SOF testing is performed to ensure that the equipment is safe for air vehicle operation. These tests demonstrate that the equipment poses no danger to the pilot, aircraft, or other equipment during both normal and emergency operating conditions.

Test Summary

Qualification (Qual) Tests

Qual testing subjects the equipment to all environments that are anticipated during its lifetime. The testing applies worse case conditions and exposes the equipment to a lifetime of simulated use in a shortened time period.

Safety-of-Flight (SOF) Tests

SOF assures basic pilot safety. Tests are shortened versions of qual tests in the areas of structural integrity, thermal and electrical safety, operation in explosive atmospheres and during explosive decompression.

Table I contains a list of tests. The tests are marked as to their application in safety-of-flight or qual testing.

Philosophy

Safety-of-flight tests are a subset of qualification testing that are performed in advance of full qualification tests. General Dynamics has separate SOF and Qual test items in all hardware contracts and, generally, all SOF tests are required to precede Qual tests.

There are two basic reasons for this: (1) SOF screens design problems before the expense/time of a full Qual test occurs, and (2) SOF satisfies basic flight safety concerns so that flight tests can start.

Most development schedules are tight and do not allow a significant amount of time for redesign and fixes that are usually required to pass qualification testing. Therefore, it is important to use SOF tests to pre-screen any equipment problems that can be identified and resolved successfully during SOF testing in order to preserve the development test schedule. SOF tests also allow the new basic configurations to get into flight test so that the aircraft system level problems can be found and worked as soon as possible.

Test Descriptions

Each test has a specific objective or purpose. The tests are in three basic categories with a fourth category which contains some miscellaneous tests for additional areas of concern. The four categories are: (1) strucural tests, (2) environmental tests, (3) functional tests, and (4) miscellaneous tests. These test categories are broken down by test in Table II below.

Functional Bench

The equipment is subjected to both static and dynamic operational tests to determine conformance to spec requirements, the operational integrity, and repeatability of the equipment design. These tests are performed under laboratory conditions and are tests of timing and output parameters.

		Qualification Tests	Safety of Flight Tests
I.	Temperature Altitude	X	X*
II.	Inoperative Storage	X	
III.	Vibration	X	X*
IV.	Gunfire Vibration	X	X*
V.	Mechanical Shock	X	X
VI.	Salt Spray	X	
VII.	Explosive Atmosphere	X	X
VIII.	Fungus Resistance	X	
IX.	Acoustic Noise	X	
X.	Humidity	X	
XI.	Linear Acceleration	X	
XII.	Explosive Decompression	X	X*
XIII.	Touch Temperature		
XIV.	Sunshine		
XV.	Moisture/Rain Resistance		
XVI.	Sand & Dust		
XVII.	Resistance to Fluids		
XVIII.	Panel Lighting		
XIX.	EMI	X	X*
XX.	Electrical Power	X	X*
XXI.	Functional Bench	X	
XXIII.	Category I Test	X	

* Safety of flight test differs from Qual test. Test duration or endurance levels are reduced and performance requirements are not as stringent as Qual test.

Table I Test List

(1) STRUCTURAL TESTS

- o Vibration
- o Gunfire Vibration
- o Mechanical Shock
- o Explosive Decompression
- o Linear Acceleration

(2) ENVIRONMENTAL TESTS

- o Moisture/Rain Resistance
- o Humidity
- o Salt Spray
- o Sand & Dust
- o Temperature Altitude
- o Explosive Atmosphere
- o Sunshine
- o Inoperative Storage
- o Fungus Resistance
- o Acoustic Noise
- o Resistance to Fluids

(3) FUNCTIONAL

- o Electrical Power
- o Panel Illumination
- o Functional Bench Test
- o Category I Test (OFP)

(4) MISCELLANEOUS TEST

- o Touch Temperature
- o RQT
- o Thermal Survey
- o EMI Test
  - oo Susceptibility
  - oo Radiated Emissions

Table II Test Categories

Vibration

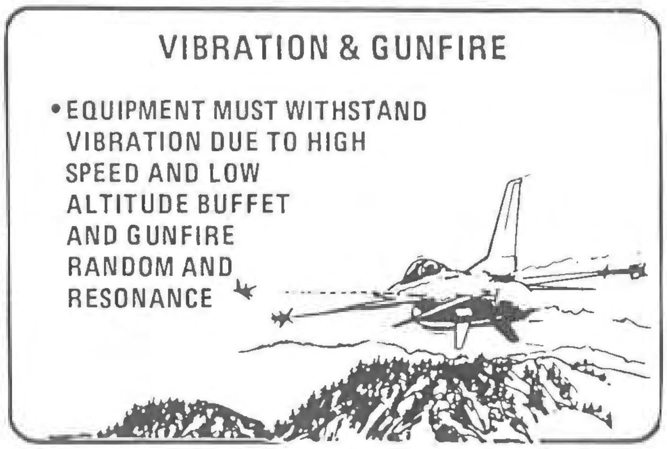
The equipment is subjected to vibration levels that are greater than actual operational levels, to simulate a lifetime of exposure to vibration in a shortened time period. Testing determines both the survivability of the equipment in normal vibration, and structural integrity in emergency situations.

1. Performance. The equipment is subjected to random vibration while operating. Vibration levels are representative of aircraft operational vibration.

2. Endurance. The equipment, while not operating, is subjected to higher random vibration levels than during performance testing to simulate a lifetime of exposure to vibration.

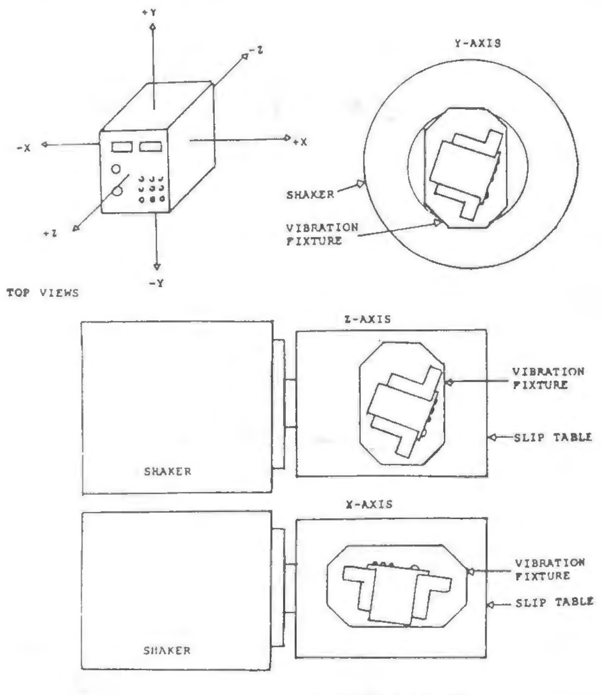
Gunfire Vibration

The equipment, while operating, is subjected to random and sinusoidal vibration representative of gunfire vibration levels.



VIBRATION & GUNFIRE

- EQUIPMENT MUST WITHSTAND VIBRATION DUE TO HIGH SPEED AND LOW ALTITUDE BUFFET AND GUNFIRE RANDOM AND RESONANCE



VIBRATION CONFIGURATION

Acceleration

The equipment is subjected to linear acceleration in each axis in excess of expected acceleration to assure that no structural fatigue or binding of mechanical components occurs. This test is usually not performed on totally electric systems.

Salt Spray

The equipment is subjected to a salt fog spray to verify its survivability in atmospheric exposure when operated near/over sea water.

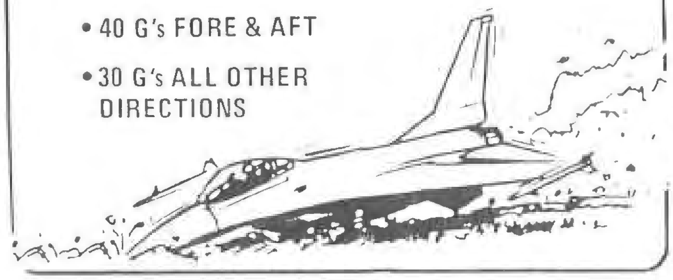
Mechanical Shock

The equipment is subjected to shocks in each axis. Shocks are of different magnitudes, those representative of normal operation and those representative of emergency conditions.

1. Crash Safety. The equipment, while not operating, is subjected to shocks representing crash loads. The equipment should not deform or rupture so as to present a hazard.

CRASH SAFETY & SHOCK

- ANY OBJECT THAT CAN BREAK LOOSE BECOMES A HAZARD (Particularly in Cockpit)
- 40 G's FORE & AFT
- 30 G's ALL OTHER DIRECTIONS

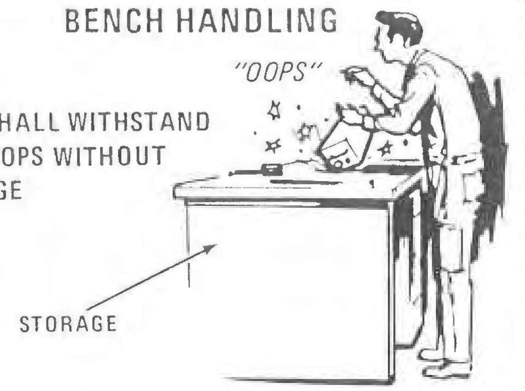


2. Basic Design. The equipment is subjected to normal shocks (landing, air buffet, etc.) while operating.

3. Bench Handling. The equipment, while not operating, is subject to shocks (droppage) which could be expected during bench handling.

BENCH HANDLING

- LRUs SHALL WITHSTAND 12" DROPS WITHOUT DAMAGE



Fungus Resistance

The equipment is exposed to fungus to determine its resistance to fungus when exposed to fungus in its normal operating environment. The test verifies that fungus does not grow on components and affect equipment operation. This requirement is normally satisfied by analysis on coatings and external materials to show fungus retardant characteristics.

Touch Temperature

The equipment when operating is subjected to operational temperature extremes to determine that sufficient heat dissipation is provided so that display panels and control knobs do not injure the pilot due to excessive heat.

Sunshine

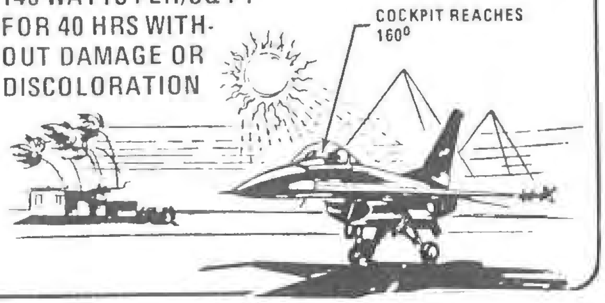
Equipment normally exposed to solar radiation is subjected to simulated lifetime exposure to assure that degradation of color, finish, etc., does not occur. This test must be performed either during Temperature Altitude Test or immediately following.

Sand and Dust

The equipment is subjected to blown sand and dust to assure that this environmental exposure will not harm the equipment.

SUN/SOLAR RADIATION, SAND & DUST

- EQUIPMENT MUST WITHSTAND 140 WATTS PER/SQ FT FOR 40 HRS WITHOUT DAMAGE OR DISCOLORATION

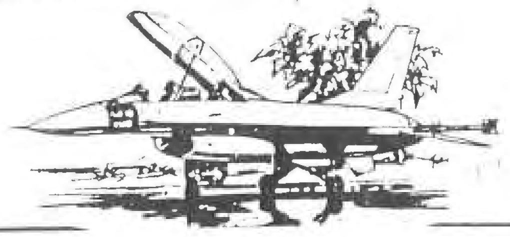


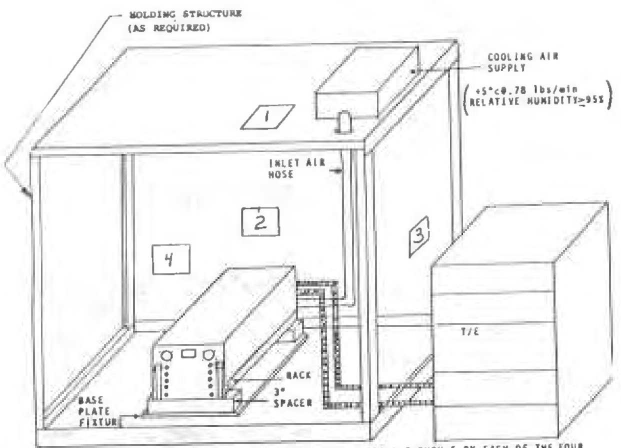
Moisture/Rain Resistance

The equipment is subjected to blown moisture, representative of lifetime exposure, to determine its resistance to rain and its ability to function after this exposure.

MOISTURE & HUMIDITY

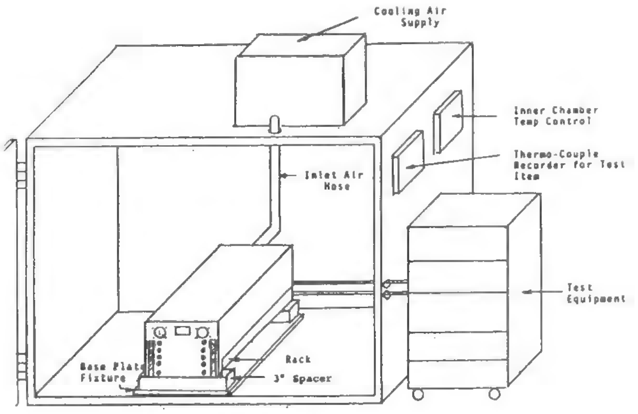
- EQUIP SUBJECTED TO 100% HUMIDITY TEST FOR 8 HRS WITH TEMPERATURES UP TO 120°F – PERFORMANCE MUST BE UNAFFECTED & LRU NOT DAMAGED





WATER NOZZLE SHALL BE RELATIVELY POSITIONED AT LOCATIONS 2 THRU 5 ON EACH OF THE FOUR WALLS AT APPROXIMATELY 45° INCLINE FROM CENTER LINE OF INU AND MINIMUM OF 30 INCHES ABOVE THE INU. POSITION 1 SHALL BE DIRECTLY ABOVE THE INU.

MOISTURE/RAIN
RESISTANCE TEST SETUP



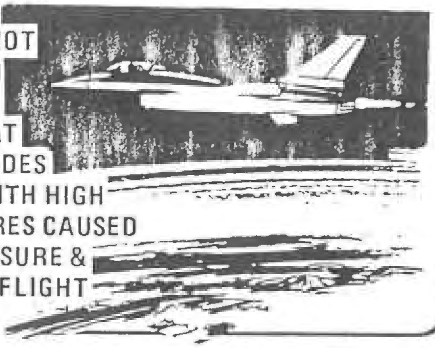
THERMAL ANALYSIS TEST SETUP

Temperature Altitude

The equipment is subjected to the temperature, altitude, and operating extremes representative of the actual operating environment. Tests cover both normal and emergency operating conditions. Test criteria are taken from MIL-STD-810 and modified to suite the operational envelope of the air vehicle.

TEMPERATURE & ALTITUDE

- EQUIP MUST OPERATE & NOT BE DAMAGED DUE TO LOW PRESSURES AT HIGH ALTITUDES COMBINED WITH HIGH TEMPERATURES CAUSED BY LOW PRESSURE & HIGH SPEED FLIGHT



Panel Lighting

Edgelit panels are subjected to solar radiation, temperature cycling, and power variation to assure no change in color or light intensity.

Thermal Survey

A thermal survey is performed prior to formal qualification testing to determine the temperature profiles to be used during qualification testing. The equipment is operated under a normal workload condition to determine a normal temperature profile and heat distribution and dissipation. This survey is a prerequisite for qualification test, but is not a test itself.

Resistance to Fluids

The equipment is subjected to various aviation fluids to assure its resistance ot hydraulic fluid, etc., to assure that components do not react or dissolve when exposed to these fluids.

Electrical Power

The equipment is subjected to power extremes and transients to determine its operational integrity under all expected transients. The test also assures that transients do not damage the equipment and assures that the equipment properly functions in emergency situations.

Reliability Qualification Test

Equipment having a mean-time-between-failure (MTBF) of less than 1,000 hours is subjected to temperature cycling, while operating, to demonstrate the equipment meets its specified MTBF.

Category I Test

Embedded operational software is tested to ensure its proper operation under all expected scenarios. Three test levels are used:

1. Computer Program Test and Evaluation (CPTE). Each module is extensively tested, using system emulation, as the software is developed.

2. Preliminary Qualification Test (PQT). Each function, which may or may not require more than one module for performance, is tested using system emulation.

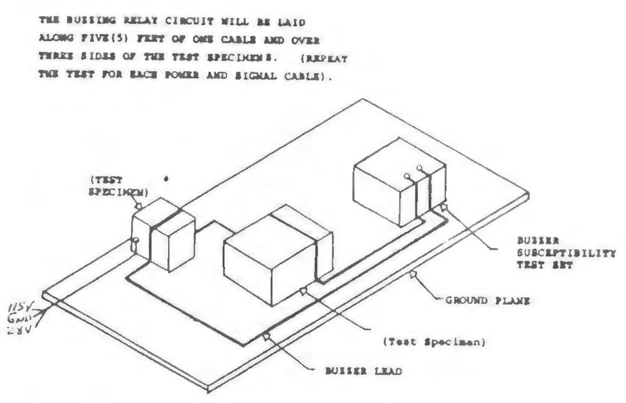
3. Functional Qualification Test (FQT). The completed, intact, operational software is tested embedded in the hardware.

Inoperative Storage

The system, in a non-operating mode, is subject to a simulated lifetime of storage under worst case conditions. Following storage, the system is required, without any maintenance being performed, to operate to specified requirements for new equipment.

EMI

The equipment is subjected to radiated emissions to determine if electromagnetic noise from other systems causes malfunctions. The equipment is also operated and its radiated emissions are measured to verify that the equipment operates within the acceptable levels for the aircraft and that it does not interfere with the operation of other equipment.


THE BUSSING RELAY CIRCUIT WILL BE LAID ALONG FIVE(5) FEET OF ONE CABLE AND OVER THREE SIDES OF THE TEST SPECIMENS. (REPEAT THE TEST FOR EACH POWER AND SIGNAL CABLE).
(TEST SPECIMEN)
BUZZER SUSCEPTIBILITY TEST SET
GROUND PLANE
(Test Specimen)
BUZZER LEAD
115V
GND
28V

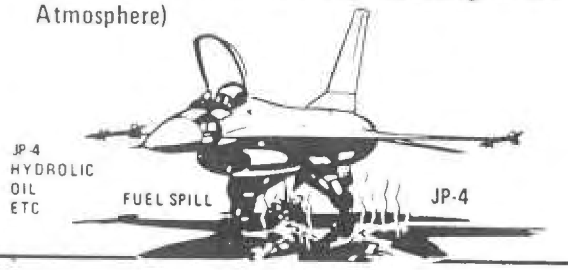
TRANSIENT SUSCEPTIBILITY TEST LAYOUT

Explosion Proof

The equipment is operated while exposed to atmosphere having an explosive mixture of air and flammable fluids to determine operational safety during fueling and in emergency conditions.

EXPLOSIVE ATMOSPHERE

- LRUs MUST NOT BE A SAFETY HAZARD IN AN EXPLOSIVE ATMOSPHERE (Internal Arcing, Uncoated Conductive Surfaces, Etc. Can Ignite the Atmosphere)



Humidity

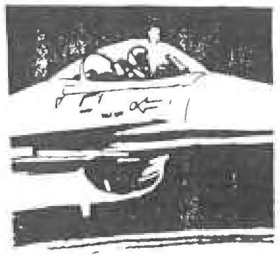
The equipment is subjected to atmosphere representative of lifetime exposure to humidity during storage.

Explosive Decompression

The equipment is subjected to rapid decompression (representative of loss of cabin pressure during ejection sequence) to determine structural safety.

EXPLOSIVE DECOMPRESSION

- A RAPID DECOMPRESSION (Cabin Equip) FROM 8K TO 60K MUST NOT CAUSE LRUs TO OIL CAN OR OTHERWISE EXPLODE



Acoustic Noise

The equipment is subjected to acoustic noise levels representative of the aircraft environment to assure that the equipment is not susceptible to these noise levels. This test is usually performed in conjunction with performance level vibration tests.

Note

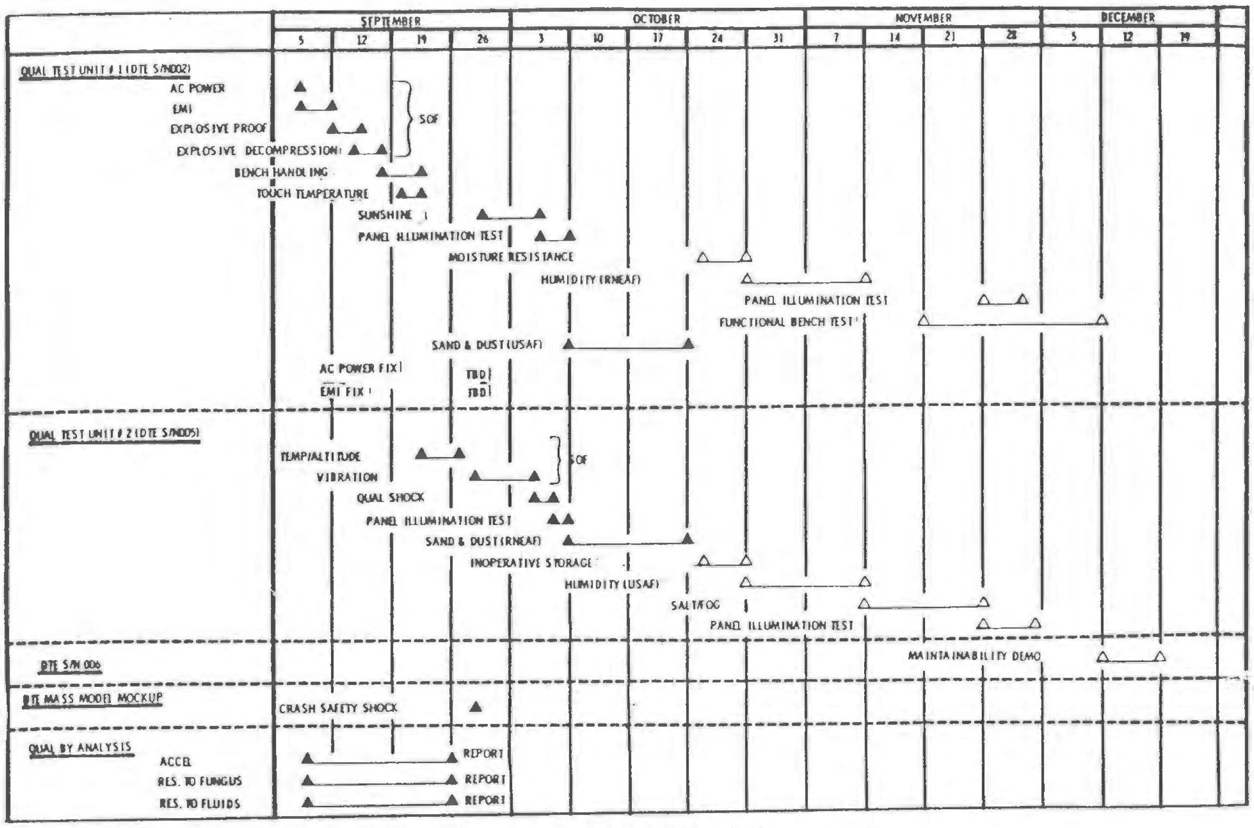
Prior to, and following completion of, qualification testing the equipment is subjected to and is required to successfully complete a full functional test or ATP. The equipment is functionally verified between each of the Qual tests. Equipment operation mode is specified for each test.

Test Sequences

Test sequences are important for two reasons: (1) some tests establish parameters to be used in/for subsequent tests (the Thermal Survey must be performed so that temperature levels for Temperature Altitude test can be derived), and (2) some tests must occur in a specific sequence or their results are invalid. Functional Bench test must be performed prior to Category I test so that the hardware-software interface can be verified during the Category I test.

Schedule

The SOF/QUAL schedule below is a sample schedule which is intended as a reference.



SOF/QUAL SCHEDULE

84-2726

EXECUTABLE ASSERTIONS AND FLIGHT SOFTWARE

Aamer Mahmood, Dorothy M. Andrews and E. J. McCluskey

Center for Reliable Computing
Computer Systems Laboratory, Stanford University
Stanford, CA 94305, USA

ABSTRACT

Executable assertions can be used to test flight control software. However, the techniques used for testing flight software are different from the techniques used to test other kinds of software. This is because of the redundant nature of flight software. The error detection capability of assertions is studied and many examples of the assertions used are given. The issues of placement and complexity of assertions are also discussed.

1 INTRODUCTION

Software testing involves generation of test data, determination of expected behaviour, program execution, observation of behaviour, and comparison of observed behaviour with the expected behaviour. The expected behaviour is usually determined by hand calculations, simulation, or by alternate solutions to the same problem. The test data can be generated either randomly, or exhaustively, or by using some kind of functional or structural analysis. The software testing process can either be static (peer review, walkthrough, flow analysis, symbolic execution) or dynamic (use of monitors and counters). [Adrion 82] and [Ramamoorthy 75] contain very good surveys of software testing and automated testing tools, respectively.

Exectable assertions can be used for dynamic testing of software. An executable assertion is a logical statement about the variables or a block of code, such that if there is no error during execution then the assertion statement results in a true value. Assertions not only serve as a good medium for documentation but they are useful for testing purposes throughout the lifecycle of software. They can be used for validation during the design phase, and for exception handling and error detection during the operation phase.

Assertions have been used in program verification [Floyd 67] [Hoare 69] [Manna 69] [Luckham 75] [King 76], in program testing [Stucki 75] [Andrews 81], and for reasonableness checks in the recovery block scheme of software fault tolerance [Horning 74] [Randell 75] [Carter 79]. The use of executable assertions for detecting hardware and software faults has also been suggested in [Saib 77] [Andrews 78] [Andrews 79]. The objective of this paper is to study the use of executable assertions for testing flight software. The error detection capability of assertions has also been studied in [Glass 80] [Andrews 81]. However, the software used in those studies was different. Also this study of assertions has a different emphasis, covering all aspects from writing of assertions to use of assertions.

2 EXECUTABLE ASSERTIONS

Assertions can be written by making use of either the specifications or some property of the problem or algorithm. Assertions are usually based either on the inverse of the problem, the range of variables, or on the relationship between variables. Some examples of assertions from [Hecht 76] [Mahmood 83] are as follows:

(1) If the problem is to find the discrete Fourier transform of an N point input sequence x(j), then Parseval's relationship can be used as an assertion

$$\sum |x(j)|^2 = \frac{1}{N}\sum |X(k)|^2 \qquad j\ ,\ k = 0 \text{ to } N-1$$

where X(k) is the discrete Fourier transform.
(2) If the problem is to find eigenvalues of a NxN matrix then the following must be true

$$\sum A_{ii} = \sum L_i \qquad i = 1 \text{ to } N$$

where A_{ii} are the diagonal elements and L_i are the eigenvalues.
(3) The longitude calculation by a routine in flight control software can be checked by

```
New_Long ≥ Prev_Long
        + (Prev_Long - Next_Prev_Long) - K
and
New_Long ≤ Prev_Long
        + (Prev_Long - Next_Prev_Long) + K
```

where K represents the threshold for the test.

3 DIGITAL FLIGHT CONTROL SYSTEM

The digital flight control system is an integrated system that provides autopilot and flight director modes of operation for automatic and manual control of the airplane during all phases of flight [DFCR-96 80] [Bendixen 83]. It includes two identical flight control computers known as FCC-201; each FCC-201 includes two CAPS-6 (Collins Adaptive Processing System) processors, referred to as Channels A and B. Figure 1 shows the architecture of the dual-dual redundant system containing two FCC-201 computers, and Fig. 2 gives the organization of each FCC-201 computer.

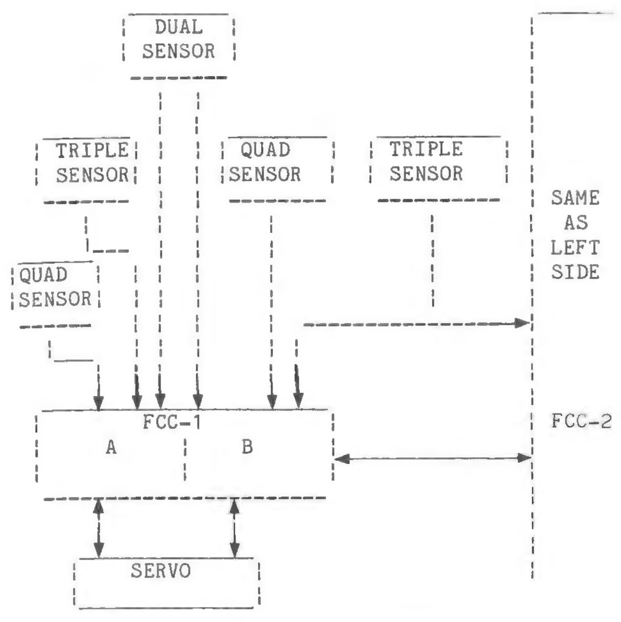


Fig. 1 Dual-Dual Architecture

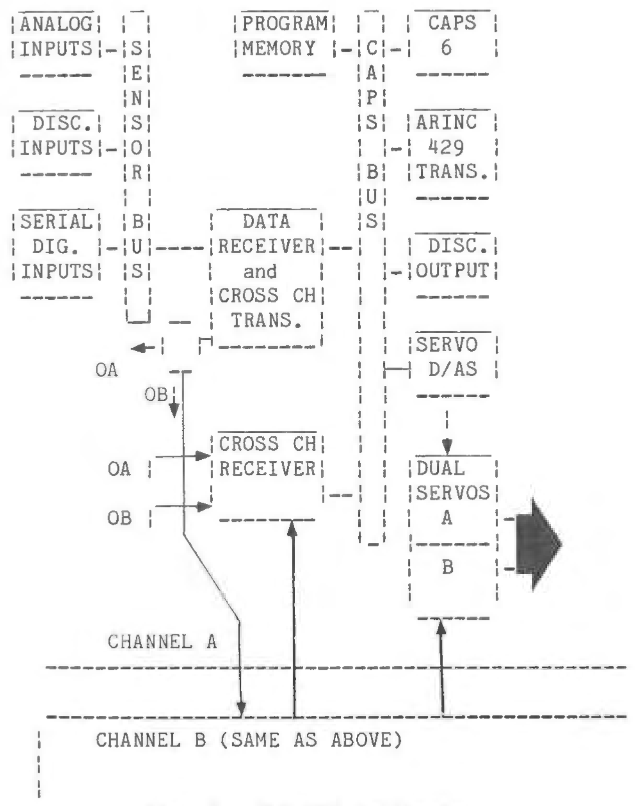


Fig. 2 FCC-201 Arhitecture

The flight control software is written in AED (Automated Engineer Design), an ALGOL like language. From a functional point of view it consists of five major parts: (a) control and navigation, (b) logic, (c) testing and voting, (d) input/output, and (e) executive. The executive software can be divided into two major groups, foreground and background. The foreground tasks consist of time critical functions such as command generation and executive monitoring. The background programs perform non-time-critical operations like processor self-test and memory checksum. Fig. 3 describes the foreground software structure and the timing relationship.

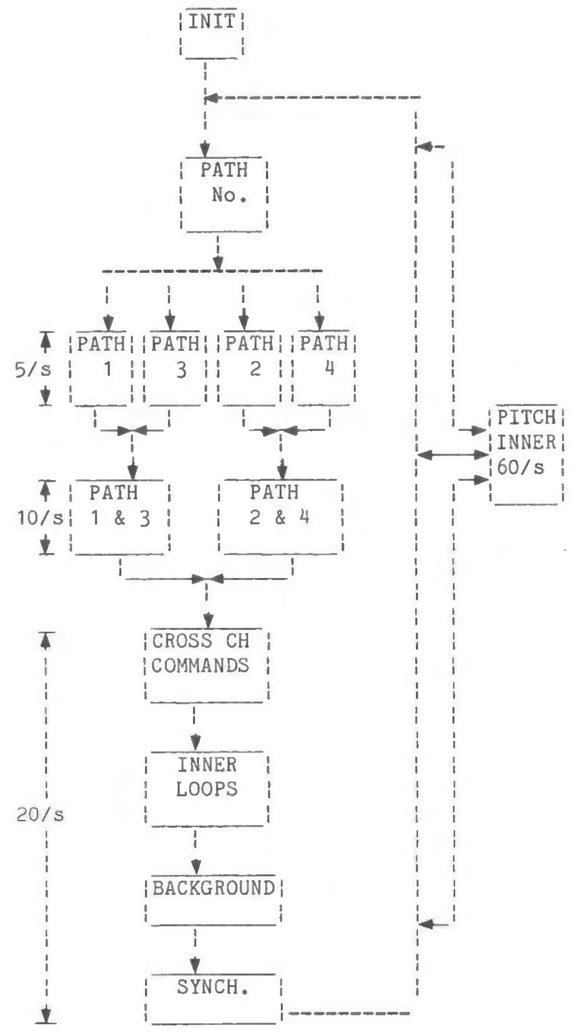


Fig. 3 Software Timing

The software consists of one segmemt performing pitch rate inner loop calculations at a rate of 60 per second. After every third execution of the 60 per second segment, the multipath software segment is restarted. This means that the multipath segment is executed 20 times per second. The multipath software segment contains segments which are executed at three different rates: 20, 10, and 5 times per second. At the end of each foreground execution, the executive schedules the background process. Synchronization between the two channels is performed 20 times per second. The software programs of the two channels are not identical. Functions performed by each of the two channels are shown in Fig. 4.

CHANNELS A and B
1 PITCH AUTOLAND
2 ROLL AUTOLAND
3 YAW AUTOLAND
4 TOGA
5 ENGAGE LOGIC
6 SERVO MONITORING
7 SYNCHRONIZATION
8 INSTRUMENTATION
9 ANUNCIATION
10 YAW SAS
11 INNER LOOPS

CH. A	CH. B
1 ROLL OUTER	1 PITCH OUTER
2 ALT ALERT	2 AUTOTHROTTLE
3 MODE LOGIC	
4 GLARESHIELD INTERFACE	
5 SENSOR COMPARISON	

Fig. 4 Flight Software Functions

4 TESTING FLIGHT SOFTWARE

The flight software was tested in the heading select mode (change of direction) at constant speed. Initial testing was done at NASA-AMES Flight Software Verification Laboratory [DeFeo 82]. The simulations for the second phase of testing, as described in Sec. 4.2, were performed on Stanford's DECSYSTEM-20.

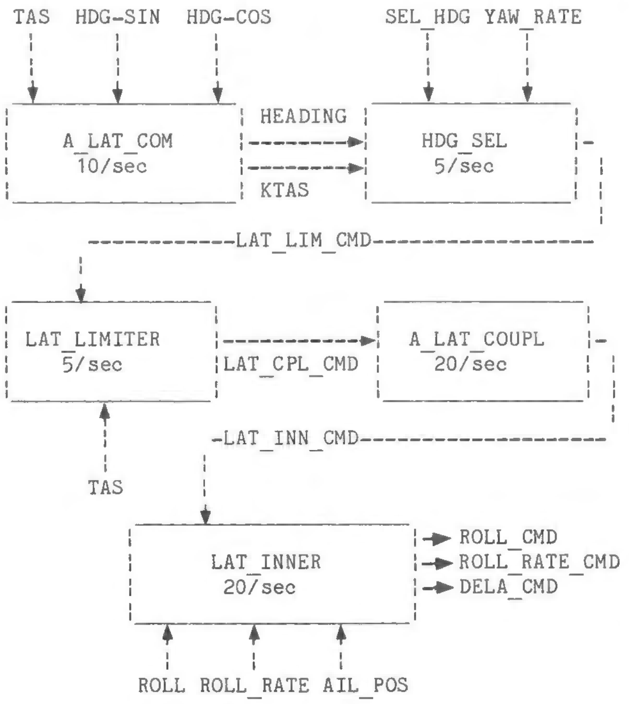


Fig. 5 Data Flow

The timing relationship between the relevant procedures and the data flow from the input (selected heading) to the output (commands to the ailerons) is shown in Fig. 5. A brief description of each of the modules is as follows:

(1) A_LAT_COM: This module computes heading and airspeed gain (KTAS) for use by the HDG_SEL module.
(2) HDG_SEL: This module performs the heading select computations using selected heading, true heading and yaw rate. It generates a roll-attitude command (LAT_LIM_CMD) which is passed to the LAT_LIMITER module.
(3) LAT_LIMITER: This module performs airspeed programmed magnitude and rate limiting of the roll-attitude command from the HDG_SEL module and generates LAT_CPL_CMD which is passed to the A_LAT_COUPL module.
(4) A_LAT_COUPL: This module performs coupling between the outer loop modules and the LAT_INNER module. It generates LAT_INN_CMD which is passed to the LAT_INNER module.
(5) LAT_INNER: This module performs the inner loop computations for the lateral axis. It includes roll attitude and rate feedback, lead-lag compensation, command limiting, aileron limit overide logic, etc. The output generated by this module includes ROLL_CMD, ROLL_RATE_CMD, and DELA_CMD (command to the ailerons).

4.1 TESTING - PHASE ONE

Initially the assertions were inserted only in the LAT_INNER module. Table 1 contains examples of some of those assertions. It was found that only 25 % of the errors inserted in the software were detected. The two main reasons were, the inadequacy of the first set of assertions used, and the nature of flight control software.

Table 1 Initial Assertions

(a) ABS(LAT_LIM_CMD) $\leq$ 0.5
(b) ABS(LAT_CPL_CMD) $\leq$ 0.11
(c) ABS(LAT_CPL_CHG) $\leq$ 0.0034
(d) ABS(LAT_INN_CMD) $\leq$ 0.18333
(e) ABS(ROLL) $\leq$ 0.165
(f) ABS(HDG_CHG) $\leq$ 0.0046
(g) TIME TO CHANGE HEADING $\leq$ MAXIMUM TIME
(h) HEADING ERROR DECREASES MONOTONICALLY

The flight control system is very redundant in hardware and software. Examples of the hardware redundacy are replication and hardware limiters. The software redundacy comes from the software limiters and from voting on the input and output. This redundacy tends to mask errors. As an example, consider the variable LAT_LIM_CMD (output of HDG_SEL module). Its value is limited to 0.5, as long as the heading error is greater than a fixed value. This makes the output independent of the input conditions. Similarily, LAT_CPL_CMD (generated by the LAT_LIMITER module) increases at a fixed rate to a fixed value. This makes the

LAT_CPL_CMD independent of the input changes. Another aspect of flight software which makes it different from the other software is that it contains a great number of boolean variables and decision points. This makes it difficult to write the same kind of assertions as were written for the other kind of more computational intensive software. For such a computational intensive code it is easy to use range assertions. This is not true for the flight software.

The inadequacy of the assertions used, was the other reason for low error detection. Ideally the assertions should be written during the design phase from specifications. The lack of any specification document made it very difficult to write good and meaningful assertions. Some of the main flaws in the assertions used were as follows:

(a) The assertions were only placed in the last module (LAT_INNER). It is very difficult to write such global assertions which can take every possible condition into consideration. The complexity of assertions starts to approach the complexity of the program itself. One solution is to use many simple assertions at various points in the program. Placement of assertions is very important for good error coverage. This has also been discussed in [Milli 81] and is confirmed by the present study.

(b) Most assertions were based on worst case conditions. However, many errors did not cause the worst case conditions to be exceeded.

(c) Some of the assertions only checked the maximum value. However, in case of some errors the maximum value achieved by the variables, during a certain time frame, was much less than the correct value. It was not possible to check for the minimum value because the correct minimum value of variables is zero most of the time. One solution is to make time a parameter of assertions. Then the values of variable can be sampled at particular times and checked to be within a maximum and minimum range.

4.2 TESTING - PHASE TWO

In order to improve the error coverage the assertions were inserted in every module. Some examples of assertions in each of the modules are as follows:

```
(1) HDG_SEL:
    (a) IF ABS(hdg_error * tas) ≥ 0.02442 then
            ABS(lat_lim_cmd) = 0.5

(2) LAT_LIMITER:
    (a) Time for lat_cpl_cmd to reach maximum
        lies between 5.5 and 6.5 seconds
    (b) IF ABS(lat_cpl_cmd) is decreasing then
        (i)  ABS(hdg_error) ≤ Constant
        (ii) ABS(0.055556 * lat_cpl_cmd) =
             (0.155556 - 0.2222 * krtas) *
             (sel_hdg - 0.736667*yaw_rate - hdg)

(3) A_LAT_COUPL:
    (a) Maximum value of lat_inn_cmd ≤ maximum
        possible value of lat_cpl_cmd
    (b) Time for lat_inn_cmd to reach maximum
        lies between 6 and 9 seconds
    (c) lat_inn_cmd, lat_cpl_cmd and lat_lim_cmd
        must all be reset to zero

(4) LAT_INNER:
    (a) lat_inn_cmd = 0.5 *
          (rl5+0.764*roll+0.1525*roll_rate)
    (b) ABS(rl7) ≤ 0.032
    (c) Time for DELA_CMD to reach maximum
        lies between 2.5 and 6 seconds
    (d) ABS(dela_cmd) ≤ 0.13
```

The software was seeded with errors, one at a time, and executed to see how many of the seeded errors cause assertion violations. Error types and frequencies were similar to those in the NASA AMES data base of errors. The insertion of errors was done independently from the writing of assertions. The results of the experiment are given in Table 2.

Table 2 Preliminary Experimental Results

ERROR TYPE	ERRORS INSERTED	% ERRORS DETECTED	
		PARTIALLY ASSERTED	FULLY ASSERTED
DATA HANDLING	22	63.6	90.9
LOGIC	19	47.3	89.5
DATABASE	19	78.9	94.7
COMPUTATIONAL	21	76.1	85.7
TOTAL	81	66.6	90.1

Currently, the software is only partially asserted, that is, the current assertions only check for the errors in the software which is executed during the heading select mode. It can be seen that 66 % of all the errors inserted in the partially asserted software were detected. Some of the reasons for undetected errors are as follows:

(a) The default value assigned to the variables by the compiler was the same as the initial value of the variables. So the error caused by deleting the initialization statement was not detected.

(b) In the case of some boolean statements the final result was independent of the value of some variables. Any error in the value of those variables could not have been detected.

(c) Some of the errors were in a section of code which was not executed during this phase of testing.

(d) Some errors changed the name of one boolean variable into another. However, since the value of both variables was the same, the error was not detected.

(e) Some errors simulated the condition of a multiple sensor failure. Such errors could not have been detected.

The error coverage can be increased to more than 90 % by fully asserting the software, that is, by writing assertions for all of the flight modes. The current assertions only check for the errors in the software which is executed during the heading select mode. Errors in the software which have no effect on the results are redundant. However, these errors would be caught by a different set of assertions, written specifically to check that particular flight mode. Currently, assertions are being written for two other modes: altitude select mode and autoland mode. It is believed that the use of these assertions would increase the error coverage to more than 90 %. More extensive assertion testing of flight software will provide more definitive results.

5 CONCLUSIONS

Executable assertions can be used for detecting errors throughout the lifecycle of software. They can be written using the information provided in the specifications. Sometimes the writing of assertions is not easy, but it can help increase the reliability of software. The use of assertions forces programmers to explicitly write their assumptions and goals, thereby not only providing good documentation but also increasing their own understanding of the problem. Preliminary experimental results show that assertions can detect more than 66 % of the errors. The error coverage can be increased to more than 90 % by using different set of assertions for different flight modes. This also reduces the complexity of individual assertions. In order to get high error coverage it is important to place assertions intelligently. Instead of using a few complex assertions many simple assertions must be used. It must be pointed out that the use of assertions by itself does not solve the problem of test data generation. It provides the means for checking the output, once appropriate inputs are applied. However, the use of excutable assertions combined with other testing techniques results in a very good and efficient testing methodology.

ACKNOWLEDGEMENTS

The authors would like to thank Mr. Roger Tapper of Rockwell International (Collins Avionics), without whose help this experiment would not have been possible. Thanks are also due to Jim Saito and Doug Doane (NASA-AMES) and to the members of Center for Reliable Computing (Stanford University), especially Ms. Lydia Christopher, for their comments and suggestions.

Partial support for this work was provided by the NASA-AMES under Contract No. NAG 2-246.

REFERENCES

[Adrion 82] Adrion, W. R., M. A. Branstad, and J. C. Cherniavsky, "Validation, Verification, and Testing of Computer Software," ACM Computing Surveys, Vol. 14, No. 2, pp. 159-192, June 1982.

[Andrews 78] Andrews, D. M., "Software Fault Tolerance through Executable Assertions," Conference Record, 12th Asilomar Conference on Circuits, Systems and Computers, pp. 641-645, Pacific Grove, California, November 6-8, 1978.

[Andrews 79] Andrews, D. M., "Using Executable Assertions for Testing and Fault Tolerance," Digest, 9th Annual International Symposium on Fault Tolerant Computing (FTCS-9), pp. 102-105, Madison, Wisconsin, June 20-22, 1979.

[Andrews 81] Andrews, D. M., and J. P. Benson, "An Automated Program Testing Methodology and its Implementation," Proceedings of 5th International Conference on Software Engineering, pp. 254-261, San Diego, California, March 9-12, 1981.

[Bendixen 83] Bendixen, G. E., "The Digital Flight Control and Active Control Systems on the L-1011," Proceedings, IEEE/AIAA, 5th Digital Avionics Systems Conference, pp. 11.2.1-11.2.11, Seattle, Washington, October 31-November 3, 1983.

[Carter 79] Carter, W. C., "Fault detection and Recovery Algorithm for Fault Tolerant Systems," EURO IFIP 79, pp. 725-739, North Holland Publishing Company, 1979.

[DeFeo 82] DeFeo, P., D. Doane, and J. Saito, "An Integrated User - Oriented Laboratory for Verification of Digital Flight Control Systems - Features and Capability," NASA Technical Memorandum 84276, Ames Research Center, Moffett Field, California, 94035, August 1982.

[DFCR-96 80] L-1011 DAFCS Software Description, DFCR-96R1, L-1011 Digital Flight Control System Report, Collins Avionics Division, Rockwell International, 1980.

[Floyd 67] Floyd, R. W., "Assigning Meaning to Programs," Proceedings Symposia in Applied Mathematics, Vol. 19, pp. 19-32, American Mathematics Society, Providence, Rhode Island, 1967.

[Glass 80] Glass, R. L., "A Benefit Analysis of Some Software Reliability Methodologies," ACM SIGSOFT, Software Engineering Notes, Vol. 5, No. 2, pp. 26-33, April 1980.

[Hecht 76] Hecht, H., "Fault-Tolerant Software for Real-Time Applications," ACM Computing Surveys, Vol. 8, No. 4, pp. 391-407, December 1976.

[Hoare 69] Hoare, C. A. R., "An Axiomatic Basis of Computer Programming," Comm. ACM, Vol. 12, No. 10, pp. 576-580, October 1969.

[Horning 74] Horning, J. J., "A Program Structure for Error Detection and Recovery," Lecture Notes in Computer Science, Vol. 16, pp. 171-187, Springer Verlag, New York, New York, 1974.

[King 76] King, J. C., and S. L. Hantler, "An Introduction to Proving the Correctness of Programs," ACM Computing Surveys, Vol. 8, No. 3, pp. 331-353, September 1976.

[Luckham 75] Luckham, D. C., and F. W. von Henke, " A Methodology for Verifying Programs," Proceedings International Conference on Reliable Software, pp. 156-164, Los Angeles, California, April 21-23, 1975.

[Mahmood 83] Mahmood, A., D. J. Lu, and E. J. McCluskey, "Concurrent Fault Detection using a Watchdog Processor and Assertions," Proceedings, 1983 International Test Conference, pp. 622-628, Philadelphia, Pennsylvania, October 18-20, 1983.

[Manna 69] Manna, Z., "The Correctness of Programs," Journal of Computer and System Sciences, Vol. 3, No. 2, pp. 119-127, May 1969.

[Milli 81] Milli, A., G. Metze, "Self-Checking Programs: An Axiomatization of Program Validation by Executable Assertions," Digest, 11th Annual International Symposium on Fault Tolerant Computing (FTCS-11), pp. 118-120, Portland, Maine, June 24-26, 1981.

[Ramamoorthy 75] Ramamoorthy, C. V., and S. F. Ho, "Testing Large Software with Automated Software Evaluation System," Proceedings of International Conference on Reliable Software, pp. 382-394, Los Angeles, California, April 21-23, 1975.

[Randell 75] Randell, B., "System Structure for Software Fault Tolerance," IEEE Trans. on Software Engineering, Vol. SE-1, No. 2, pp. 220-232, June 1975.

[Saib 77] Saib, S. H., "Executable Assertions - An Aid to Reliable Software," Conference Record, 11th Asilomar Conference on Circuits, Systems and Computers, pp. 277-281, Pacific Grove, California, November 7-9, 1977.

[Stucki 75] Stucki, L. G., G. L. Foshee, "New Assertion Concepts for Self Metric Software Validation," Proceedings of International Conference on Reliable Software, pp. 59-71, Los Angeles, California, April 21-23, 1975.

SESSION 12 ROTORCRAFT AVIONICS

Chairmen:

Dallas G. Denery
NASA Ames Research Center

George Stech
Avionics Research and Development Activity

This session focuses on advances in avionics integration and control, and navigation technologies that allow rotorcraft and VSTOL aircraft to take full advantage of their unique capabilities.

84-2672

INTEGRATED AVIONICS FOR ADVANCED ARMY ROTORCRAFT

W.L. Eversole, W.F. Kiczuk, J.A. Lambrecht, R.L. Rivard*,
J.J. Williams, and R.E. Branstetter

Texas Instruments Incorporated
P.O. Box 660246 M/S 3105
Dallas, Texas 75266
(214) 995-1941

Abstract

An approach to the development of an advanced avionics architecture to meet the functional requirements of the Army's next generation of rotorcraft (LHX) is presented. Mission requirements are briefly discussed to identify the functional partitioning of the LHX system. This is followed by a brief overview of each subsystem. The requirements of the processing algorithms are discussed to quantitatively identify the performance requirements of the avionics architecture. Also described is a multiprocessor, multisensor architecture involving high bandwidth sensors, low bandwidth sensors and a hierarchy of processing structures and interconnections that provides the flexibility, reliability, availability and fault tolerance within the power, volume, and weight constraints imposed by LHX.

Introduction

The many missions defined for the Army's next generation of rotorcraft represented by the Light Helicopter (LHX) family dictate the design of an avionics system that can operate in both offensive and defensive roles against both ground and air targets using weapons appropriate to the target and mission. In the past, avionics systems have been designed using the self-contained, black-box approach in which each sensor in the system and its associated processing are designed to interconnect to the system bus without regard to the other sensors or processors. This horizontally partitioned architecture severely limits processing resource sharing because processors of different sensors cannot share input data. This approach is convenient for the system integrator and the procurement agency but results in an inefficient overall system design in terms of operator workload and mission parameters. Provisions for degraded system performance, fault tolerance, availability reliability, and survivability are difficult to incorporate without massive physical redundancy techniques. Also, since the operator is required to provide the information-blending function, high workload periods are created that significantly reduce the probability of mission success.

An alternative to the black box design is a vertically partitioned system that contains the processing and communications resources to meet the requirements of the various LHX subsystems. This integrated architecture must support the resource sharing of processors and be constructed from a set of common processing modules that specifically support a 2-level maintenance concept. This advanced integrated avionics system design requires integrated circuits, processor modules, multiprocessor interconnection schemes, structured software control, and system support tools to be melded into an affordable system that can be operated by a single person. This is a formidable problem that requires a comprehensive understanding of system objectives and requirements, a thorough knowledge of digital signal data and information processing, an understanding of the practicalities imposed by the system operational environment, and full appreciation of the state-of-the-art for digital processing components and architectures.

The design of any complex system requires a methodical, iterative approach that is tops-down and hierarchical in nature to ensure a modular design in which the lowest hierarchical elements are autonomous.

The following sections briefly describe the design approach utilized in defining the integrated avionics system for LHX. The mission requirements that provide a functional partitioning of the LHX system, the processing requirements of the avionics subsystem, and the integrated architecture required to implement these processing requirements are discussed. While the mission and processing requirements are unique to LHX, the architecture design methodology is generic. The processing architecture and modules are flexible and modular and thus are applicable to a wide range of systems.

Mission Analysis

An understanding of the LHX missions and their influence on the functional partitioning of the LHX architecture is necessary to ensure validity of the overall architecture structure. By examining the missions identified for LHX and evaluating threats, environmental conditions, and types of targets, the design-driving requirements can be defined. The requirement that the LHX aircraft engage a variety of targets and defend itself against both ground and air threats makes target acquisition and multicapable weapons the major driving requirements, followed closely by navigation in day/night and adverse weather and by communications. To give LHX this capability, a mission equipment package (MEP) is required that provides nap-of-the-earth piloting; short target-acquisition time lines; multiple radio communications; and automated, accurate navigation.

The MEP must be evaluated in terms of cost, weight and performance. The avionics functions can be

* Now with Thermotron Industries Inc., Holland, Michigan

defined in qualitative terms based on the LHX mission requirements. However, translating qualitative functions into quantitative requirements is necessary before the avionics system can be defined. The quantitative requirements must be determined from mission requirements to allow a "mission-driven" rather than a "technology-driven" processing architecture.

LHX Subsystems

The LHX system consists of nine functional subsystems as shown in Figure 1. The processing required by these subsystems, coupled with the stringent physical constraints of an advanced rotorcraft, dictates an integrated architecture approach that crosses functional boundaries to permit the sharing of processing resources.

Table 1 outlines the functions performed by each subsystem, and Table 2 presents estimates of required program memory, data memory, and processing throughput. Although these processing requirements depend heavily on the algorithmic approach selected for implementation and will vary with the mission equipment package, they do provide a quantitative baseline for defining the avionics architecture for LHX.

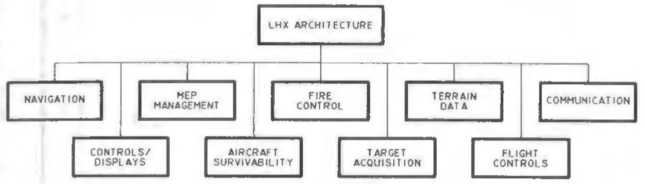


Figure 1. LHX Functional Partitioning

LHX Processing Architecture

Figure 2 shows a processing architecture configured to efficiently utilize the signal and data processing resources required to execute the algorithms of the various LHX subsystems. This architecture supports the resource-sharing of processors and can be constructed from a set of common processing modules. A high degree of system availability and reliability is provided through the application of spare processing resources at the system level: spares can provide backup in case the primary resources fail. The architecture also supports graceful degradation: when spare resources are exhausted, remaining resources can be assigned to the highest priority functions on a mission basis. This processing architecture implements the sensor fusion algorithms that combine the data of multiple subsystems to create a highly synergistic system; however, the control structure also allows autonomous operation of the subsystems while maintaining deterministic processing time lines.

In developing the processing architecture, desired system attributes of flexibility, modularity, fault tolerance, and processor independence impose requirements upon the interconnection topology while the ability to solve a wide range of problems places requirements upon the processing module design. Table 3 lists the desired system attributes of LHX

TABLE 1. TOP-LEVEL AVIONICS SYSTEM FUNCTIONS

AVIONICS SYSTEM FUNCTION	FUNCTION ATTRIBUTES
MEP MANAGEMENT	AUTOMATE HIGH WORK LOAD FUNCTIONS; MONITOR SYSTEM FAULT STATUS; EASY INTERACTIVE INTERFACE TO THE PILOT
TARGET ACQUISITION SYSTEM	AUTOMATIC DETECTION, RECOGNITION, IDENTIFICATION, AND LOCALIZATION; AUTOMATIC PRIORITIZATION, CUEING, WEAPON HANDOFF; MULTIPLE TARGET TRACKING; DAY, NIGHT, ADVERSE WEATHER CAPABILITY, ACTIVE SENSORS LOW PROBABILITY OF INTERCEPT (LPI) WAVEFORMS; GENERATE FIRE CONTROL TRACKING DATA
NAVIGATION	ABSOLUTE AND RELATIVE 3-D POSITION; THE AIRCRAFT ATTITUDE; COMPUTER-GENERATED MAP; BULK LOADING AND RETRIEVAL OF NAVIGATION MISSION DATA; AIRCRAFT AIR AND GROUND SPEED; APPROACH AND LANDING CAPABILITY; PATH FINDING AND TA/OA IN NIGHT AND POOR VISIBILITY
FLIGHT CONTROL	NOE, BOB-UP, LOW LEVEL, AND CONTOUR FLIGHT CONTROL AUTO UNMASK AND REMASK; AUTO RETURN HOME
AIRCRAFT SURVIVABILITY	PASSIVELY SEARCH, DETECT, AND LOCALIZE RADAR AND EO THREATS; PASSIVELY IDENTIFY LETHAL AND NONLETHAL THREATS; PASSIVELY DETECT AND WARN PILOT IN TIME TO TAKE EVASIVE ACTION TO AVOID NUCLEAR, BIOLOGICAL, OR CHEMICAL DANGERS; AUTOMATICALLY INITIATE RF, IR, AND OPTICAL JAMMERS
WEAPONS AND FIRE CONTROL	SCAT: CONTROL FIRING AND GENERATE FIRE CONTROL SOLUTIONS FOR WEAPONS CAPABLE OF NEUTRALIZING A/A AND A/G TARGETS (ARMORED AND SOFT) IN CLEAR AND ADVERSE WEATHER; HANDLE MULTIPLE TARGETS
	UTILITY: CONTROL FIRING AND GENERATE FIRE CONTROL SOLUTIONS FOR A/A AND CLOSE-IN PROTECTION
COMMUNICATIONS	BAND COVERAGES: HF, VHF, UHF, L; SIMULTANEOUS COMMUNICATION VOICE—ALL BANDS; DIGITAL DATA—ALL BANDS; DF—ALL BANDS; IFF NONCOOPERATIVE AND PASSIVE; SECURE, JAM RESISTANT, TEMPEST-QUALIFIED—ALL BANDS; AUTO TARGET HANDOFF; LOW PROBABILITY OF INTERCEPT (LPI)
CONTROL/DISPLAYS	PROVIDES CONTACT BETWEEN THE OPERATOR AND THE AIRCRAFT
TERRAIN DATA	COMPUTER-GENERATED MAP TO PERFORM TERRAIN CORRELATION NAVIGATION AND AID TF/TA AND MISSION PLANNING

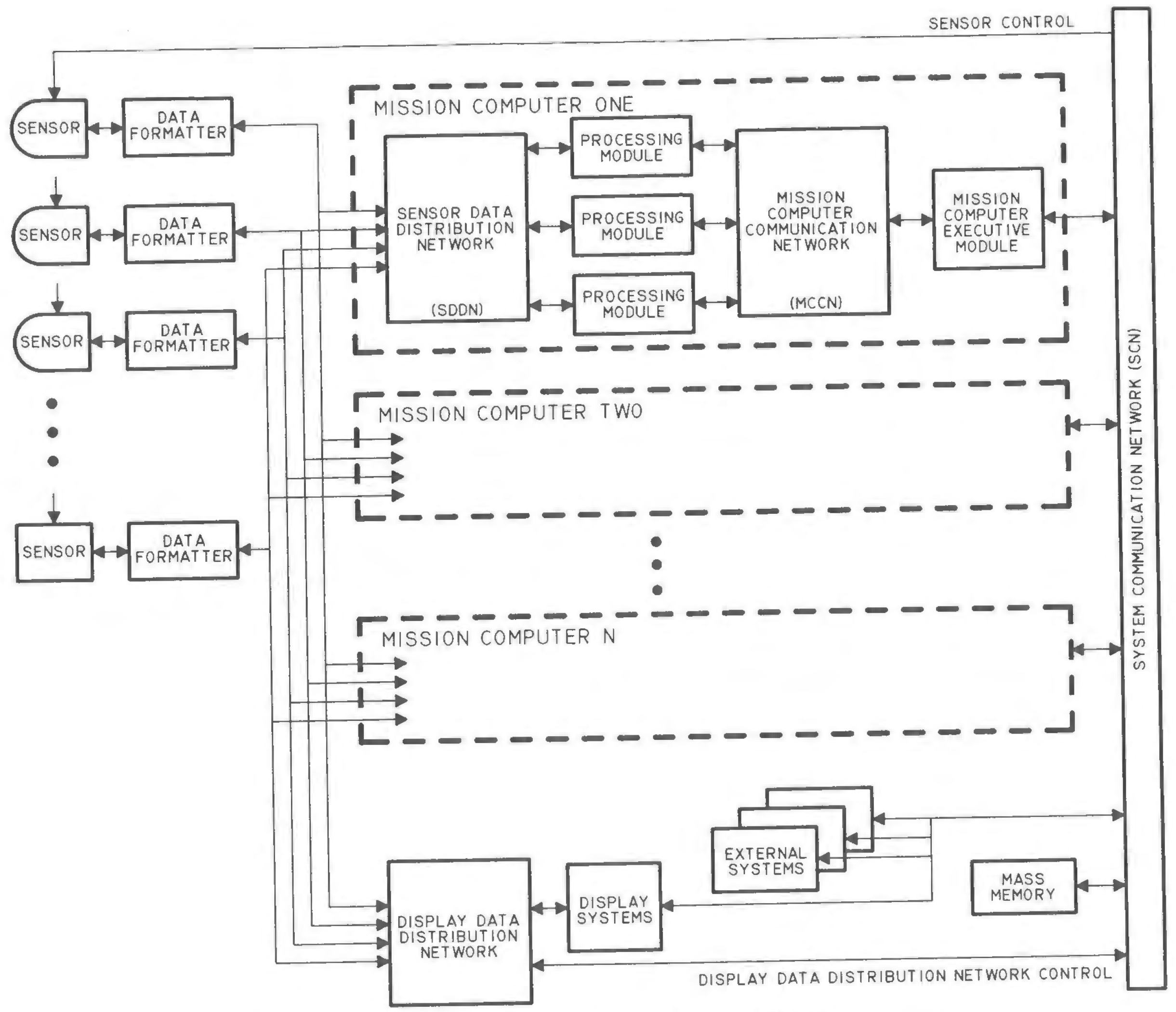


Figure 2. Integrated Processing Architecture

along with the requirements and how the LHX processing architecture meets these requirements. The LHX avionics architecture incorporates these attributes into three main objectives: modularity for incremental changes in system capability and reliability/ maintainability; fault tolerance to permit reconfiguration and degraded modes of operations; and a hierarchical control structure for ease of programming.

The avionics processing is distributed among the multiple-mission computers. Each mission computer receives input data from the sensor suite, processes this sensor information, and communicates with the other mission computers via the system communications network. The sensor suite, which consists of the sensors and data formatters, sources the input data for the mission computers, formats this data, and distributes it to each mission computer's sensor data distribution network (SDDN) and the control/display subsystem.

The sensor suite accommodates a wide variety of sensensors (e.g., radar, FLIR, laser rangefinder, and RF front ends of the communications subsystems). To provide maximum resource-sharing, as much processing as possible (including sensor signal conditioning/processing) should reside in an integrated system. Generally, however, all analog sensor signal conditioning, including analog-to-digital conversion, must be placed as close to the sensor as possible to reduce the introduction of extraneous noise into the sensor signal. Additionally, to effectively provide resource-sharing between the mission computer processors and the sensors, all sensor data must be presented to the mission computers in a common format. This requirement drives additional processing into the sensors in the form of data formatters to convert each sensor's data into a common format for transfer to the mission computer processing modules. The amount of processing which the data formatter must provide varies greatly among sensors. For sensors that require unique processing resources that cannot be used by other subsystems, the processing is within the data formatter; otherwise, the processing resources must be duplicated in each mission computer to maintain identical mission computers.

The mission computers consist of an SDDN, processing modules, modular power supplies, and an internal communications network. The processing modules utilize VHSIC technology and consist of multiple processing elements, memory, and interfaces. The mission computer receives data from either the sensors (through the SDDN) or the system communications network (through the executive module's system communications network interface). Each mis-

TABLE 2. LHX PROCESSING REQUIREMENTS

FUNCTION	PREPROCESSING (MOPS)	ARRAY PROCESSING (MOPS)	DATA PROCESSING (MIPS)	PROGRAM MEMORY (KWDS)	DATA MEMORY (MBITS)
TARGET ACQUISITION	978	125	20	150	33
FIRE CONTROL	—	—	0.75	139	1.5
ASE	70	2	5	40	0.2
TERRAIN DATA	—	343	7.5	120	81
NAVIGATION	—	10	9	96	4
COMMUNICATION	—	138	3.5	127	4
MEP MANAGEMENT	—	0	24	148	27
FLIGHT CONTROL (3)	—	0	2.3	48	—
TOTAL	1,048	618	72	868	151

TABLE 3. LHX ARCHITECTURE REQUIREMENTS

ATTRIBUTE	REQUIREMENT	IMPLEMENTATION
INTERCONNECTION		
FLEXIBILITY—ABILITY TO ADAPT TO MODE REQUIREMENTS	MULTIPLE DATA/CONTROL PATHS	HIERARCHY OF DATA HIERARCHY OF CONTROL
FAULT TOLERANCE—DETECT, LOCALIZE, EXCISE FAULTS AND RECONFIGURE RESOURCES	REDUNDANCY FAULT MONITORING	SPARES (PHYSICAL AND FUNCTIONAL) FAULT MONITORING CIRCUITRY
MODULARITY—EASE OF PROGRAMMING AND CONTROL OF PROCESSOR	UNIPROCESSOR ENVIRONMENT DISTRIBUTED CONTROL	ISOLATE PROCESSORS INTO MODULES DISTRIBUTED SYSTEM CONTROL
MONITORABILITY—REAL-TIME SYSTEM STATUS FOR DEBUG	CHIP LEVEL TESTING PROCESSOR LEVEL TESTING	SYSTEM MONITOR SYSTEM MAINTENANCE BUS (SMBUS)
PROCESSOR INDEPENDENCE—USE A BROAD CLASS OF PROCESSORS	PROCESSOR SLAVE TO MASTER CONTROL I/O AND CONTROL REMOVED FROM PROCESSORS	MODULE CONTROLLER
PROCESSOR		
ABILITY TO SOLVE WIDE RANGE OF PROBLEMS	HIGH-THROUGHPUT REAL PROCESSING HIGH-THROUGHPUT COMPLEX PROCESSING DATA-DEPENDENT PROCESSING STORE DATA/PROGRAMS	ARRAY PROCESSOR COMPLEX PROCESSOR DATA PROCESSOR MEMORY

sion computer's SDDN routes the sensor data to the appropriate sensor interface. The network is capable of rerouting sensor data to multiple destinations; however, the rerouting of data is not a dynamic clock-by-clock capability. The SDDN is capable of rerouting data paths within the acceptable reconfiguration time for the system.

Video is routed from the sensors to the control/display subsystem for display in the cockpit. The interface to the control/display subsystem is similar to the SDDN. The display data distribution network contains input ports for each sensor that may be displayed and an output port for each display in the cockpit. The appropriate input port is connected to the appropriate output port through a switching configuration. The system level controller controls the display data distribution network via the system communications network.

Mass storage provides a central storage/program downloading facility for LHX. Mass storage, accessed via the system communications network, contains a smart interface, RAM, ROM, and a program downloading device.

The number of mission computers in the system is determined by the functions to be performed, physical constraints, and reliability/maintainability considerations. The functions determine the number and size of mission computers by dictating how many processing modules need to be grouped in a single mission computer. (If functions that require a high data rate between them are distributed among different mission computers, communications will not be as efficient and the bandwidth of the system communications network may be exceeded.)

Physical constraints dictate the maximum mission computer, module, and system size. Processing requirements and system size (power and weight) constraints dictate the maximum number of mission com-

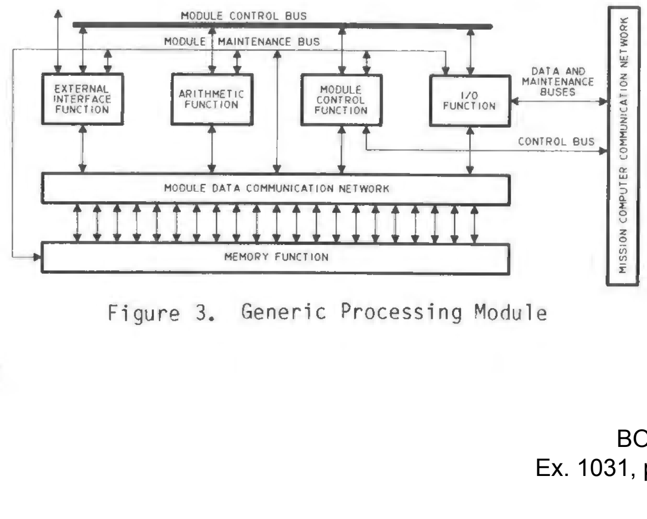

MODULE CONTROL BUS
MODULE MAINTENANCE BUS
EXTERNAL INTERFACE FUNCTION
ARITHMETIC FUNCTION
MODULE CONTROL FUNCTION
I/O FUNCTION
DATA AND MAINTENANCE BUSES
CONTROL BUS
MODULE DATA COMMUNICATION NETWORK
MEMORY FUNCTION
MISSION COMPUTER COMMUNICATION NETWORK

Figure 3. Generic Processing Module

puters; the greater the number of mission computers, the more overhead in the system. To achieve an acceptable level of fault tolerance, multiple mission computers must exist in a large system. For reconfiguration and maintainability considerations, identical mission computers are desirable.

The LHX architecture contains four levels of interconnect networks: the SDDN interconnects the sensor suite and the mission computer; the Module Communication Network (MCN) is internal to each module and interconnects the resources that make up a module; the Mission Computer Communication Network (MCCN) is internal to each mission computer and interconnects its modules; and the System Communication Network (SCN) interconnects the mission computers and other LHX subsystems. This hierarchical interconnect structure is necessary because the interconnect scheme at each level possesses different characteristics. The optimum interconnect structure has been implemented at each level to provide LHX with efficient communications and data transfers.

The LHX control scheme is also hierarchical in nature with system level, mission-computer level, and module-level tasks. Upon system initialization, one of the mission computer executives (the primary mission executive) assumes the system control functions and coordinates boot-up of the system. This primary mission computer executive loads and distributes all data needed from mass storage and coordinates the execution of system-level test functions. In addition, the primary mission computer executive and all other mission computer executives are responsible for executing mission computer-level tests and downloading software into the mission computer processing modules. System-level control ensures coordination between mission computers and provides data fusion processing from multiple mission computers. All mission computer executive modules are able to perform the system control tasks in case of failure. To enable the backup mission computer executives to assume the role of system controller, all information needed to perform the system control tasks is available to all mission computer executives.

A high degree of fault tolerance is needed for the system-level tasks. This is achieved by placing critical data in all mission computer data bases, executing critical tasks in all mission computers, and voting on the results.

The mission computer executives are responsible for executing the mission computer-level control functions that enable the mission computers to operate autonomously. A mission computer data base maintained by each mission computer executive module contains both volatile and nonvolatile memory. This data base is used for program storage, shared data, and configuration tables. Also stored in each mission computer data base is the system status information needed to enable the module to smoothly transition into the role of system controller, if needed. To permit the mission computers to be interchangeable, a portion of each mission computer data base contains identical information.

The mission computer executive module coordinates processing within the mission computer by setting up each module with the information needed to perform the tasks, triggering the start of certain tasks, and receiving status information from the modules. Although a degree of autonomous processing is allowed under the control structure, a deterministic processing flow is maintained by sending status to the executive and receiving triggers back.

To perform fault monitor functions, the executive receives fault reports via the mission computer maintenance bus and generates the appropriate response. Examples of these responses are: run self-test; retransmit data; and reconfigure the system.

Control is distributed to the module level to achieve a high degree of fault tolerance and reduce communications overhead. Module-level control functions are executed by the module's control function. Internally, the module controller is responsible for monitoring and coordinating the module's resources. Externally, the module controller responds to processing requests, which are made in either of two ways: (1) data flow is implemented whereby the arrival of data and parameters triggers the execution of a task; or (2) control flow is implemented whereby a control signal from the mission computer-level control function triggers the execution of a task. Data flow enables distributed, autonomous processing at the lower levels of the hierarchy, thereby eliminating the need for a complex centralized system controller and permitting a highly modular software structure. A control-flow structure is implemented to service the data-flow graphs, synchronize task execution where necessary, and ensure that latency requirements are met. This hybrid control/data-flow structure enables the system designer to control execution within the system in an optimum manner.

Fault Tolerance

The integrated architecture proposed for LHX uses redundant busing, spare and redundant common modules, self-testing hardware, and built-in test to achieve fault tolerance. The processing required to achieve a fault-tolerant system is implemented at all architectural and control levels: at the module level, LHX monitors system status by taking advantage of the self-test capability of the VHSIC chips; at the mission computer level, multilevel testing is used to detect and isolate faults; at both the system and mission computer levels, the flexibility inherent in the interconnect scheme is used to improve fault tolerance. Comprehensive testing of all resources is initiated upon system initialization, thus ensuring system integrity. On completion of the test procedure, the pilot is notified of system status.

Multiple reconfiguration levels are supported to ensure continued operation with minimum impact on the probability of mission success. The appropriate level of reconfiguration is determined by task priority, its relationship to mission success/failure, and its corresponding maximum allowable fault recovery time. Reconfiguration levels include the capability to execute a function in an alternate module, the capability to perform a mission computer's functions in another identical mission computer, and the capability to provide degraded operational modes.

The mission computer executives are responsible for collecting fault reports from the module's fault monitoring functions, logging the faults, and determining the correct response. The mission computer executives also monitor each other to ensure correct operation.

The module's fault-monitoring function is performed by the module control function and is responsible for all fault detection and localization at the module level. The module's fault monitor works with the module controller injecting and evaluating test data. Testing is performed both as a low-priority task (executed while the processor is idle) and at regular intervals (scheduled with other tasks) or upon request from the mission computer executive.

The self-test features of the VHSIC chips are exploited for fault detection during normal operation. Test stimulus generation (or storage), test results, data compression, final data comparison, test decisions, and control are built into many of the VHSIC chips. If the self-test fails, a failure is detected and isolated to the chip. The on-chip self-test is run at the system clock rate; therefore, timing-related features can also be detected.

In addition to the self-test, other processing is implemented to provide comprehensive fault detection. Data transmissions are checked with parity, message-length count checking, and word-length count checking for serial transmissions. Data validity checks are also performed. Test patterns are run through the system and checked to ensure proper operation.

The required level of testing, redundancy, and reconfiguration varies for each function. The more critical functions require more processing to ensure fault tolerance. Here, voting techniques and a high degree of redundancy are used. Less critical functions may be checked with techniques that do not require as high a degree of overhead while still ensuring an acceptable level of fault tolerance. Parity checking and periodic self-testing are examples of testing with a minimum amount of overhead and are implemented throughout the system.

Critical data is protected by providing battery back-up to the memory that stores the data. This ensures that the system will continue to operate at an acceptable level if power is interrupted momentarily.

Processing Modules

Based on the investigation of the processing requirements of the various LHX subsystems, five types of processing modules, plus a power supply module, are required to support LHX requirements: an array processing module to perform relatively data-independent processing on real, repetitive, generally fixed-size blocks of data; a complex vector processing module to perform relatively data-independent processing on complex, repetitive, generally fixed-size blocks of data; a data processing module structured for data-dependent decision-making and scalar processing; a mission computer executive module to control processing coordination within the mission computer and to provide interfaces to external subsystems; and a memory module to provide bulk data and program storage. Other special-purpose processors may need to be developed to solve specific problems with less hardware overhead. Additions to this set are expected to include new modules as well as higher- or lower-performance versions of the defined modules. For generality, these modules have not been addressed.

All modules have been designed to adhere to the generic model shown in Figure 3. This ensures compatibility and reduces hardware and software overhead resulting from unique implementations. The generic module consists of six major sections: a module communications network; a module controller; an MCCN interface; an external interface; an arithmetic function; and memory. Specific functional modules are constructed by adding arithmetic, memory, and external interface functions needed to form the specific module.

The module communications network interconnects the functional processors that make up the module, transmitting data, control, and maintenance information. The size of the data communications network depends on the memory configuration required to support the modules's processing requirements.

All modules contain a 1750A control processor that executes an operating system and applications programs. In addition to controlling the other functional processors in the module, the controller performs scalar processing. The ability to supplement the capabilities of the module and the rest of the system with a scalar processing function is essential to system efficiency.

The module operating system is written in Ada to the maximum extent feasible and supports an embedded real-time multitasking, multiprocessing, interprocess/intertask communications system. Functions of the module operating system are module control support, task management, time management, interrupt management, I/O management, exception handling, memory management, fault tolerance, debug support, and Ada operating system support. The mission computer communications network interface provides a bus structure internal to the mission computer for data transfers between modules. Data, control, and maintenance information must be transferred.

The external interface varies for each module. A sensor interface is provided on the array, complex vector, and data processing modules. A system communications network interface is provided on the mission computer executive module. The memory modules have no external interface.

The arithmetic function is unique to each processor. It is designed to perform the processing required of of the module. For the array processing module, the arithmetic function is implemented as a high-speed array processor; for the data processor, the arithmetic function is implemented as additional 1750A scalar processors.

Each module contains a memory function composed of multiple memory banks that interface to memory ports on the module's other functional processors through the module data communications network. Each module may require a unique memory configuration to support the processing it must perform.

A power-supply module is also needed to provide the power required for each mission computer. The use of small modular power supplies complements the reconfigurable integrated processing architecture.

Although versions of some of these modules are being defined under the DoD VHSIC program, they may not be compatible with the LHX architecture because they lack form, fit, and function standardizations. To achieve the performance required of the LHX architecture, the processing modules must be defined within the confines of the total integrated system. This design approach will ensure the optimum architecture/processor definition.

Summary

Avionics processing equipment has been traditionally designed for specific subsystems in which it was intended to operate. This approach has led to different architectures, hardware, and software supported within the same system. A low-cost, sustainable, mission-effective avionics system requires fault-tolerant, modular processing architectures and processing modules. Essential in the development of this advanced integrated architecture and processing modules is the timely and cost-effective insertion of very-high-speed integrated circuit (VHSIC) technology.

Using a tops-down, mission-driven architecture design methodology that couples mission requirements, system attributes and processing requirements to interconnection topologies, reconfigurability, and control issues of processor architectures, an integrated avionics architecture has been defined. The LHX processing modules are grouped into multiple, identical mission computers that share access to the sensor suite and system communications network. Control is hierarchical, with system-level, mission computer level, and module-level tasks. Tasks are decentralized as much as possible, resulting in a hierarchical structure in which control is largely distributed to the lower levels. The LHX architecture uses redundant busing, spare and redundant common modules, self-testing hardware, and built-in test to achieve fault-tolerance. The processing required to achieve a fault tolerant system is implemented at all architectural and control levels: at the module level, system status is monitored by taking advantage of the self-test capability of the VHSIC chips; at the mission computer level, multilevel testing is used to detect and isolate faults; at both the system and mission computer levels, the flexibility inherent in the interconnect scheme is used to improve fault tolerance.

The mission computers consist of a sensor data distribution network, VHSIC processing modules, and an internal communications network. They interface to the other sections of the LHX avionics system via two communications (control and data) networks. The sensor data distribution network provides input data from the sensor suite through data formatters (used to convert each sensor's data into a common format) directly to the processing modules. A system communications network interfaces the mission computers to each other and to the other subsystems.

The processing modules are grouped into identical mission computers, thereby reducing system maintainability costs and providing a high degree of flexibility for system configurations. All processing modules are defined as a board(s) controlled by a single controller and sharing I/O to the mission computer communications network. To ensure that all the modules defined are compatible in a system, certain features are common to all modules --- common data, control, and maintenance interfaces and a control function containing the module's Ada compiled operating sysem.

Five modules have been defined to construct a general signal processing architecture: a mission computer executive module; a data processing module; a complex vector processing module; an array processing module; and a global memory module. Other special-purpose processors may need to be developed to solve specific problems with less hardware overhead. The defined architecture and processing modules are combined to create a highly modular, efficient, integrated avionics system that can provide substantial improvements in availability and mission effectiveness for the Army's family of light helicopters.

EVALUATION OF A REAL-TIME PREDICTIVE GUIDANCE LAW FOR LANDING VTOL AIRCRAFT AT SEA

84-2673

Clyde H. Paulk, Jr.*

Aerospace Engineer
NASA Ames Research Center
Moffett Field, California

and

Anil V. Phatak†

Senior Engineer
Analytical Mechanics Associates, Inc.
Mountain View, California

Abstract

A piloted simulation evaluation was conducted to assess the merits of a predictive lull/swell guidance law for landing a vertical takeoff and landing aircraft at sea. Two cases were evaluated. The first was performed by the pilot without the aid of the lull/swell guidance indicator, which indicates a landing opportunity. The second was performed in a similar manner with the aid of the guidance indicator. The pilot was instructed to use the guidance indicator only as a landing aid. The results indicated the pilots were able to visually determine ship lulls and swells prior to landing. However, use of the lull/swell guidance resulted in increased pilot confidence in the existence of a landing opportunity, which resulted in significantly shorter hover waiting times prior to landing. The performance of the lull/swell guidance was conservative, in that it forecast the onset of deck lulls and swells later than the pilot could detect them, but with greater reliability and therefore greater pilot confidence. In no instances did the guidance algorithm predict a false ship lull with which the pilot did not agree.

Introduction

Landing a vertical takeoff and landing (VTOL) aircraft aboard a small ship represents a complex operation and a demanding task. Under the most demanding of conditions, the landings must be accomplished at sea and onto a moving shipboard landing pad 40 ft square. Specifically, present fleet capability for landing aboard small ships is limited by low visibility, ship motion, and airwake turbulence. Operational capability is further reduced by the high pilot workload caused by aircraft control and display limitations.

The 15 to 20 sec before aircraft touchdown are critical and involve terminal guidance and control problems, not only because the aircraft is disturbed by the airwake turbulence, but also because the landing pad is always moving. The need for deck-motion prediction for carrier landing operations was pointed out several years ago (1-3), and the influence of ship motions on helicopter operations from small ships (destroyers) was discussed in Ref. 4.

*Member AIAA.
†Member IEEE.

Two categories of deck-motion guidance laws may be formulated. The first is based upon the assumption that ship-motion time histories display a pattern of alternating low-amplitude quiescent periods called "lulls" followed by large-motion interlull segments called "swells." The second is based upon other designs that are independent of the presence of ship-motion pattern structures (i.e., lulls and swells).

Previous studies (2,5,6) of the shipboard landing problem have ignored the existence of ship-motion lulls, and have concentrated on designs that either predict the short-period time history of the ship motion, or have considered a flight-control system or landing controller that forces the aircraft to track or follow the instantaneous motion of the landing pad. The limitation of the time-history prediction techniques has been the inability to accurately predict more than 6 to 10 sec into the future, and the limitation of the landing pad controllers is the high bandwidth requirement of the flight-control system.

A previous study by the authors analytically investigated a letdown guidance law that examined the inherent alternating lull/swell ship-motion patterns and predicted the landing opportunities (7). The objective of the current study was to evalute the guidance algorithm in a real-time, man-in-the-loop simulation. The algorithm is based upon the alternating lull/swell patterns evident in actual ship motions. The algorithm provides a "LAND" or "HOLD" signal displayed to the pilot on a head-up display (HUD) to indicate the initiation and termination of the landing-opportunity window.

This paper discusses the lull/swell landing guidance algorithm, and presents a flow chart of its real-time implementation. In addition, results obtained from a piloted simulation evaluation of the guidance algorithm for landing a helicopter on a destroyer in rough seas are presented.

Lull/Swell Landing Guidance

Ship-Motion Pattern Analyses

Ship-motion response-time histories, as shown in Fig. 1, can be classified into one of two categories or modes: 1) a lull mode representing time intervals during which overall ship motion may be described as relatively low; and 2) a swell mode during which ship motion is relatively high. In this effort, the vertical motion of the landing pad was used to develop a real-time lull/swell classification algorithm.

The concept of a lull is best displayed in Fig. 1, where the top two plots (a) and (b) show the vertical position $z(t)$ and velocity $\dot{z}(t)$ of a sample landing pad over a 100-sec interval. As described in Ref. 7, the lull/swell classification algorithm is characterized by an indicator function $I(z)$ which maps the ship heave position into a binary output sequence of zeros and ones, where $I(z) = 0$ corresponds to a lull period, and $I(z) = 1$ describes an interlull interval or swell.

A simple criterion one might consider for generating $I(z)$ is to dichotomize the motion into lulls and swells by defining an indicator function only in terms of a threshold amplitude, z_T, such that

$$I(z) \equiv I_z \triangleq \begin{cases} 0 \text{ (lull)}: & z(t) \le z_T \\ 1 \text{ (swell)}: & z(t) > z_T \end{cases}$$

The technique is not practical, however, because it does not account for the developing patterns. As a result, the lull/swell switch plots I_z that are obtained using this definition display a high switching frequency (0 to 1 and vice versa), making such a criterion unacceptable from a practical viewpoint. The reason for this behavior is the existence of a relatively large number of single peaks in $z(t)$ that exceed the threshold z_T during a long period when $z(t) \le z_T$ (I_z switches from 0 to 1), or that are below the threshold z_T during a period when $z(t) > z_T$ (I_z switches from 1 to 0). The net result is a large number of lulls and swells of extremely small duration ($\le$3 sec) that are embedded in relatively longer-duration swell and lull intervals.

A more reasonable approach, and the one used in this study, is to define the initiation and termination of a lull in terms of parameters that reflect the motion response characteristics over a finite past or memory interval. Thus, initiation of a lull may be defined by a parameter "NSTART" as follows: A lull is said to begin (i.e., I_z switches from 1 to 0) as NSTART consecutive positive peaks of $z(t)$ fall below a prescribed threshold value z_T. Similarly, termination of a lull (or onset of a swell) may be defined by a parameter "NSTOP" as follows: A lull is said to have terminated (i.e., I_z switches from 0 to 1) when NSTOP consecutive positive peaks of $z(t)$ exceed a prescribed threshold value z_T. Figure 1(c) shows a plot of the indicator function I_z for the lull/swell periods for nominal parameter values of NSTART = 2, NSTOP = 2, and $z_T = 5.04$ ft. The threshold value of z_T was chosen as the mean value of the positive peak-amplitude envelope of $z(t)$. Over the 100-sec segment of data shown in Fig. 1, the interval (602.2 - 644.4) sec is identified as a swell period. In other words, the ongoing lull from $t = 555$ sec is terminated at $t = 602.2$ sec when two consecutive positive peaks of $z(t)$ (i.e., NSTOP = 2) have exceeded the threshold of $z_T = 5.04$ ft (see Fig. 1(a,c)). Similarly, a lull is initiated at $t = 644.4$ sec when two consecutive positive peaks of $z(t)$ (i.e., NSTART = 2) have stayed under the threshold of $z_T = 5.04$ ft (see Fig. 1(a,c)).

Note that NSTOP = 2 results in true swells (i.e., swells which usually last for $>3 \doteq 4$ cycles or approximately $\ge$ 20 sec) being detected one cycle after their actual onset. Selecting NSTOP = 1 would apparently solve this problem by detecting true swells one cycle earlier than with NSTOP = 2. However, this selection would indicate too many additional swells of 6- to 10-sec duration because of isolated single peaks of $z(t)$ exceeding the prescribed threshold z_T.

The termination of lulls can be made one cycle earlier, when the first peak of $z(t)$ exceeds z_T, if at that time one can confidently forecast that the next successive positive peak will also exceed the prescribed threshold. This condition is illustrated by the dotted line in Fig. 1(c), where the effect of being able to forecast one cycle ahead is shown. Assuming that such a forecast can be made, lull termination occurs 5.4 sec earlier, at $t = 596.8$ sec. Unfortunately, a reliable on-line, one-cycle-ahead forecasting method is not available for practical implementation of this approach.

An alternative method for early detection of the onset of a swell, based upon the z versus $\dot{z}$ phase-plane pattern characteristics at lull/swell transitions, was used in this study. Phase-plane plots of $z(t)$ versus $\dot{z}(t)$ for several lull/swell transition segments (as defined by NSTOP = 2 and $z_T = 5.04$ ft) were analyzed to determine whether a pattern or a sequence of events in the z and $\dot{z}$ time histories could be identified and used as a precursor for lull termination prior to the crossing of the threshold z_T by the second positive peak of $z(t)$. Based on this analysis, a phase-plane-based criterion for lull termination that performs nearly as well as the one-peak-ahead, forecasting-based method was developed. According to this criterion, a lull is terminated if successive peak magnitudes of either of the following exceed their prescribed thresholds: 1) vertical position $z(t)$ and the following vertical velocity $\dot{z}(t)$; or 2) vertical velocity $\dot{z}(t)$ and the following vertical position $z(t)$. Thus, $I_{z,\dot{z}}$, the indicator function for this phase-plane approach, switches from 0 to 1 according to

$$I_{z,\dot{z}} = 1 : |z_{Peak}(t)| > z_T \quad\text{and}\quad |\dot{z}_{Peak}(t^+)| > \dot{z}_T \tag{1}$$

$$\text{or}\qquad |\dot{z}_{Peak}(t)| > \dot{z}_T \quad\text{and}\quad |z_{Peak}(t^+)| > z_T \tag{2}$$

where $\dot{z}_{Peak}(t^+)$ is the first peak in $\dot{z}(t)$ following the event $|z_{Peak}(t)| > z_T$, and $z_{Peak}(t^+)$ is the first peak in $z(t)$ following the event $|\dot{z}_{Peak}(t)| > \dot{z}_T$. Figure 1(d) shows a plot of the resulting indicator function $I_{z,\dot{z}}$. Notice that the end of the lull is detected at $t = 596.4$ sec. This result compares favorably to the performance of the assumed one-peak-ahead, forecast-based threshold test shown in Fig. 1(c).

Lull/Swell Guidance Algorithm

The flow diagrams of the lull/swell guidance algorithm implemented in the real-time program are given in the Appendix. The algorithm consists of 1) the main driver program LULSIM, Fig. A1, which receives the ship-heave and heave-rate data, calls the logic program to determine a lull condition, and generates a binary switching sequence, IZ, where IZ = TRUE indicates the LAND signal and IZ = FALSE indicates the HOLD signal for the pilot; 2) LULLON, Fig. A2, a logical function that returns a value of TRUE if the conditions for a lull are satisfied, and FALSE if they are not satisfied; 3) LULOFF, Fig. A3, a logical function that returns a value of TRUE if the conditions for the termination of a lull are met and FALSE if they are not met; and 4) PKDTCT, Fig. A4, which detects a peak in the data given three values of a heave or heave rate. If a peak is detected, the absolute value of the peak is returned. These four subprograms were implemented and make up the lull/swell guidance algorithm.

Simulation Description

Simulator Facility

The evaluations were conducted at Ames Research Center on the Vertical Motion Simulator (VMS), with an interchangeable cab (Fig. 2) and a four-window, computer-generated-imagery (CGI) visual system (8). Included in the CGI data base was a DD-963 destroyer under way at sea. Special visual effects were included to represent various sea states, bow and stern wakes, ship motions, and variable ceiling and visibility conditions. A photograph of the SH-2F cockpit with a view of the CGI destroyer scene near the landing pad is shown in Fig. 3.

Simulation Mathematical Models

The helicopter simulated was a representation of an SH-2F helicopter with an advanced flight-control system. The mathematical model of the SH-2F helicopter contained a complete six-degree-of-freedom representation of the aircraft equations of motion, including all of the aerodynamic response characteristics of the SH-2F servo-tabbed main rotor, tail rotor, and fuselage through a ±180° range of angle of attack and sideslip.

The generic flight-control-law response characteristics that were implemented are categorized by mission phase-approach and hover/landing. The mode change from one to the other is initated by the pilot. For the approach phase, the control-response characteristics were rate-command attitude hold (RCAH) in pitch and roll, turn coordination in yaw, and vertical-velocity command in heave. For the hover/landing phase, two control options were evaluated. The first was attitude command (AC) in pitch and roll, heading hold in yaw, and vertical-velocity command in heave. The second was velocity-command position hold (VCPH) in pitch and roll, heading hold in yaw, and vertical-velocity command in heave. The VCPH has the property that, if the pilot centers the stick, the aircraft will automatically hold a constant position relative to the ship.

The landing environment around a DD-963 class destroyer was simulated. The ship-motion model was based on the ship response-amplitude-operator scheme (9) and used a sum of 70 sine waves to represent the motion of the ship for each of the ship's six degrees of freedom. Long-crested seas were assumed, with the mean wind vector parallel to the direction of the wave propagation. To allow the ship-motion computation to be performed in real time, an approach other than the direct computation of the sum of 70 sine waves was used. Instead, 20 min of ship motion were precomputed off-line, and nine continuous segments, each 4 min in length and spaced at 2-min intervals, were extracted. The nine segments were then stored directly into a file array at 0.05-sec intervals. While the simulation was in the INITIAL CONDITION mode, the desired segment file was transferred into the real-time program. In the OPERATE mode, a linear interpolation routine was used to update the ship-motion components at the desired time of the run.

A mathematical model of ship airwake turbulence defined by Fortenbaugh (10) was included. The airwake model generated both mean and random components of the three-dimensional flow field aft of the ship, as a function of aircraft position, wind-over-deck (WOD) magnitude, and WOD angle. A Dryden turbulence model was used outside the airwake, with shaping functions for smooth transition into the airwake.

A comprehensive model (11) of a conceptual, ship-based, microwave, scanning-beam, precision-landing guidance system (PLGS) was used to develop a model for use in the simulation. In the PLGS stabilization design, sensors were located at the PLGS antenna site to provide ship attitude and acceleration information for calculating a landing-pad deviation vector (LPDV). In practice, the LPDV would be sent to the aircraft via a data link, where it would be used to compensate for the PLGS antenna linear translations owing to ship motion. The scanning-beam antenna pattern was assumed to be stabilized with respect to the ship angular degrees of freedom through the use of a stabilized platform. The filtered estimate of ship heave and heave rate from the LPDV computation was used as the input to the lull/swell guidance algorithm.

The cockpit configuration for the simulation (Fig. 3) provided the pilot with the controls, displays, and instruments to effectively fly the aircraft. Instrument scan, control feel, and system functions were realistic. The simulator instrument panel closely matched that of the SH-2F aircraft. The pilot's controls in the simulator were the center cyclic stick, pedals, velocity-command lever (collective), trim switches, mode switches on the stick grip and collective, and a panel-mounted pushbutton annunciator panel.

The force-feel characteristics for use with the two flight-control law configurations (RCAH/AC and RCAH/VCPH) are summarized in Table 1. The center stick and pedals had the same configuration as in the SH-2F helicopter, but with somewhat different parameters. The collective level also had the same configuration as in the SH-2F. However, with the helicopter vertical-velocity response augmented, the collective lever became a vertical-velocity command lever, and a central detent was added to the friction-force characteristics. The detent indicated the central, zero position of the lever, at which the vertical-velocity command is zero. The force characteristics allowed small deflections from the center despite the presence of the detent.

Besides the conventional SH-2F primary instrument panel, a HUD device presented the pilot with flight trajectory, aircraft status, and command information in such a way as to permit a view outside the cockpit. The display is an overhead-mounted system in which a holographic or diffraction optic combiner is used. The hologram is a diffraction pattern which acts as a color- or wavelength-sensitive mirror that totally reflects the narrow-band green light emitted by the cathode-ray tube (CRT) phosphor, but transmits essentially all other wavelengths without attenuation. The projected symbols are magnified, collimated to infinity, and projected at a large instantaneous binocular field of view of approximately 30° laterally and 24° vertically.

The display format used to present the lull/swell guidance to the pilot is shown in Fig. 4 and was the same as described in Ref. 8. The format integrates explicit vertical and horizontal plane symbols. In the center of the display is the case-fixed aircraft reference symbol, ∇. Flight trajectory and aircraft status information is presented on the perimeter of the display in terms of range, lateral offset, vertical speed, altitude, and heading. Aircraft roll attitude is indicated on the scale at the bottom of the display above the heading tape. The pitch ladder, to the right of the aircraft symbol, rolls with the aircraft about the case-fixed aircraft symbol and has pitch lines of 10° intervals.

The horizontal situation is presented by the three slightly convergent lines which represent an extension of a runway, and move in relation to the helicopter's approaching the ship. The position of the case-fixed aircraft symbol relative to these moving lines represents the aircraft's location and heading relative to the desired approach path. When the helicopter is close to the ship, the convergent lines terminate into a symbol representative of the landing pad. The moving-map portion of the display changes scale continuously with range so as to keep it in view on the display at all times.

As the aircraft approaches the ship, a schematic landing pad appears at the end of the extended runway. At a range of 100 ft, the landing pad expands, over a 17-sec period, to twice its original size. At the same time, the extended runway and closure-rate symbols disappear and the sensitivities of the velocity vector, velocity reference, and acceleration cues double over the 17-sec interval. The schematic landing pad is approximately 70 ft long and 40 ft wide with a 24-ft-diam circle surrounding the touchdown point. By introducing control inputs to center the velocity vector over the desired touchdown spot, the pilot flies and maintains horizontal position over the deck. At an altitude of 50 ft, the rising runway appears to convey information on height above the deck until touchdown. During the ship landing, all of the information presented on the display is stabilized with respect to ship motion.

Upon the selection by the pilot of the hover mode, the LAND or flashing HOLD landing signal is presented to the pilot in the lower left portion of the display under the rising runway symbol.

Test Scope and Method

Two test pilots participated in the evaluations. The first had a total of 9300 hr of flying time, which included 1300 hr in a variety of helicopters, with some SH-2F and small-deck landing experience. The second pilot had 1100 hr of helicopter time, 900 of which were in the SH-2F, and had made about 400 landings on small ships, including ships of the DD-963 class. Most of the data collection runs were made by the second pilot.

For all cases, constant-bearing approaches were flown along a 30°-to-port radial with the ship assumed to be moving at a constant speed of 25 knots into a 25-knot wind. This combination of factors translated into a WOD condition of 43 knots, with the ship in a condition of sea state 5.

For baseline comparisons, a set of approaches to and landings upon the ship were made without the use of the lull/swell guidance. The pilots were instructed to perform the approach visually, assess the landing opportunity visually, and perform the landing. The baseline runs were performed with each control law configuration (VCPH and AC), and for each of the nine ship-motion segments.

To evaluate the lull/swell guidance aid, the run sequence was repeated with the guidance aid information presented to the pilot on the lower left portion of the HUD display. Performance data measuring the state of the helicopter and the ship as well as subjective Cooper-Harper pilot ratings (12) were recorded for each run.

Approach and Landing Procedures

The reference trajectory was a constant 30°-bearing approach from the starboard quarter of the ship, with the aircraft in trim at an indicated airspeed of 70 knots. The approach procedure used by the pilots was to fly a constant-attitude decelerating descent to a closure rate of 25 knots at 0.25 n. mi. and 20 knots at a range of 700 ft under instrument meteorological conditions (IMC). Visual breakout occurred at 700 ft, and subsequent approach, hover, and vertical descent to a landing occurred under visual meteorological conditions (VMC). The limited visibility condition (e.g., fog) was simulated so that no horizon was visible after visual acquisition of the ship.

The pilots used the HUD flight director and status information to track localizer, glide slope, and airspeed. The collective was used to track glide slope, and the lateral stick was used to maintain localizer. The constant-attitude deceleration profile was flown with pitch attitude.

After visual breakout, the pilots continued to the ship, using the HUD lineup cues and the visual cues provided by the ship's landing-pad and centerline drop lights. At a range of 100 ft or less, having reduced the closure rate to 10 knots or less, the pilot would engage the hover-control law (VCPH or AC) and proceed to establish a stable hover over the landing pad. At this point, the lull/swell guidance indicator would appear on the HUD. A steady LAND signal indicated that the ship's landing pad was in a lull condition and that a landing opportunity existed. A flashing HOLD signal indicated that the landing pad was experiencing a swell condition and that a landing opportunity did not exist. Upon encountering a

HOLD indication, the piloting technique was to remain at the hover point above the landing pad until a LAND indication occurred. If the pilot was performing a landing and a flashing HOLD indication occurred, the pilot was to abort the landing, climb to the hover point and wait for another LAND signal. In all cases, the pilots used the LAND/HOLD indicator as an aid to assist in the landing process and not as an explicit command.

Results

Approach and Landing Pilot Ratings

Cooper-Harper pilot ratings for the approach and transition-to-hover phases using the two control-law configurations are shown in Fig. 5, where the center of the ϕ symbol represents the mean value of the ratings, and the wings represent the variance. The approach and transition phase resulted in consistent ratings in the satisfactory and adequate ranges for all approaches performed. The variability in the data is due to the difficulty in performing the transition phase of the approach.

Pilot-rating data for the landing phase are shown in Fig. 6, where, as above, the data are presented as a function of control-law configuration. In addition, the rating data are shown for the approaches both with and without the lull/swell guidance. As in the approach phase, the ratings for the hover-landing phase are in the satisfactory and adequate ranges for all landings performed, with the lull/swell guidance having no significant effect on the pilot ratings.

For the landings with the VCPH control system, the pilots rated the landings between 3.0 and 4.5; and for the AC system, between 3.5 and 5.0. Although the pilots preferred the VCPH system to the AC system, the pilots did comment that, particularly in the high-sea-state and high-turbulence conditions, the VCPH did not hold position as well as desired. This deficiency required the pilots to actively control the aircraft at the hover point instead of using the position-hold feature as much as desired.

Landing Times

The piloting procedure used for the baseline landings was to approach the ship, shift into the hover-control law, maneuver up to the deck edge, hover, and wait for a lull period. Soon after, the ship motion would build up to a swell. The pilot would remain in the hover until the ship motion would enter into a lull a second time. Then the pilot would land.

The piloting procedure used with the lull/swell guidance was essentially the same; however, when the guidance indicated a positive ship landing-pad condition, the pilot would in many cases not wait for the second lull period to attempt the landing.

The effect of these procedures on the time required to land is illustrated in Fig. 7, where the elapsed time for each run from the start of the approach until touchdown is shown for each of the nine ship-motion segments. The landing-time data are shown superimposed over the lull/swell periods (I_z) estimated by the guidance algorithm that existed for each ship-motion segment. This figure shows the landing times for both the baseline and the lull/swell-guidance cases with each control system, and also shows the average time that the aircraft was 50 ft from the landing pad, $\bar{R}_{TD}$.

The data for the baseline cases indicate that the pilots were able to perceive landing opportunities subjectively without the aid of the lull/swell guidance, and always landed in a lull condition. The data further show that, of the 18 landings that were made on either of two successive lulls or landing opportunities for a given ship-motion segment, nine used the baseline procedure, and nine used the lull/swell guidance. For the baseline cases, three landings, or 33%, were made on the first lull, and six landings, or 67%, were made on the second lull.

Similarly, for the lull/swell-guidance cases, the pilots used the landing aid to assess the landing opportunity, and always landed in a lull condition. However, for the nine guidance cases, seven landings, or 78%, were made on the first lull, and two landings, or 22%, were made on the second lull. This was an improvement of 44% in the landing time over the baseline cases. Use of the lull/swell guidance resulted in significantly shorter landing times (by 30 to 50 sec) than occurred without its use.

The landing-time data also show that the performance of the lull/swell guidance was conservative in that it forecast the onset of deck lulls and swells later than the pilot could subjectively detect them. This effect appears in the data for those cases in which landings were made on the same lull opportunity of a given segment. For such cases, the baseline landings took less time (10 sec or less) than the corresponding landings with the lull/swell guidance. However, pilot comments indicate that the reliability of the guidance aid led to an increased pilot confidence in the existence of a landing opportunity. In no instances did the lull/swell guidance predict a ship lull with which the pilot did not agree.

The effect on landing time of the control system used by the pilot can also be seen in Fig. 7. Of the total of 10 guidance and baseline cases in which the pilots landed on the first lull opportunity instead of on the second, four were using the VCPH control system, and six were using the AC system. Thus, the control system configuration had little effect on the landing time.

Touchdown Performance Comparison

The landing performance of the aircraft with respect to the ship can be assessed in terms of the impact velocity and position error of the aircraft at touchdown. In the case of impact velocity, the relative vertical velocity between the ship and the deck as a function of the control system and the guidance algorithm is shown in Fig. 8. In all cases, the impact velocity is less than the 12 fps sink-rate structural limit of the landing gear that is allowed for the SH-2F helicopter in the ship landing environment (13).

The impact velocities for the VCPH control system are shown on the left, and the AC control system on the right. The VCPH landings show a slight improvement over the AC landings because the position-hold feature of the VCPH system allowed the pilot to concentrate on the vertical axes letdown. The slight increase in impact velocity of the guidance landings in comparison to the baseline landings occurred either because the pilots remained longer at the hover point and were therefore anxious to land before another swell arose, or because the added confidence provided by the guidance caused them to use less precision in the control of vertical speed.

The total position error of the aircraft at touchdown as a function of the control system and the guidance algorithms is shown in Fig. 9. In all cases the aircraft was within the 24-ft-diam landing circle on the deck. As for the impact velocity data, the position error data for the VCPH control system are shown on the left, and the AC control system data are shown on the right. The VCPH landings show a slight decrease in landing precision over the AC landings because of the drifting problem of the VCPH that was discussed earlier. The effect of the use of the lull/swell guidance algorithm with either control system shows only a slight improvement in favor of the use of the algorithm.

Lull/Swell Guidance Improvements

The lull/swell guidance output was implemented as a bi-state "LAND" or "HOLD" signal requiring the pilot to go from "land" to "don't land" discretely. Assuming the pilot was maneuvering to the hover point or controlling the aircraft from a gust upset while the "LAND" signal was on, the bi-state signal tends to make the pilot want to wait for the next landing opportunity to have the full period of the lull to make the landing. An in-between state, such as the amber light on a traffic signal, would give the pilot additional lead information of an impending swell useful in attempting a landing during long lulls.

For the combination of ship heading, wave direction, and wind direction evaluated in this paper, the primary ship motion was in the heave and pitch axis. For other conditions in which the primary motion may be in another axis, such as ship roll, the guidance algorithm can be easily modified and tuned as long as the ship motion displays the necessary lull/swell pattern structure.

The z vs $\dot{z}$ phase-plane concept was mechanized in the algorithm for early detection of the onset of a swell after a lull period. Use of this concept to detect the upcoming lull after a swell period may reduce the delay the pilots observed between their perception of a lull and the guidance algorithm's indication of a lull.

Conclusions

The results of the piloted simulation evaluation of the performance of a real-time predictive guidance law for landing a helicopter on a destroyer in adverse weather led to the following conclusions:

1. Use of the lull/swell guidance law resulted in increased pilot confidence in the existence of a ship landing opportunity, and resulted in significantly shorter hover wait times prior to landing.

2. Without the lull/swell guidance, pilots were able to visually perceive ship lulls and swells prior to landing, but because of a lack of confidence often elected to land on the second landing opportunity instead of on the first.

3. The pilots preferred the VCPH aircraft control law over the AC law even though use of these control laws had little effect on the impact velocity and position error at touchdown.

Acknowledgments

The authors are grateful to Bryan Martin and Mike Alderate (NASA Ames Research Center) for their assistance in the data analysis, and to Ron Gerdes (NASA Ames Research Center) and Lt. Brian Watkins (Naval Air Test Center) for their participation as evaluation pilots.

References

(1) T. S. Durand and R. Wasicko, "An Analysis of Carrier Landing," AIAA Paper 65-791, 1965.

(2) P. Kaplan, "A Study of Prediction Techniques for Aircraft Carrier Motion at Sea," J. Hydronautics, Vol. 3, No. 3, pp. 121-131: 1969.

(3) J. L. Loeb, "Automatic Landing Systems Are Here," in Proc. AGARD Conference, 1970, Paper 14.

(4) A. E. Baitis, "The Influence of Ship Motion on Operations of SH-2F Helicopters from DE-1052 Class Ships; Sea Trial with U.S.S. Bowen (DE-1079)," DTNSRDC Rep DPD 556-01, July, 1975.

(5) M. M. Sidar and B. R. Doolin, "On Feasibility of Real-Time Prediction of Aircraft Carrier Motion at Sea," IEEE Transactions on Automatic Control, Vol. AC-28, No. 3, March, 1983.

(6) C. G. McMuldrock, G. Stein, and M. Athans, "VTOL Control for Shipboard Landing in High Sea States," 18th IEEE Conference on Decision and Control, Dec., 1979.

(7) A. V. Phatak, M. S. Karmali, and C. H. Paulk, "Ship Motion Pattern Directed VTOL Letdown Guidance," 1983 American Control Conference, San Francisco, Calif., June, 1983.

(8) C. H. Paulk, D. L. Astill, and S. T. Donley, "Simulation and Evaluation of the SH-2F Helicopter in the Shipboard Environment Using the Interchangeable Cab System," NASA TM-84387, Aug., 1983.

(9) R. G. Brown and F. A. Camaratta, "NAVAIRENGCEN Ship Motion Computer Program," NAEC Report NAEC MISC-903-8, 1978.

(10) R. L. Fortenbaugh, "Progress in Mathematical Modeling of the Aircraft Operational Environment of DD 963 Class Ships," AIAA Paper 79-1677, Boulder, Colo., 1979.

(11) L. A. McGee, C. H. Paulk, S. A. Steck, S. F. Schmidt, and A. W. Merz, "Evaluation of the Navigation Performance of Shipboard VTOL Landing Guidance Systems," AIAA Journal of Guidance and Control, Vol. 4, No. 4, July-Aug., 1981.

(12) G. E. Cooper and R. E. Harper, "The Use of Pilot Rating in the Evaluation of Aircraft Handling Qualities," NASA TN D-5153, 1969.

(13) "Detailed Specification for Conversion of HH-2D Helicopter to Model SH-2F Helicopter," Dept. of the Navy, SD-557-3, June 29, 1973.

Table 1 Advanced Controller Characteristics

Parameter	Longitudinal stick	Lateral stick	Rudder pedals	Collective	
				Unaugmented	Augmented
Travel, in.	±5.5	±5.5	±3.0	0 to 11.5	±5.75
Breakout force, lb	±1.25	±1.0	±2.5	*a*	*b*
Force gradient, lb/in.	2.0	1.0	6.0	*a*	*b*
Maximum deflection force, lb	±12.25	±6.25	±20.5	*a*	*b*

[a]Adjustable friction force, 1 lb minimum.

[b]Adjustable friction force with central detent.

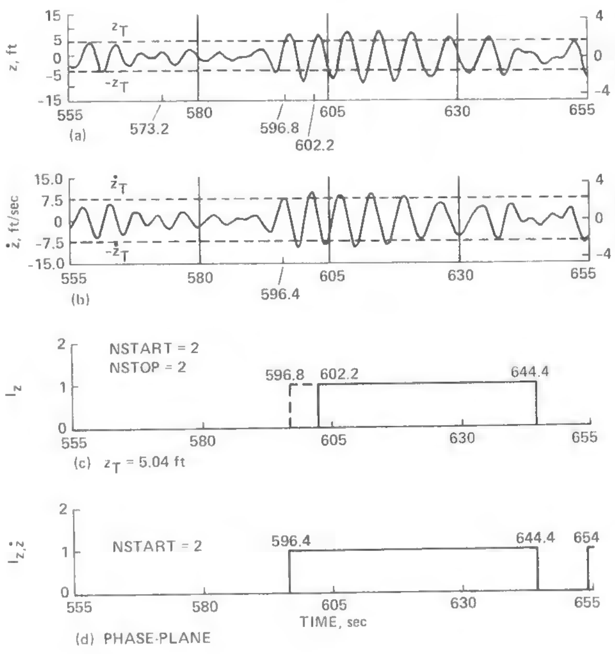


Fig. 1 Lull/swell classification algorithm.

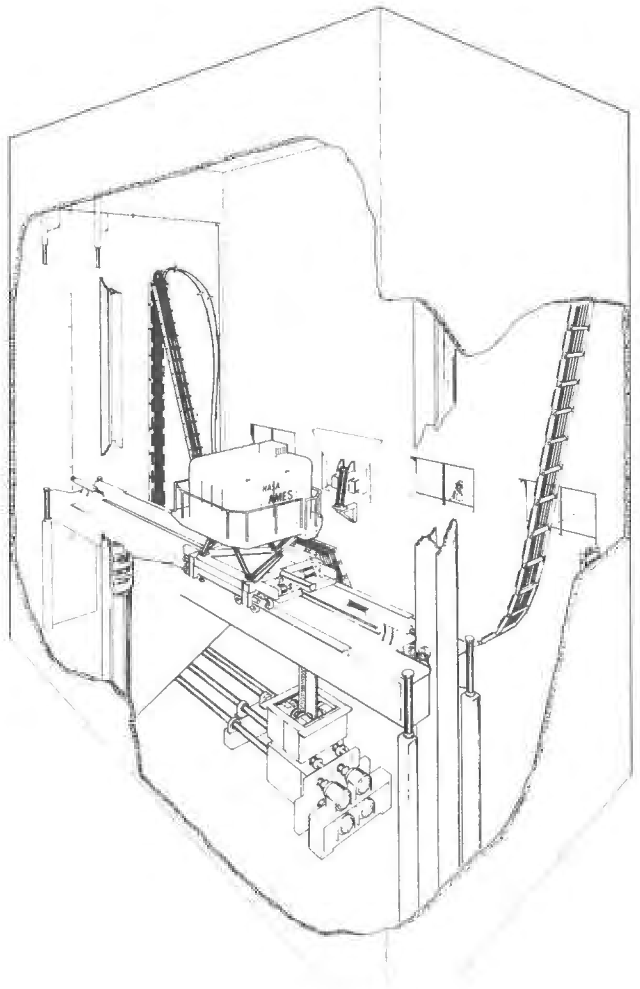


Fig. 2 Vertical Motion Simulator with interchangeable cab.

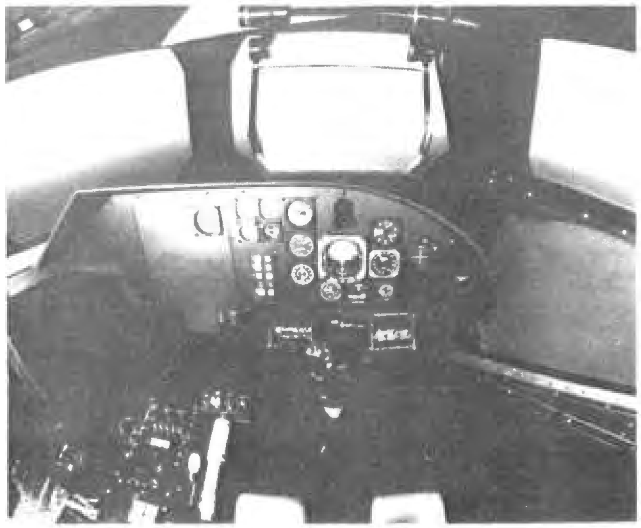
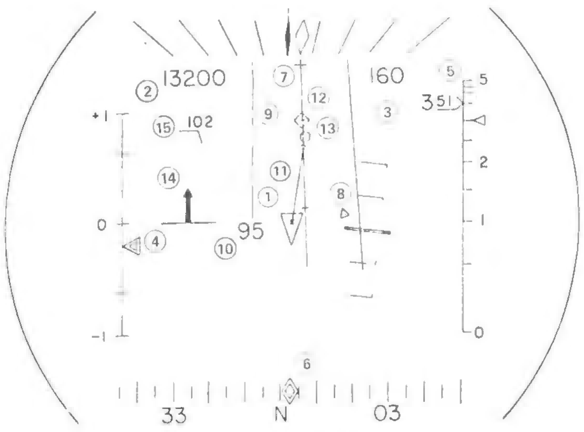
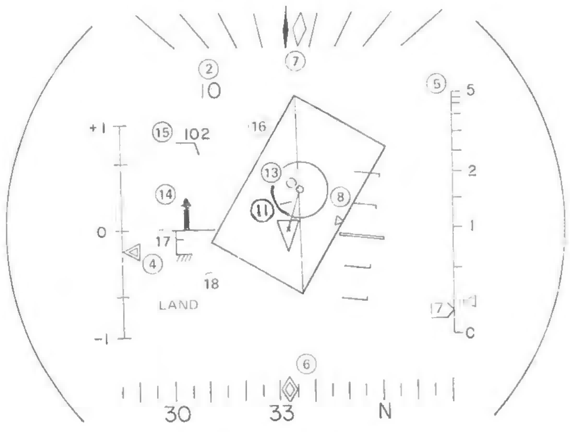


Fig. 3 SH-2F simulator cockpit.



APPROACH FORMAT



HOVER/LANDING FORMAT

SYMBOL	INFORMATION	APPROACH FORMAT	HOVER/LANDING FORMAT
(1) AIRCRAFT REFERENCE ▽	CASE FIXED REFERENCE	X	X
(2) RANGE	DME RANGE TO SHIP	X	X
(3) LATERAL OFFSET	DIGITAL LOCALIZER OFFSET	X	
(4) VERTICAL SPEED	MOVING POINTER WITH FULL-SCALE DEFLECTION OF ±1000 ft/min	X	X
(5) ALTITUDE	HEIGHT ABOVE GROUND. COMMANDED ALTITUDE IS SMALL POINTER.	X	X
(6) HEADING	MOVING TAPE INDICATION OF HEADING	X	X
(7) ROLL ATTITUDE	SCALE, INDEX & MOVING DIAMOND	X	X
(8) PITCH ATTITUDE	MOVING POINTER ON PITCH LADDER. LADDER ROTATES AROUND A/C SYMBOL WITH ROLL ATTITUDE.	X	X
(9) HORIZONTAL SITUATION	3 ALMOST VERTICAL LINES REPRESENT EXTENSION OF RUNWAY & MOVING MAP	X	
(10) AIRSPEED/CLOSURE RATE	AIRSPEED ABOVE 40 knots, SHIP CLOSURE RATE OTHERWISE	X	
(11) VELOCITY VECTOR ⚲	SHRINKING LINE REPRESENTS VELOCITY TO SHIP	X	X
(12) VELOCITY REFERENCE ◇	SMALL DIAMOND. DISPLACEMENT OF END OF VELOCITY VECTOR RELATIVE TO ◇ REPRESENTS VELOCITY ERROR	X	
(13) "ACCELERATION" CUE ○	DISPLACEMENT BETWEEN ○ AND ◇ CORRESPONDS TO CYCLIC DIRECTOR COMMANDS	X	X
(14) COLLECTIVE DIRECTOR	COLLECTIVE DIRECTOR COMMANDS	X	X
(15) TORQUE	DIGITAL READOUT. HOCKEY STICK SHOWS PERCENT OF MAXIMUM.	X	X
(16) LANDING PAD	DENOTES 70 x 40 ft LANDING PAD		X
(17) RADAR ALTITUDE	RADAR HEIGHT ABOVE GROUND/ WATER/DECK		X
(18) LULL/SWELL GUIDANCE	LAND & FLASHING HOLD INDICATION		X

Fig. 4 Display symbology for HUD.

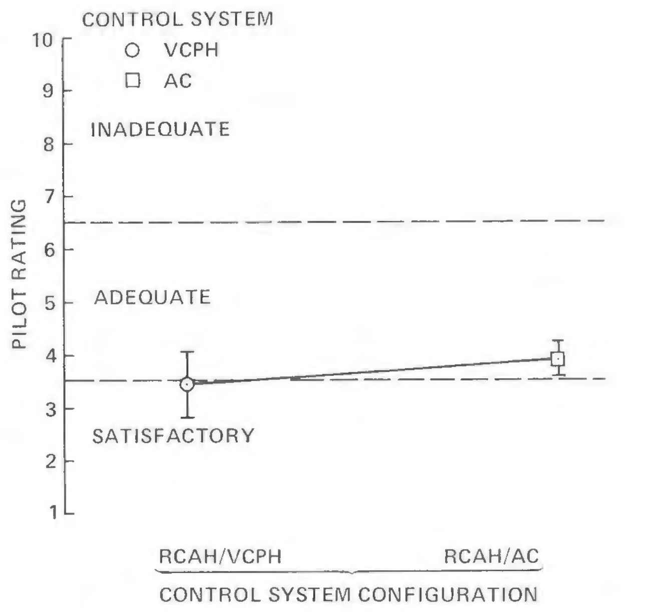


Fig. 5 Pilot rating comparison for approach phase.

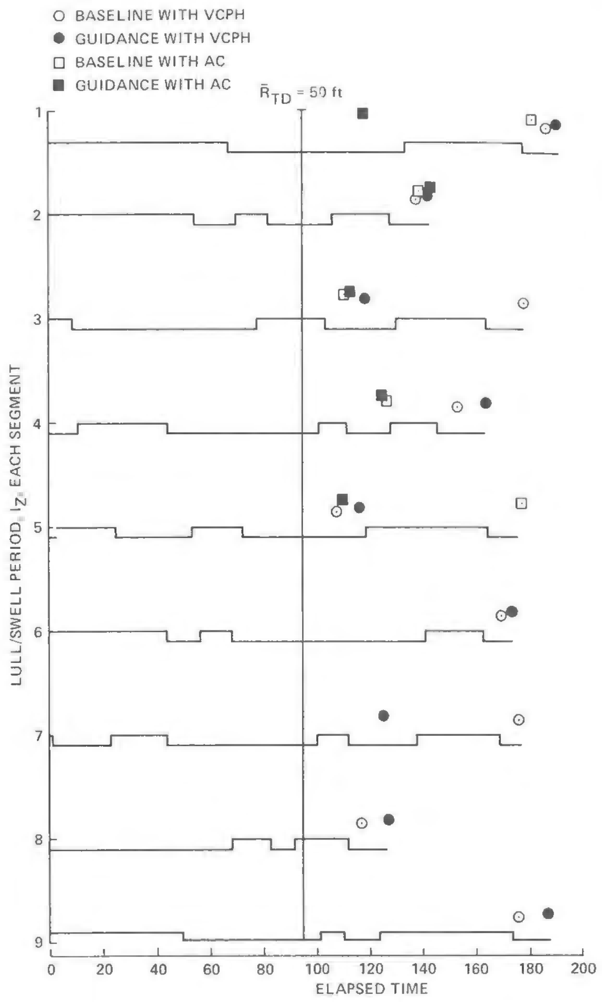


Fig. 7 Landing time composite for all cases.

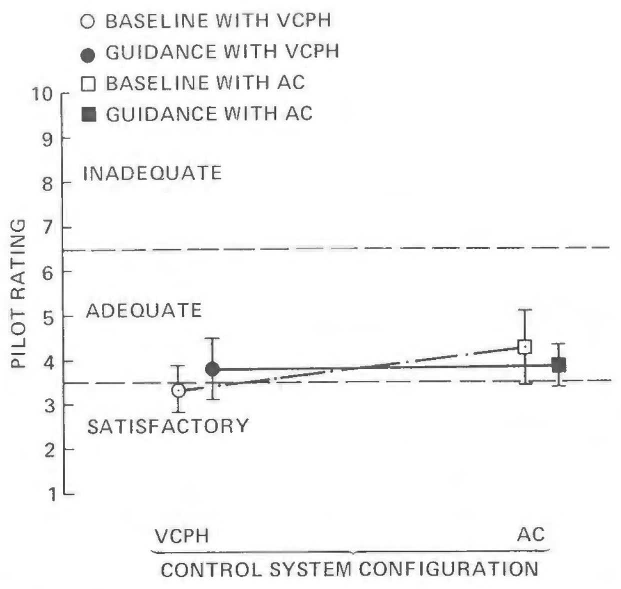


Fig. 6 Pilot rating comparison for hover/landing phase.

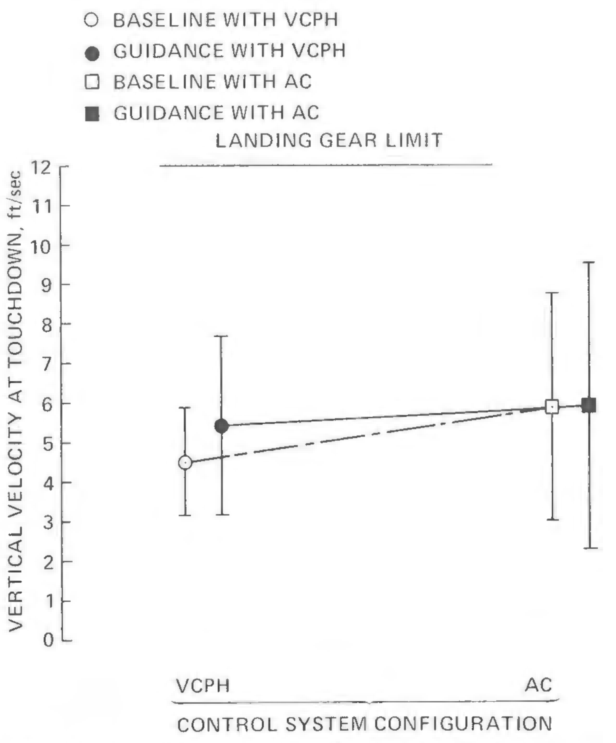


Fig. 8 Impact vertical velocity performance comparison.

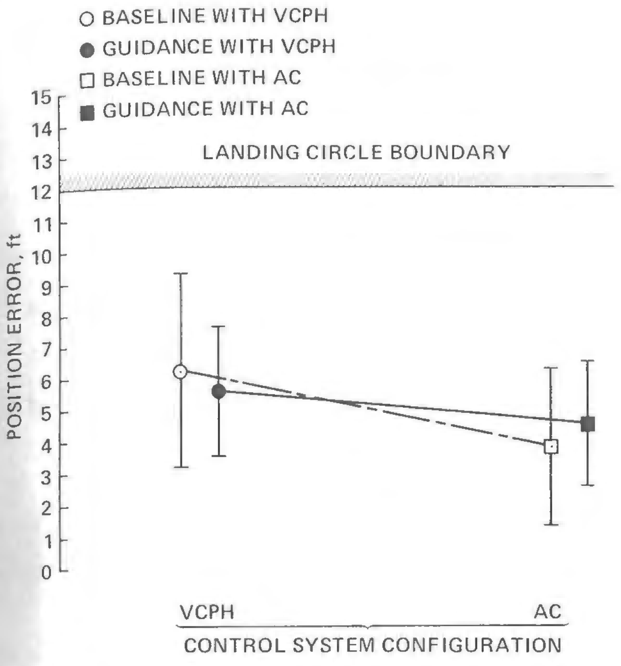
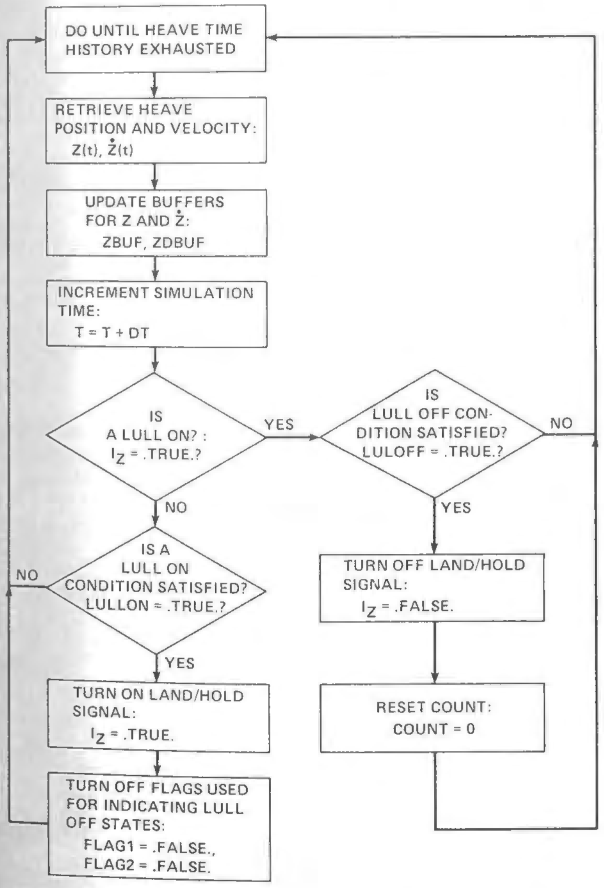


Fig. 9 Touchdown position error comparison.

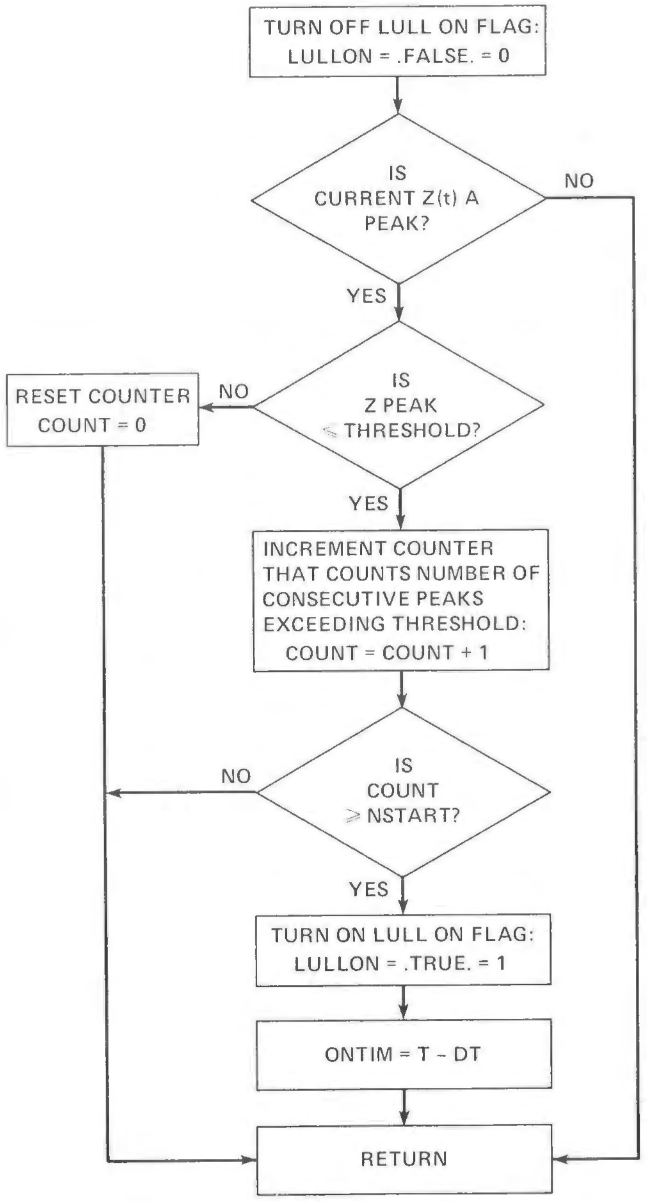
Appendix

Following are flow diagrams of the lull/swell guidance algorithm. The algorithm was implemented in real-time to provide the LAND or HOLD indication to the pilot.



DO UNTIL HEAVE TIME HISTORY EXHAUSTED
RETRIEVE HEAVE POSITION AND VELOCITY: Z(t), Ż(t)
UPDATE BUFFERS FOR Z AND Ż: ZBUF, ZDBUF
INCREMENT SIMULATION TIME: T = T + DT
IS A LULL ON? : I_Z = .TRUE.?
YES
IS LULL OFF CONDITION SATISFIED? LULOFF = .TRUE.?
NO
YES
TURN OFF LAND/HOLD SIGNAL: I_Z = .FALSE.
RESET COUNT: COUNT = 0
NO
IS A LULL ON CONDITION SATISFIED? LULLON = .TRUE.?
NO
YES
TURN ON LAND/HOLD SIGNAL: I_Z = .TRUE.
TURN OFF FLAGS USED FOR INDICATING LULL OFF STATES: FLAG1 = .FALSE., FLAG2 = .FALSE.

Fig. A1 LULSIM—Main driver program for lull/swell guidance.



TURN OFF LULL ON FLAG: LULLON = .FALSE. = 0
IS CURRENT Z(t) A PEAK?
NO
YES
IS Z PEAK ≤ THRESHOLD?
NO
RESET COUNTER COUNT = 0
YES
INCREMENT COUNTER THAT COUNTS NUMBER OF CONSECUTIVE PEAKS EXCEEDING THRESHOLD: COUNT = COUNT + 1
IS COUNT ≥ NSTART?
NO
YES
TURN ON LULL ON FLAG: LULLON = .TRUE. = 1
ONTIM = T - DT
RETURN

Fig. A2 LULLON—Logical function to determine a lull.

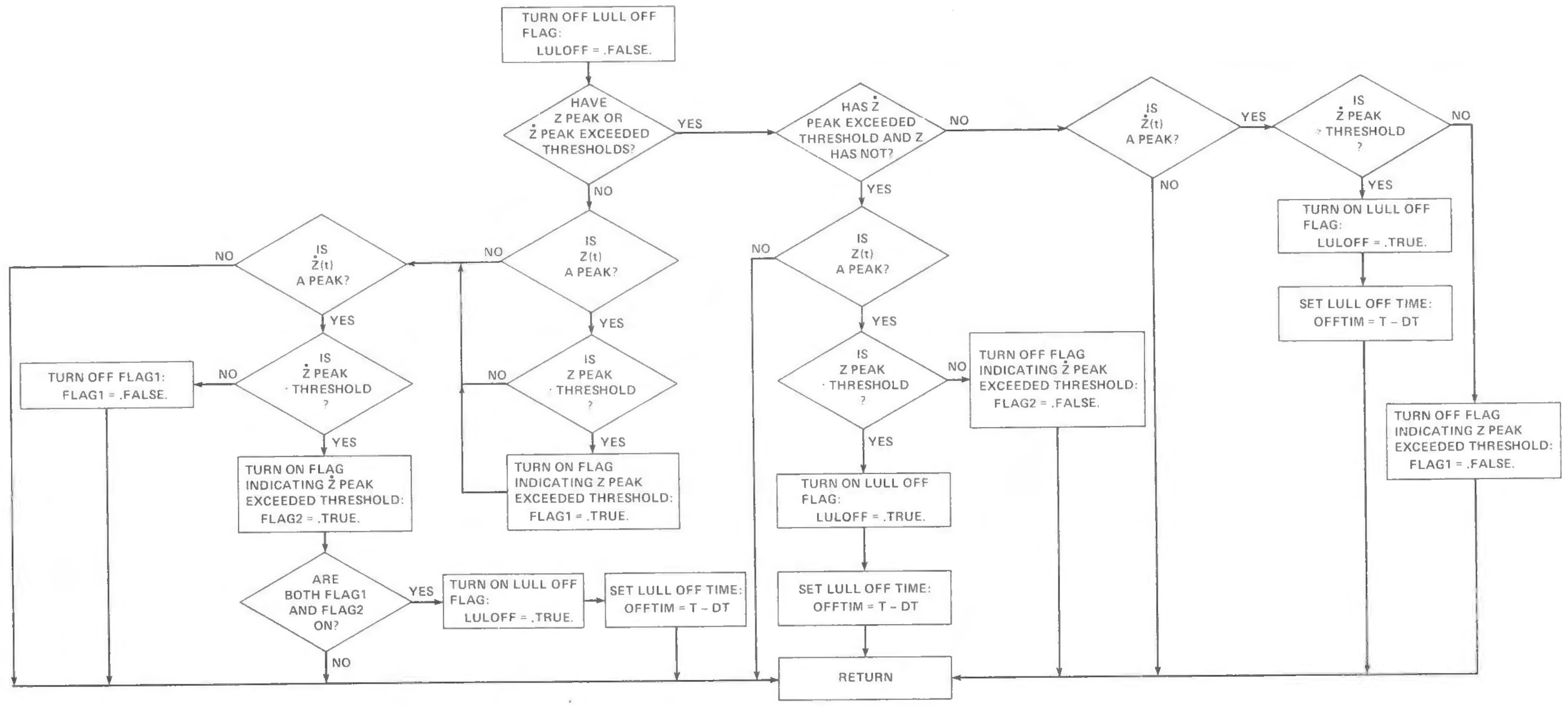

TURN OFF LULL OFF FLAG: LULOFF = .FALSE.
HAVE Z PEAK OR Ż PEAK EXCEEDED THRESHOLDS?
YES
NO
HAS Ż PEAK EXCEEDED THRESHOLD AND Z HAS NOT?
NO
YES
IS Ż(t) A PEAK?
YES
NO
IS Ż PEAK > THRESHOLD ?
YES
NO
TURN ON LULL OFF FLAG: LULOFF = .TRUE.
SET LULL OFF TIME: OFFTIM = T – DT
TURN OFF FLAG INDICATING Z PEAK EXCEEDED THRESHOLD: FLAG1 = .FALSE.
IS Z(t) A PEAK?
NO
YES
IS Z PEAK > THRESHOLD ?
NO
YES
TURN ON FLAG INDICATING Z PEAK EXCEEDED THRESHOLD: FLAG1 = .TRUE.
IS Ż(t) A PEAK?
NO
YES
IS Ż PEAK > THRESHOLD ?
NO
YES
TURN OFF FLAG1: FLAG1 = .FALSE.
TURN ON FLAG INDICATING Ż PEAK EXCEEDED THRESHOLD: FLAG2 = .TRUE.
ARE BOTH FLAG1 AND FLAG2 ON?
YES
NO
TURN ON LULL OFF FLAG: LULOFF = .TRUE.
SET LULL OFF TIME: OFFTIM = T – DT
IS Z(t) A PEAK?
NO
YES
IS Z PEAK > THRESHOLD ?
NO
YES
TURN OFF FLAG INDICATING Ż PEAK EXCEEDED THRESHOLD: FLAG2 = .FALSE.
TURN ON LULL OFF FLAG: LULOFF = .TRUE.
SET LULL OFF TIME: OFFTIM = T – DT
RETURN

Fig. A3 LULLOFF—Logical function to determine a swell.

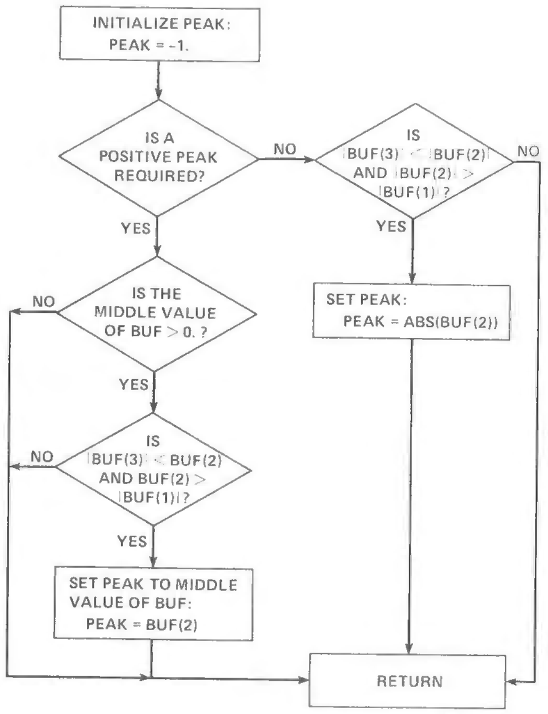

INITIALIZE PEAK:
PEAK = -1.
IS A POSITIVE PEAK REQUIRED?
NO
YES
IS |BUF(3)| < |BUF(2)| AND |BUF(2)| > |BUF(1)| ?
NO
YES
IS THE MIDDLE VALUE OF BUF > 0. ?
NO
YES
SET PEAK:
PEAK = ABS(BUF(2))
IS BUF(3) < BUF(2) AND BUF(2) > BUF(1) ?
NO
YES
SET PEAK TO MIDDLE VALUE OF BUF:
PEAK = BUF(2)
RETURN

Fig. A4 PKDTCT— Logical function to detect a peak.

84-2674

DIGITAL AVIONICS AND FLIGHT PATH DIRECTOR FUNCTIONS OF THE HH-60 HELICOPTER

Frank G. Kilmer, Robert L. Kilmer, and Blake A. Thalacker

IBM Federal Systems Division
Owego, New York 13827

Abstract

The HH-60 helicopter horizontal and vertical guidance techniques, flight director modes, and display cue symbology are described. Terminal area approach to a hover and terrain following over an isolated peak are demonstrated by means of non-real time simulation. The paper concludes with a plan for piloted helicopter simulation at the NASA-Ames facility to further evaluate director and display concepts.

Introduction

The HH-60 is a United States Air Force (USAF) search and rescue helicopter program. The missions of the HH-60 Night Hawk include search, rescue and recovery by the Air Force Aerospace Rescue and Recovery Service. The HH-60 avionics system permits low level, night, adverse weather flight and the precision operations that may be required for successful survivor location and rescue.

Integrated Avionics

The HH-60 integrated avionics features a digital data bus, two Mission Computers and four cathod ray tube Multi-Purpose Displays (MPDs) as shown in Figure 1. A Military Standard (MIL-STD) 1553B data bus provides the primary equipments interface. Equipment which is not bus compatible is connected to the bus through Remote Terminal Units (RTUs). Essential engine and flight data are routed through multiple RTUs to avoid a single point failure.

The on-board sensors include a Multi-Mode Radar (MMR), a Map Reader (MR), Night Vision Goggles (NVGs), and a Forward Looking InfraRed (FLIR). The MMR is a modified LANTIRN Terrain Following Radar originally developed for the F-16 aircraft, having the HH-60 modes of Terrain Avoidance, Ground Mapping, and Air-to-Ground Ranging. The Terrain Following mode is based on the ADLAT algorithm discussed in the Vertical Guidance section of this paper. In this mode, the MMR stores terrain data within 15 degrees azimuth of either side of the aircraft groundtrack to provide terrain data during turning flight. The FLIR has three fields of view (FOV): a wide 30 by 40 degree FOV, a medium 15 by 20 degree FOV, and a narrow 5 by 6.7 degree FOV. FLIR pointing can be controlled either manually or by the Mission Computer. When FLIR pointing is controlled by the Mission Computer, the FLIR line of sight is stabilized to either a point on the ground or the aircraft flight path vector. The Map Reader contains a 35 millimeter film map that is positioned by the Mission Computer; the MR video signal can be displayed on any MPD.

An Inertial Navigation System (INS) with platform inertial sensors, an Attitude Heading Reference System (AHRS) with body-mounted inertial sensors, and a Doppler velocity sensor provide for self contained navigation. The primary navigation mode is a Doppler/INS mode that provides damped inertial navigation with a relatively low drift rate. Alternative navigation modes are INS, Doppler/AHRS, AHRS, and Air Data Dead Reckoning.

The MIL-STD 1750A Misson Computers are redundant. One Mission Computer is designated "prime" while the other is a backup. The prime computer collects input data from the data bus and then passes that data to the backup computer, which has the same resident software. Failure detection is achieved by computer self test and output data comparison of the two "loosely synchronized" computers. An automatic switch-over to the backup Mission Computer occurs in the event of a detected computer failure.

Crew Interface

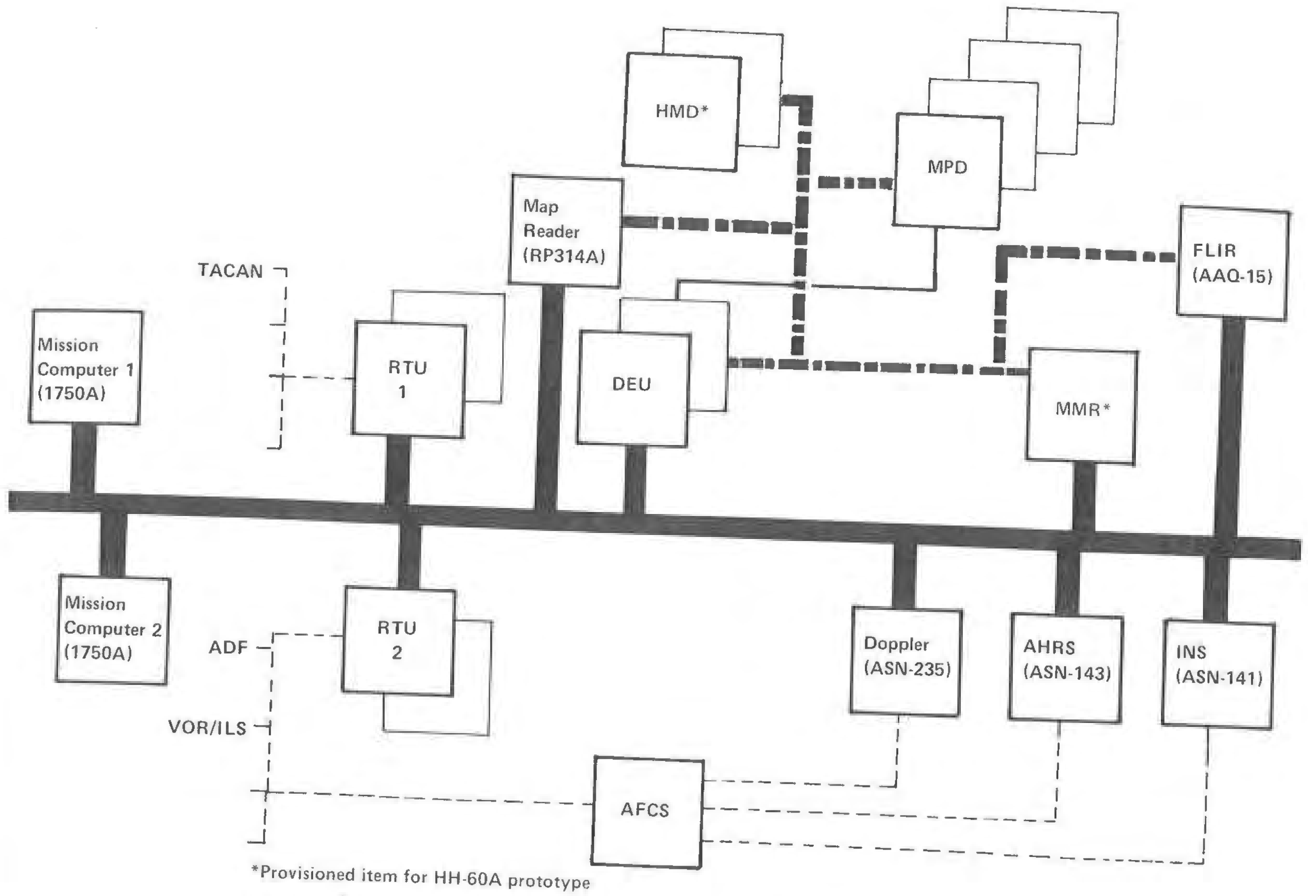
The crew interacts with the avionics through the instrument panel mounted MPDs, Helmet Mounted Displays (HMDs), alpha-numeric keyboards, and a tracking handle.

The MPDs are located on the instrument panel as shown in Figure 2. Software programmable switches (softkeys) located around the periphery of the MPDs allow the operators to select display formats and control system moding. Switches located on the cyclic and collective grips allow the operator to select softkeys without removing his hands from the flight controls. Any display format can be displayed on any one of the MPDs, with each of the MPDs able to simultaneously display a different format. The displays are driven by two redundant symbol generators called Display Electronics Units (DEUs). If one DEU fails, the remaining DEU can drive all four displays but with, at most, two selected formats. Sensor generated video (FLIR, Radar or Map) can overlay flight instrument and command cue symbology on the MPDs. The MPDs can also display tabular data such as communications channel selection, navigation parameters, or flight plan data.

The HMDs provide a "heads-up" flight capability by repeating the outboard display video. When the HMD is in use, the FLIR may be slaved to the pilot's helmet orientation to allow FLIR pointing by head movement. The FLIR video may be scaled to match the outside scene.

Two redundant keyboards are provided to allow for data entry by either pilot. Flight planning is accomplished by entering Navigational Reference Points via the keyboard or by slewing the map position to a desired location and storing the latitude and longitude of the map position.

Basic (backup) flight instruments are located in the center of the instrument panel. These instruments are made NVG compatible, in order to be



*Provisioned item for HH-60A prototype

MMR	-	Multi Mode Radar
FLIR	-	Forward Looking Infrared
MPD	-	Multi Purpose Display
HMD	-	Helmet Mounted Display
DEU	-	Display Electronics Unit
RTU	-	Remote Terminal Unit
AHRS	-	Attitude Heading Reference System
INS	-	Inertial Navigation System
AFCS	-	Automatic Flight Control System

▬ (thick)	Serial Digital Data (1553B)
▬ (thin)	Serial Digital Data (UART)
■■■■	Composite Video
– – –	Analog Data

Figure 1. Functional Diagram of the HH-60 Digital Avionics System

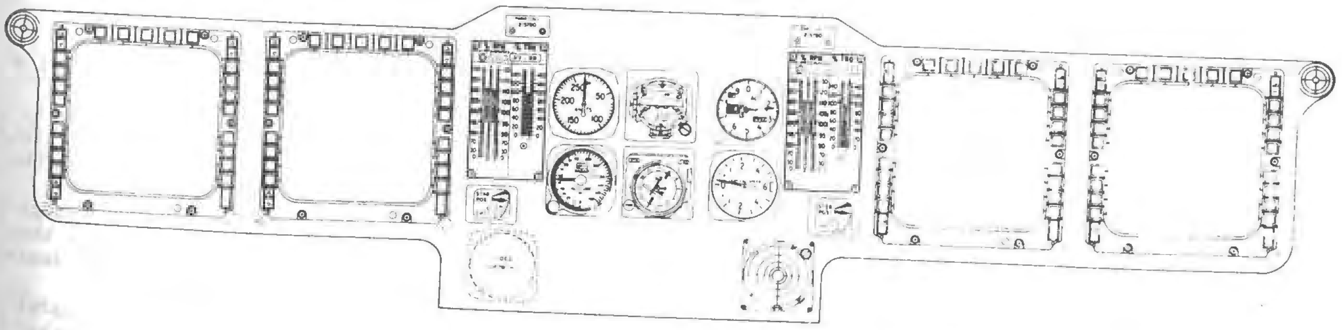


Figure 2. HH-60 Instrument Panel Layout

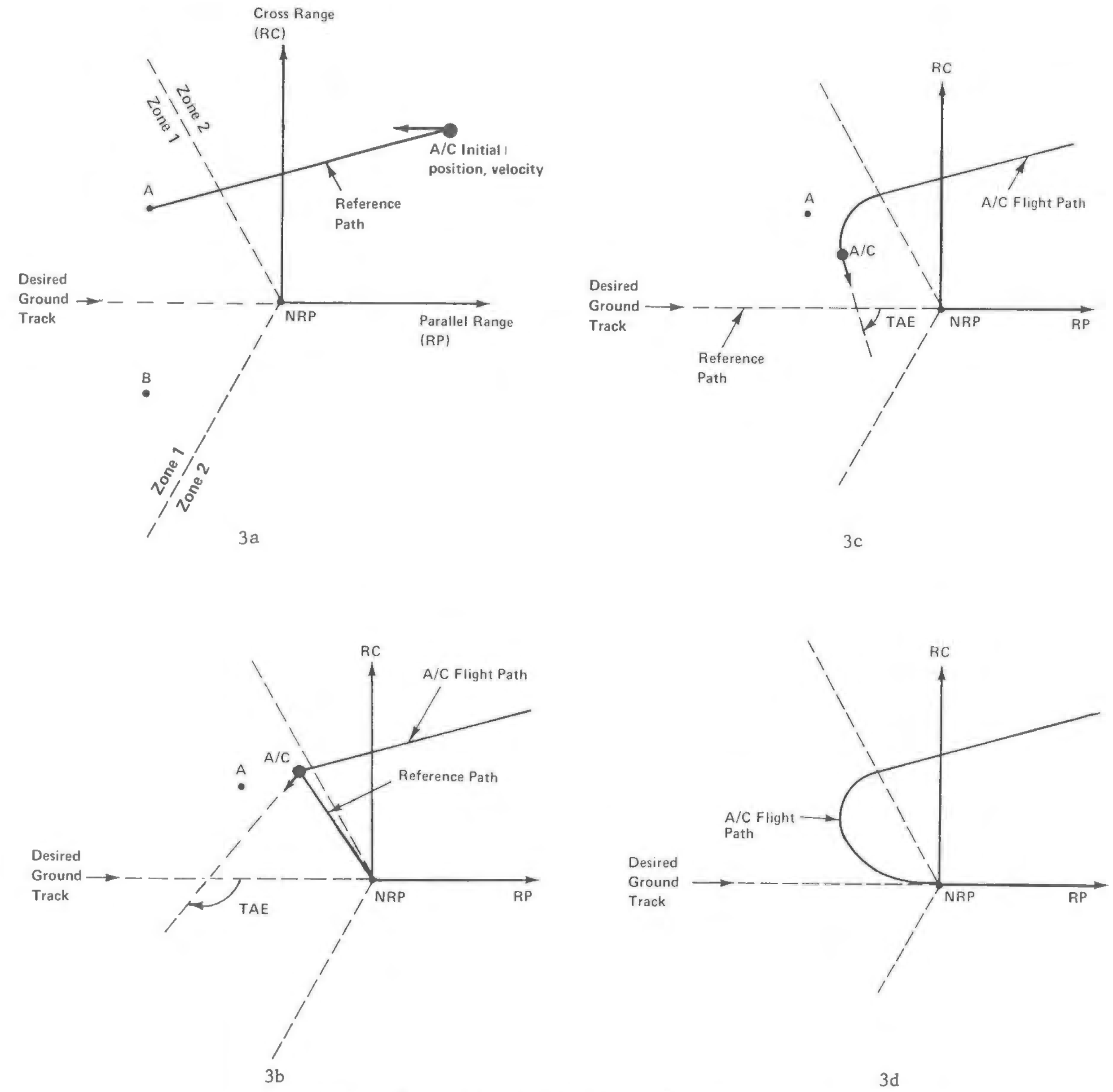


Figure 3. Horizontal Guidance Geometry

consistent with the requirement for NVG compatibility of the entire cockpit.

Guidance

The HH-60 Guidance algorithms compute reference paths for aircraft flight. Horizontal guidance determines a reference path to guide the aircraft to a Navigation Reference Point (NRP) along a prescribed groundtrack. Vertical guidance for the HH-60 is either a Terrain Following path or an Approach-to-Hover path.

Horizontal Guidance

The horizontal guidance technique divides the region surrounding the NRP into two zones as shown in Figure 3a. Zone 2 is a "go-around" region while Zone 1 is a "direct-approach" region. If the aircraft is in Zone 2, the reference path is a radial from the aircraft present position to point A (or point B for negative cross range); a reference path that continuously changes with time. When the aircraft transitions to Zone 1, if the magnitude of the groundtrack angle error (TAE) is greater than 90 degrees (Figure 3b), then the reference path is a radial from the present position to the NRP; again, continuously changing with time. If the magnitude of TAE is less than 90 degrees, the reference path becomes the desired groundtrack through the NRP (Figure 3c). Reference paths computed in this manner consist of straight line segments. Achieved flight paths corresponding to the described set of reference straight line segments will closely approximate a straight line, circular, straight line path as shown in Figure 3d. Simulation has shown that these achieved flight paths compare favorably in path length with the near-minimum path technique of Reference (1).

Vertical Guidance

The HH-60 autonomous vertical guidance techniques consist of Terrain Following and Approach-to-Hover descent.

Terrain Following. The primary vertical guidance mode is Terrain Following (TF). In the TF mode, the MMR collects and stores terrain data along the projected groundtrack corridor to compute TF commands for output to the Director TF control laws. The TF command computation is based on the Advanced Low Altitude Technique (ADLAT) developed by Cornell Aeronautical Laboratory (now Arvin/Calspan) to obtain a commanded flight vector that will guide the aircraft over the terrain at the set clearance. This technique compensates for system dynamics as well as climb, dive limits and maximum desired accelerations to provide an "ideal" trajectory over isolated obstacles while minimizing the deviation from the set clearance at the peak crossings.

The ADLAT algorithm employs the two-parabola geometry shown in Figure 4. If the climb capability of the aircraft is not exceeded, then the two-parabola geometry provides a constant acceleration ("g") pull-up maneuver followed by a constant g push-over with the intent of producing level flight at the peak. In the climb limited case depicted in Figure 5, ADLAT computes the point for initiating the climb as a function of the aircraft climb limit. The ADLAT algorithm is described in detail in Reference (2).

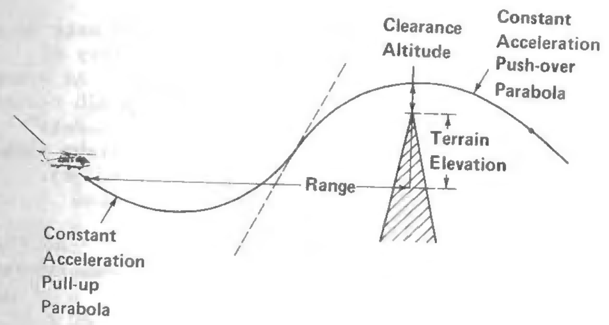

Clearance Altitude
Constant Acceleration Push-over Parabola
Terrain Elevation
Range
Constant Acceleration Pull-up Parabola

Figure 4. ADLAT 2-Parabola Geometry

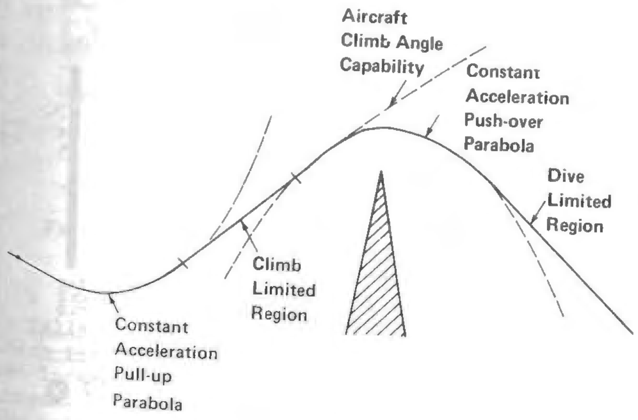

Aircraft Climb Angle Capability
Constant Acceleration Push-over Parabola
Dive Limited Region
Climb Limited Region
Constant Acceleration Pull-up Parabola

Figure 5. ADLAT Climb Limited Geometry

Approach-to-Hover (APPR). Guidance for Approach-to-Hover, as shown in Figure 6a, provides controlled deceleration/descent to the hover point as a function of the Range-To-Go (RTG) to the hover point. The altitude profile is a 6 degree glide-slope to an offset point, allowing the hover to be approached at a constant altitude. The velocity profile is parabolic velocity vs RTG, switching to linear velocity vs RTG to prevent excessive helicopter pitch attitudes as the hover point is approached (3,8). Expressed as functions of time, the parabolic segment corresponds to a constant deceleration; the linear segment is a time decaying exponential flare (see Figure 6b). The altitude is parabolic in time with an exponential flare.

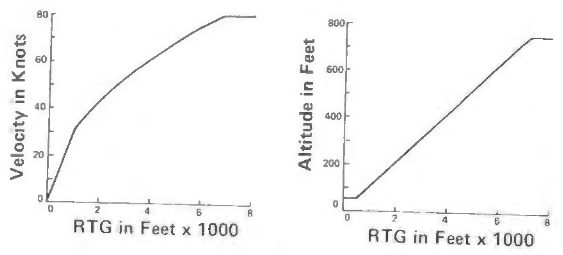


Figure 6a. Approach-to-Hover Profiles

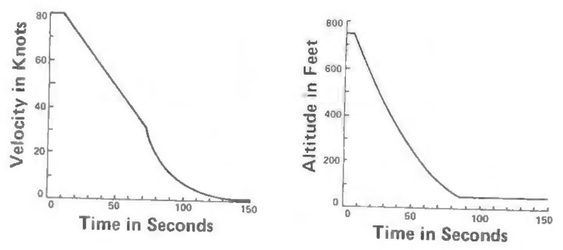


Figure 6b. Approach-to-Hover Profiles

Flight Path Director

The HH-60 Director algorithms transform flight path errors into commanded displacement of MPD symbols, i.e., Director cues. Piloted flight, using the cyclic and collective controls, in response to the Director cues results in convergence to the reference paths, with moderate workload.

Director Modes

Flight Path Director modes are provided for virtually every phase of the mission, as indicated by the list of HH-60 Director modes in Table I. Director modes are activated upon operator request using the MPD softkeys. Moding logic precludes contention among the Director modes for output commands. For example, TF is the highest priority vertical mode.

Director mode capabilities are described below, with emphasis given to the 3-cue APPR mode.

Altitude Hold. The Altitude Hold mode cues the pilot to hold either the barometric or radar altitude existing at the time of mode activation.

Airspeed Hold. The Airspeed Hold mode cues the pilot to hold the true airspeed existing at the time of mode activation.

Table I. HH-60 Director Modes

Director Mode	Cues
Altitude Hold	Vertical (Ver)
Airspeed Hold	Longitudinal (Long)
Terrain Following	Vertical
Lateral Steering	Lateral (Lat)
Waypoint (Direct)	↓
Waypoint (Course)	
Search Patterns	
Expanding Squares	
Creeping Lines	
Sector Search	
VOR	
TACAN	
ADF	
Approach-to-Hover	Long, Lat, Ver
Depart Hover	Long, Ver
ILS	Lat, Ver
ILS-BC	Lat

Terrain Following. The TF mode aids pilot control of the vertical flight path angle (or equivalently, vertical velocity) in order to achieve Terrain Following with essentially level flight at a set clearance over terrain peaks.

Lateral Steering. The Lateral Steering mode aids flight along the computed reference groundtrack during Waypoint, Search Pattern and Approach-to-Hover modes; and along radio beams during VOR, TACAN and ADF modes.

Depart-from-Hover. This depart mode cues the pilot for gradual climb-out to an altitude of 500 feet and airspeed of 100 knots.

Instrument Landing System. This approach mode cues the pilot for capture and track of the instrument landing radio beams.

Approach-to-Hover. The APPR mode cues the pilot for a programmed deceleration and descent along a prescribed groundtrack.

The Director outputs during APPR mode operation are expressed in functional notation as:

$$DVERC = K_{VER}\ (K_{V_Z}(K_Z\epsilon_Z+V_{Z_{lead}}-V_Z)-COLL_{WO})$$

$$DLONC = K_{LON}\ (K_q(K_{AS}\epsilon_{AS}+\theta_{lead}-\theta_{WO})-q)$$

$$DLATC = K_{LAT}\ (K_p(\phi_c-\phi_{FB})-p)$$

where $\epsilon(\ )$ denotes the difference between command and measured value, commands are proportional to gains K( ), and the subscript WO denotes a washed-out variable. Other variables are described in Table II.

In DVERC, the term $V_{Z_{lead}}$ is the nominal vertical velocity required to follow the altitude profile; in DLONC, the term θ_{lead} is the nominal pitch attitude required to follow the groundspeed profile (4).

The Director control laws (5) are implemented in the Mission Computer, with moding independent of the aircraft AFCS (see Figure 1).

Table II. Director Control Law Inputs

Variable	Description
p, q	Measured A/C roll, pitch rates
ϕ_c, ϕ_{FB}	A/C roll command, feedback
V_Z	A/C earth-referenced vertical velocity
θ	A/C pitch angle
COLL	Pilot's collective control position

Director Cues

Director outputs consist of three cues that drive the MPD symbology shown in Figure 7. Separate vertical situation display (VSD) formats exist for enroute and hover flight. The Director cues are included in the enroute VSD format as shown in Figure 7a. Two of the Director cues (DVERC and DLATC) drive the "phantom aircraft" symbol vertically and laterally, and the third (DLONC) drives the "airspeed cue." The hover format (Figure 7b) presents only a plan view of the helicopter nadir and its acceleration and velocity relative to the desired hover point as an aid to the pilot in attaining and holding hover. The pilot task is to overlay the survivor symbol with the nadir and acceleration symbols (the velocity vector being zero).

Horizontal situation display (HSD) formats do not contain any Director cues; however, they do present useful flight situation data. An example is shown in the decentered, track-up HSD format of Figure 8. This format includes "trend dots" indicating extrapolated horizontal flight path, useful as a basis for lateral steering (6).

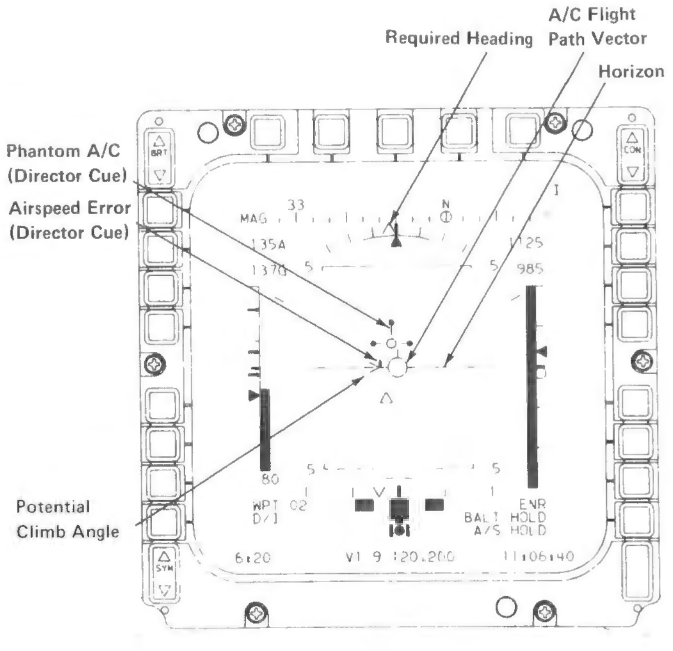


Figure 7a. VSD Flight Path Vector Mode

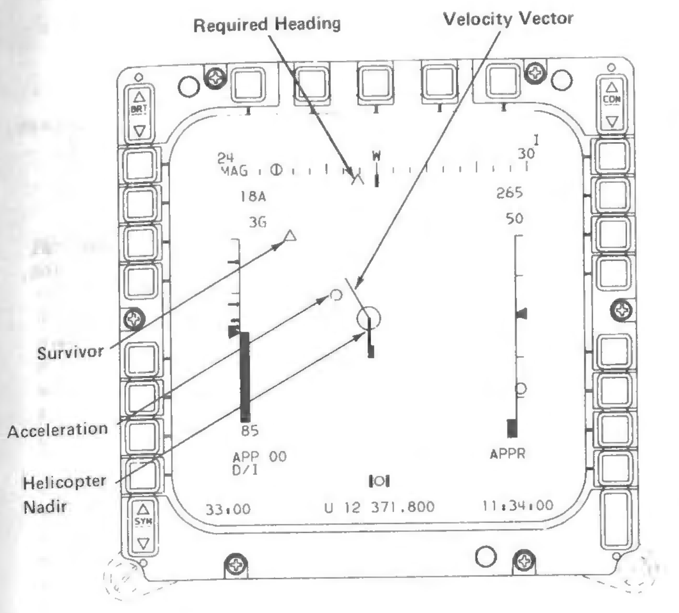


Figure 7b. VSD Flight Path Vector Mode-Hover

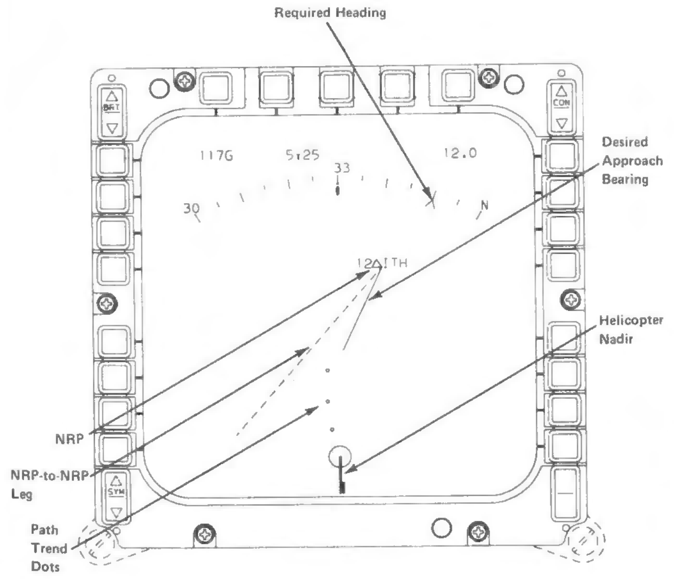


Figure 8. HSD Decentered, Track-Up Mode

Terminal Area Flight

Typically, upon reaching the terminal area, a search for the survivor or a hover point begins. When located, a relative position measurement between the aircraft and the point of interest is made using the MMR or FLIR, to compensate for navigation system error. The Guidance function computes a reference path for approach along a desired groundtrack (e.g., into the wind or along a terrain corridor). The Director function computes the displacement of the active Director mode cues.

Preliminary evaluation of the Guidance and Director analytical designs was accomplished through use of a non-real time simulation employing models of aircraft dynamics, pilot response and environment. A brief summary follows.

Approach-to-Hover

A simulated approach-to-hover from 700 feet altitude and 80 knots velocity to a hover at 50 feet altitude is shown in Figure 9. As indicated, mild initial transients are well damped and the altitude and velocity profiles are closely followed.

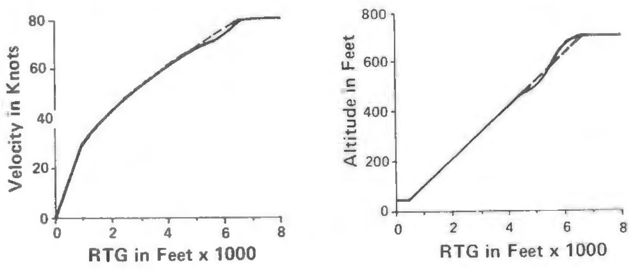


Figure 9. Simulated Approach-to-Hover (APPR) Flight Path

Terrain Following

Simulated constant velocity terrain following over an isolated obstacle with 6 degree slopes is shown in Figure 10. Although the shape of the flight profile varies with velocity, each profile closely approximates the ADLAT goal of achieving zero

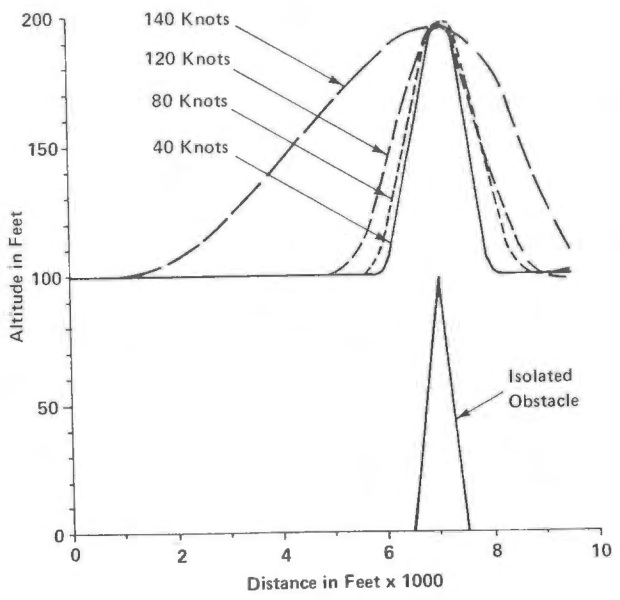


Figure 10. Terrain Following Performance as a Function of Aircraft Velocity

flight path angle at the set clearance height over the peak. For velocities below 120 knots, the geometry of Figure 4 applies. However, at 140 knots, the aircraft incurs climb limiting and an earlier initiation of the climb is required to clear the peak at the set clearance (Figure 5). The 140 knot flight path clearly demonstrates a requirement for an in-flight computation of the aircraft climb capability, as it critically affects the timing of the climb commands. That computation is performed in the Mission Computer.

Summary

The Guidance and Director functions presented in this paper have been verified through use of non-real time simulation. It is anticipated that further advances in generic director/display concepts and implementation will be derived, in part, from a piloted simulator program scheduled for late 1984 at the NASA-Ames Research Center. There, piloted simulation with visual and motion cues (7) will be used to:

1. determine the proper structure of the Director control laws, i.e., appropriate feedbacks and related interpretation of cues,

2. determine Director output assignments to display symbology,

3. attain minimal coupling of multiple Director cues,

4. provide appropriate mix of Director symbols and flight situation data,

5. evaluate compatibility of simultaneous Terrain Following and Approach-to-Hover Director modes.

The HH-60 avionics system will be subjected to flight evaluations at the Air Force's Edwards Flight Test Center, beginning in 1985.

References

1. "Fuel-Conservative Guidance System for Powered-Lift Aircraft," H. Erzberger, J. D. McLean, AIAA Journal of Guidance and Control, Volume 4, No. 3, 1981.

2. "ADLAT Terrain Avoidance Techniques Evaluation," E. C. Schwartz, Cornell Aeronautical Laboratory Report No. AL TDR-64-145, August, 1964.

3. "VTOL Craft Deceleration Control System," J. C. Dendy, R. J. Miller, United States Patent No. 3,916,688, Nov 4, 1975.

4. "Navigation, Guidance, and Control for Helicopter Automatic Landings," J. R. Kelly, F. R. Niessen, NASA Paper 1649, 1980.

5. "Description and Analysis of the HH-60D Guidance and Director Functions," C. A. Mills, F. G. Kilmer, R. L. Kilmer, IBM Report No. 673-2152, March 6, 1984.

6. "The Integration of Control and Display Concepts for Improved Pilot Situational Awareness," L. H. Person, C. G. Steinmetz, 34th International Air Safety Seminar, November 9-12, 1981.

7. "Helicopter Simulation Technology: An Ames Research Center Perspective," R. S. Bray, NASA Helicopter Handlings Qualities Conference, April, 1982.

8. "Automatic Helical Rotorcraft Descent and Landing," L. A. McGee, J. D. Foster, G. Xenakis, Journal of Guidance, Control, and Dynamics, AIAA, Volume 7, Number 4, July-August 1984.

84-2676

APPLICATION OF DIFFERENTIAL GPS TO CIVIL HELICOPTER TERMINAL GUIDANCE

Robert P. Denaro*

TAU Corporation
Los Gatos, California

Abstract

NASA Ames Research Center is conducting a research program to investigate the potential application of differential GPS to civil helicopter operations. The emphasis of current work is on landing applications, which present one of the most difficult accuracy goals. In particular, the vertical axis accuracy requirement, as specified by FAA standards, is the performance driver for any proposed system.

Developmental research in progress has addressed various parts of the system requirements. Presented here are results in three areas; satellite selection algorithm design, ground reference station design, and flight test. Mission-tailored satellite selection algorithms may provide a 10% enhancement in geometric dilution of precision for helicopter operations. Among various alternative ground reference station designs, the range-error architecture appears to be the best tradeoff. Finally, preliminary results from NASA differential GPS flight tests, using current off-the-shelf GPS receivers, are presented. Efforts are continuing to apply advanced processing techniques developed in simulation to the flight test demonstration program.

Introduction

The NASA Ames Research Center is conducting a research program to evaluate differential GPS concepts for civil helicopter navigation [1]. The civil helicopter community will probably be an early user of GPS because of the unique mission operations in areas where precise navigation aids are not available. Many of these applications will have accuracy requirements that are very demanding, beyond that of conventional GPS. Such applications include remote area search and rescue, offshore oil platform approach, remote area precision landing, and other precise navigation operations.

Differential GPS is a promising solution to meeting the increased accuracy needs of the civil aviation community. Besides removing many of the dominant error sources naturally occurring in the system, it has the potential for locally removing much of the effect of the intentional signal degradation to be imposed by the DOD on the C/A signal for national security reasons [2]. Differential GPS used for non-precision approach, or, if possible, precision approach, can be independent of ground aids in the terminal area [3]. Thus, it is a reasonably low-cost enhancement to terminal area operations, probably being an add-on feature to aircraft already configured with conventional GPS.

To determine and demonstrate the feasibility of using differential GPS for terminal guidance, NASA has designated this application as the first target for its differential GPS test program. Earlier results were reported which traded off various implementations of differential GPS in a computer simulation [4]. Current research work is continuing with navigation Kalman filter model tailoring to the landing approach environment for maximum performance. In addition, satellite selection algorithms are under investigation which are optimized for the landing scenario. On the ground side of the differential GPS system, design is underway on a system architecture and algorithms for implementing the reference station range and range-rate differential correction generation process.

Finally, NASA is conducting simultaneous flight tests and ground static tests of GPS navigation with current off-the-shelf equipment to serve as a baseline for the GPS landing experiment. Independent data recording from these simultaneous operations are being combined post-mission to assess differential GPS potential with the current equipment.

This paper presents results in three of these areas. First, results of simulation analysis of GPS satellite geometry dilution of precision (DOP) are presented. The DOP analysis compares conventional selection algorithms with a landing-optimized algorithm. Second, design concepts for a GPS ground reference station are presented. Finally, some preliminary flight test results from NASA's current GPS helicopter flight tests are presented.

Mission-Tailored Satellite Selection

Much has been written on the subject of "optimal" satellite selection for GPS. Techniques generally optimize the geometry of the four satellites needed for stand-alone, continuous GPS tracking. These techniques minimize the position dilution of precision, or PDOP, which is the root sum square of the geometry-induced errors in each of three orthogonal axes. Minimum PDOP, therefore, results in the minimum sum of the squares of the errors in each axis, assuming that the satellite to user range errors are all equal and therefore can be normalized out.

Other techniques have also been proposed, some of which weigh *a priori* known values of each satellite's measurement error variance [5]. In addition, the satellite selection process can weigh observed errors by mathematically inferring their source. Although these more sophisticated techniques clearly make better use of all available information and will probably be significantly better

*Member AIAA, IEEE, ION

if the satellites are not uniformly corrupted, they do pose computational problems. Observation of operational satellite data will decide this trade-off. In any case, geometry-based selection algorithms can be studied independently of range variance-based methods, since geometry methods simply assume optimal performance of any range variance-based method, and thus would augment such a technique.

Of course, the subject of optimal satellite selection can be avoided altogether by employing an "all-in-view" tracking strategy. This technique observes all satellites either continuously (with a multichannel or multiplex set) or sequentially. There are tradeoffs involved in this case also, however, due to the complexity and increased uncertainty of the tracking and switching environment created.

Satellite Selection Algorithm Concept

An immediately applicable geometry-based satellite selection concept is to consider the mission requirements in choosing satellites. This was investigated in the present study. To provide the foundation for this geometry-based analysis, however, a brief derivation of the GDOP (PDOP plus the time term) concept is first presented.

GDOP is defined for a system whose measurements, $\overline{z}$ (the pseudorange errors), are related to the error state, $\overline{x}$, by the expression:

$$\overline{z} = H\,\overline{x} + \overline{v}$$

where

$\overline{v}$ = unmodeled errors (white noise)
H = direction cosines to the four satellites

and

$$E[\overline{v}\;\overline{v}^T] = R$$

For a best linear unbiased estimate of $\overline{x}$ given $\overline{z}$,

$$\hat{\overline{x}} = [H^T R^{-1} H]^{-1} H^T R^{-1} \overline{z}$$

The error in this estimate has the covariance:

$$\mathrm{cov}\,(\hat{\overline{x}}) = E[(\overline{x} - \hat{x})(\overline{x} - \hat{x})^T] = [H^T R^{-1} H]^{-1}$$

GDOP is defined by assuming that the measurement errors are uncorrelated and identically distributed so that:

$$R = \sigma^2 I$$

Thus the error covariance is:

$$\mathrm{cov}\,(\hat{\overline{x}}) = \sigma^2\,[H^T H]^{-1}$$

where GDOP is defined as:

$$\mathrm{GDOP}\;(\mathrm{tr}\;[H^T\,H]^{-1})^{1/2}$$

Note that the GDOP term is a "compromise" between the various components of the $[H^T\,H]^{-1}$ matrix, where a less than minimum DOP value in one axis of a particular constellation's GDOP may be sacrificed (accepted) to avoid selection of a very large DOP value in some other axis of another constellation.

The concept to be investigated is whether or not, for a particular mission application, one may want to weigh certain axes that are more important to mission needs. In the landing situation, the critical axes of concern are the vertical and cross-track coordinates. The vertical axis carries the most restrictive specification in the landing criteria as shown by the FAA Navigation System Accuracy Standards presented in Table 1 [6]. The implication from the table is that the along-track axis accuracy can be relaxed somewhat.

Table 1. Minimum Guidance Accuracy

	Height		Lateral		Vertical	
Category	(ft)	(m)	(ft)	(m)	(ft)	(m)
I	100	(30.5)	30.0	(9.1)	10.0	(3.0)
II	50	(15.3)	15.0	(4.6)	4.5	(1.4)
IIIABC	0	(0)	13.5	(4.1)	1.8	(0.5)

To pursue this possibility, the GDOP matrix was modeled in a "landing mission coordinate frame" with three orthogonal axes in the along-track, cross-track, and vertical directions as shown in Figure 1. GDOP, being the root sum square of all three coordinates, is the same in either coordinate frame, of course. The vertical axis is oriented normal to the local tangent plane, although it could be tilted by the glideslope angle if desired. At a typical 3° glideslope angle, the difference would be negligible.

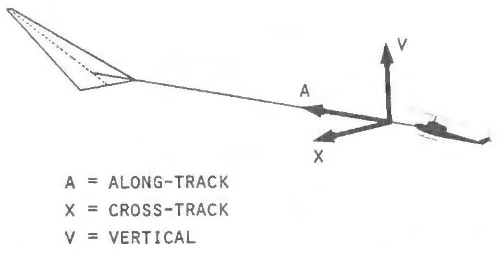


Figure 1. Landing Mission Coordinate Frame

The error covariance in the landing coordinate frame will be:

$$\mathrm{cov}\,(\overline{x}) = \sigma^2[A^{-T}\,H^T\,H\,A^{-1}]^{-1}$$

$$= \sigma^2 \begin{bmatrix} V_A^2 & & & \\ & V_X^2 & & \\ & & V_V^2 & \\ & & & V_T^2 \end{bmatrix}$$

where

V_A^2 = along-track dilution factor

V_X^2 = cross-track dilution factor

V_V^2 = vertical deviation dilution factor

V_T^2 = time dilution factor

A = Rotation matrix from normal GPS coordinate frame to A-X-V frame

A satellite selection algorithm optimized for this frame may seek to minimize V_V or some weighted combination of V_V and V_X.

Satellite Selection Algorithm Simulation Results

The above GDOP coordinate frame rotation was modeled in NASA's DIFFGPS Simulation DOPS Analysis Module. Then, representative areas of operation for the remote helicopter mission were identified, and both conventional minimum PDOP and the modified satellite selection algorithms were executed over a 12-hour period. The satellite constellation was selected as the proposed 18-satellite, 6-orbit configuration [7].

The modified satellite selection algorithm used for these runs is an even-weighted "XVDOP", where the criterion was a sum square of the cross-track and vertical error values:

$$\text{XVDOP} = V_X^2 + V_V^2$$

In addition, a VDOP criterion is analyzed, which minimizes the error only in the vertical direction.

Figure 2 presents a typical plot of dilution of precision over a 12-hour period. The plot presents the "VDOP" (vertical dilution) component values achieved by three different satellite selection criteria. The first criterion is PDOP, position dilution of precision, which utilizes all three axes and is represented by "XYZ" in the figure. The next criterion is XVDOP as described above, represented by YZ in the figure. The last criterion is VDOP, which is the criterion of minimizing only the vertical component, regardless of the values in the other two axes (in this case, cross-track accuracy may suffer).

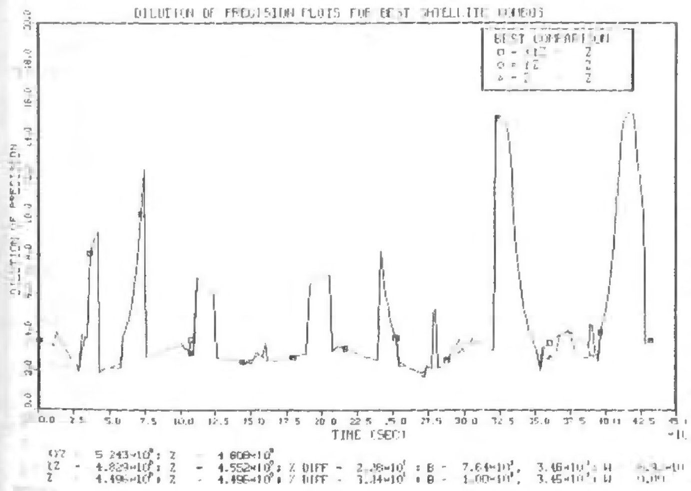


Figure 2. Vertical Dilution of Precision, Seattle, WA

The XVDOP case is dependent on the azimuth orientation of the vertical plane, of course. In all cases presented here, the orientation was normal to the runway heading of the referenced city. In other analyses, not presented here, the sensitivity of satellite selection to azimuth orientation of the vertical plane was studied.

Although differences did exist, they were generally not significant to the VDOP value. Therefore, the random sample presented here is considered "representative."

The statistical results are calculated as follows. First, the PDOP value is calculated. Next, the percentage of time over the 12 hours that VDOP differed from the nominal PDOP selection case is calculated. Finally, the percentage improvement of VDOP, during those periods when it differs from the nominal case, is calculated. The results are tabulated in Table 2.

Table 2. VDOP Statistics for Seattle, WA Case

	Selection Criterion	
	XVDOP	VDOP
Nominal PDOP	5.2	5.2
VDOP:		
Percentage of Time Different from PDOP Criterion	22.8%	32.4%
Improvement Amount when Better than PDOP Criterion	.35	.35

The improvement in VDOP of .35 represents about a 10% improvement, which as indicated, occurs about 23% of the time for the XVDOP selection criterion case and 32% of the time for the VDOP selection criterion case. Table 3 presents results for several other representative locations, for the XVDOP criterion only. It should be noted that VDOP is not always better, when different, in the XVDOP selection criterion case. For these results, it was better on the average over 75% of the time. Better weighting of the cross-track and vertical terms would correct this.

One noteable result illustrated by the plot in Figure 2 is that this improvement occurs primarily when overall DOPs are "good", and not during the VDOP "spikes" that occur due to changing geometry. This result is in general true for all cases tested; this is an unfortunate result since it would be beneficial to find a means to improve VDOP during these "bad" periods.

A possible complementary solution would be to add an altimeter measurement and state. The altimeter's inherent inaccuracies would probably eliminate it's influence during periods where the VDOP is "good," but this is where the XVDOP selection algorithm improves performance. However, in periods where the VDOP is poor and where the XVDOP algorithm was shown to have no effect, the altimeter input is likely to be more heavily weighted thereby substituting its vertical "measurement" for the poorly resolved GPS vertical observation.

Table 3. VDOP Results for XVDOP Selection Criterion

Location	Percent of Time Different	Improvement in VDOP
Fairbanks	14.9	.12
Kodiak	41.5	.24
Cold Bay	19.1	.20
Juneau	45.6	.23
Seattle	22.8	.35
Seattle W.	21.2	.26
Portland	58.1	.24
Portland W.	70.5	.17
SFO	7.9	.32
SFO W.	7.9	.26
L.A.	11.2	.23
L.A. W.	11.6	.21
Bangor	18.7	.29
Bangor E.	18.7	.25
St. John	12.0	.30
St. John E.	12.4	.27
Average	17.4	.25

W. = 100 miles west of city
E. = 100 miles east of city

Differential GPS Ground Station Design

A differential GPS system relies on a ground-based reference station for the computation and data link transmission of the differential corrections. Although other differential GPS techniques are in study as well (e.g. the translator concept and the "pseudo-satellite" concept), the current emphasis is on the data link concept.

TAU Corporation has been involved in the development of algorithms and system concepts for differential GPS ground reference stations for several years. The ideas presented in this section are highlights of more detailed work performed for NASA, the Air Force, and other Government Agencies as well as future commercial operators of differential GPS.

Overall Reference Station Concept

The differential GPS reference station has three basic functions:

1. Compute GPS measurements
2. Compute differential corrections
3. Format and transmit corrections to users.

These functions have several variations in modes and rates at which they can be performed. The tradeoffs in these issues are resolved by an anlysis of the mission requirements,user equipment in use, and GPS signal error characteristics.

GPS Measurement Computation

Computation of GPS measurements can proceed as simply as a standard receiver. However, since different users to whom the differential corrections are being sent may be navigating off different sets of satellites, the reference station must track all satellites in view. The need for corrections to different satellite sets is quite likely to occur. This is due to different user satellite selection algorithms, independent user maneuvers which mask the view of otherwise desirable satellites, and user structural blockage of particular satellite signals at varying times. There still may be some mismatch of satellite tracking between user and reference station for those users who can "see over the horizon" by virtue of their altitude or distance.

In addition to tracking all satellites in view, the reference receiver may have other unique features. Because it is static, the doppler frequency shift of each satellite's signal can be calculated very precisely. Hence, tracking loop bandwidths may be narrowed for higher signal-to-noise operation. Knowledge of the navigation message for each satellite may be used to demodulate the 50 Hz. data for simpler tracking loop design.

Differential Correction Computation

Differential correction computation can vary in complexity. The simplest method involves differencing of range data. In this case, the receiver makes pseudorange measurements, PR_i, to each satellite. The best estimate of the true range to the satellite can be obtained by differencing the broadcast ephemeris position, $\overline{S}_i$, with the known, static receiver location, $\overline{U}$:

$$\triangle PR_i = PR_i - \left|\overline{S}_i - \overline{U}\right|$$

Note that $\triangle PR$ is a pseudorange correction, not a range correction, since it still contains the unknown reference receiver clock bias. For most users this is immaterial, however, because this constant bias on all corrections would be absorbed by the user clock bias estimate.

The only problems with this formulation are that for sequential users, a non-constant reference receiver clock bias will be observed, and the corrections may be overly "noisy" due to short term variations caused by independent reference receiver noise errors. To alleviate these effects, one may choose to optimally filter the differential range error estimates. In this case, the filter could also estimate reference receiver clock bias and remove it in the correction formulation process.

This concept has assumed that a user incorporates differential range corrections by subtracting them from his own raw range measurements, prior to Kalman filter processing. This is the most efficient way to implement differential GPS since with a maximum of 8 satellites in view, the reference receiver need transmit only 8 corrections which cover all possible combinations of satellites that could be in use (e.g. there are 70 possible 4-satellite combinations with 8 in view). Alternatively, the reference receiver could calculate sets of navigation coordinate corrections (e.g. x, y, z) for each possible 4-set combination. A user would then subtract these corrections from his post-filter navigation solution. Although the transformation from the "range domain" errors to the "navigation domain" is a simple multiplication by the inverse of the GPS satellite observation matrix, there are potentially a large number of such transformations to be performed at any point in time.

Still, the use of navigation domain differential corrections is not totally without merit. Simple, stand-alone GPS receivers may not allow access to the input side of the navigation filter for range correction. Instead, a simple add-on component may perform the correction after GPS solution output from the receiver. For certain constrained, limited user applications, this method may be preferable.

Differential Correction Formatting and Transmission

The Department of Transportation and the Radio Technical Commission for Marine Services (RTCM) have devised a proposed differential correction message format [8]. This format provides flexibility to accommodate most users. The differential corrections generated by the previously discussed process would be appropriately formatted and transmitted.

One further function that would be desirable prior to message transmission would be some sort of differential correction integrity management check. The transmitted corrections should not be allowed to corrupt a user's solution. Built-in test features and internal data analysis checks should be able to validate the corrections before transmission. Alternatively, a second reference station could be used to cross compare corrections for further insurance.

A system concept that incorporates most of the above described features is shown in Figure 3. The system includes continuous all-in-view tracking (multiplexing 8 satellites on 4 channels), filtered solution of differential corrections, integrity management, data formatting and transmission.

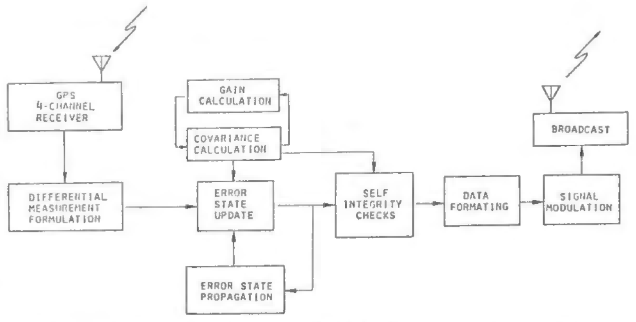


Figure 3. Reference Station System Concept

Preliminary Differential GPS Flight Test Results

In parallel with their research and development activities, NASA is conducting a flight test program to investigate the potential of GPS to support landing operations. While the flight test program is ultimately planned to demonstrate the implementation feasibility and performance capability of the advanced algorithms and concepts in development, these preliminary tests are invaluable for setting a baseline against which future results can be compared and for shaking out the test procedures and instrumentation system.

Tests conducted to date have involved two GPS receivers. A Phase I Magnavox Z-Set has been in use at NASA for several years, and is operating aboard the SH-3 helicopter. More recently, NASA obtained a Trimble Navigation, Inc. GPS receiver which is being used in a static location much as a differential reference receiver would be operated. At the present time no data link is available for transmitting differential corrections; nor is there any means for the Magnavox set to process these corrections. However, simutaneously collected data from both of these receivers are being processed in TAU Corporation's GPS Simulation and Data Analysis Facility, including generation of filtered conventional and differential GPS solutions. These results can be rationally extrapolated to results expected in a full-up real-world environment test using these receivers.

Figures 4 and 5 depict the lateral and vertical profiles flown by the SH-3 on a flight test on June 14, 1984. The mission was a landing approach to Runway 17 at the Navy Crows Landing Facility in California. The plots are from combined radar and laser "truth" data.

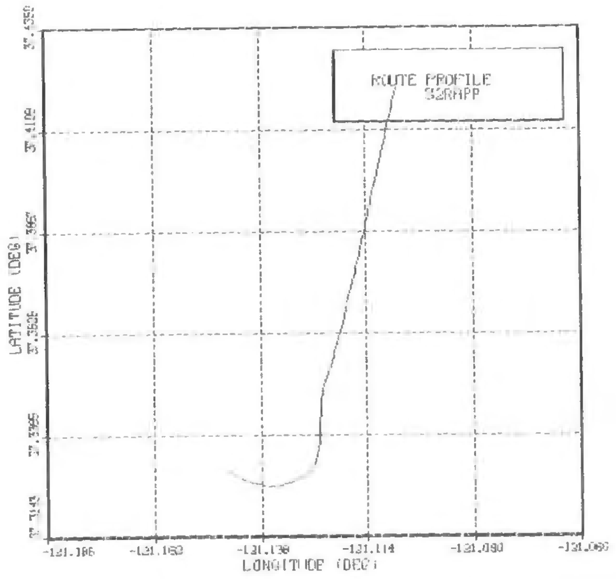


Figure 4. Lateral Flight Test Profile

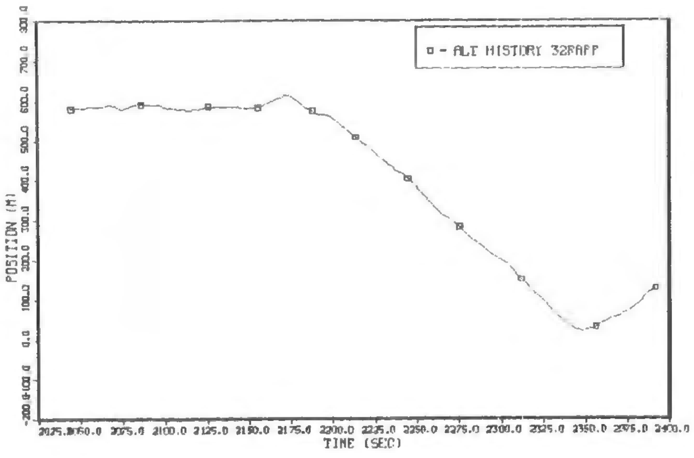


Figure 5. Vertical Flight Test Profile

Unfortunately, in these preliminary tests, the processing of the static receiver data was suspect due to improper field recording of the satellite ephemeris data. The errors manifested themselves primarily in the horizontal axes. The vertical data appeared relatively insensitive to this problem, although it is reasonable to assume that some corruption of the vertical data occurred from the erroneous horizontal data. Nevertheless, the vertical data do give some positive insight into the performance of differential GPS navigation for this run. Figure 6 is a plot of a 5.5 minute stretch of data from the on-board Z-Set. The set demonstrated a negative 16.7 meter average error over this period, with a standard deviation of 9.4 meters.

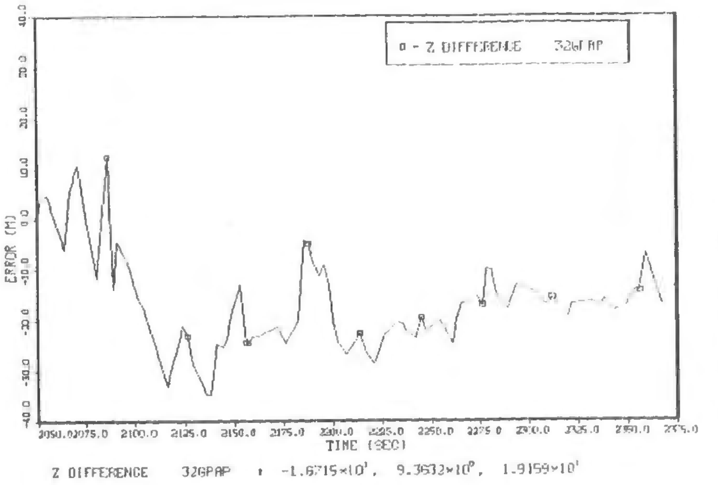


Figure 6. Vertical Error, Airborne Z-Set

Figure 7 is a plot of the static Trimble Set data for the identical period of time. This set demonstrated a negative 16.3 meter average with a 9.5 meter standard deviation. The different signature of the Trimble data from that of the Magnavox data is due in part to the different navigation algorithms on each set. The Magnavox set uses an 8-state Kalman filter to sequentially process the GPS pseudorange and delta range signals at about 1.2 second intervals. The Trimble receiver processes measurements in a simple least squares filter and solves for position by averaging over a six-second interval. The period of largely uncorrelated errors in the first 50 seconds is probably due to the helicopter Z-Set errors induced by the turning dynamics prior to final straight-in approach. The static receiver would not observe these motion-induced errors.

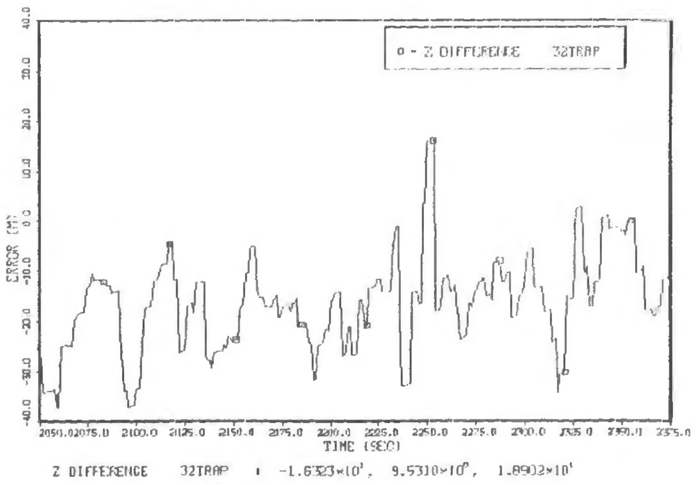


Figure 7. Vertical Error, Static Trimble Set

Differencing these two data sets results in the plot in Figure 8. The mean error has been reduced to 0.4 meters, but the standard deviation is now 13.5 meters. The result is quite typical of differential GPS performance; bias terms are eliminated but independently generated receiver and other noise will further corrupt the short term accuracy of the receiver. The noise impact is perhaps exaggerated in this case due to the very different navigation solution techniques employed by the two receivers. Clearly the next step in analyzing these results is to work with the pre-navigation filter range measurements in each case, process them in comparable Kalman filters, and compare the results. This work is in progress.

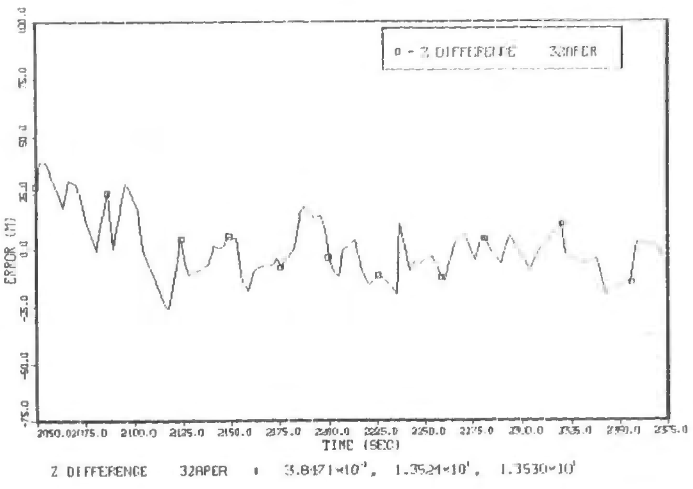


Figure 8. Differential Error

This brief example of actual flight test data supports analytical conclusions regarding the performance of differential GPS, even for C/A-code receivers. Further tests are scheduled to document total errors over several missions, thus establishing a baseline for future tests of advanced algorithms.

Conclusions and Future Study

The mission-tailored satellite selection algorithm may provide moderate improvements during good GDOP periods. Coupled with altimeter aiding, this technique provides the greatest promise for performance enhancement in the vertical dimension. Further improvements will include weighing the vertical and cross-track axes and incorporation of measurement noise-based algorithms.

The ground reference station is most efficient in a range error configuration for multi-user applications. Calculation of local oscillator bias preserves maximum accuracy of correction for sequential tracking users. Differential correction integrity monitoring is a necessary feature of differential GPS systems used for critical guidance. Optimal filtering techniques are in development to maximize processing efficiency and correction accuracy.

The ultimate verification of these developments will be in flight test. Preliminary tests with very rudimentary comparisons of dissimilar GPS receivers have been promising. Documentation of further flight tests is in progress, and the next phase will demonstrate the advanced processing algorithms in flight test.

Acknowledgments

This work was performed in part under sponsorship by the NASA Ames Research Center. The author is indebted to Mr. Fred Edwards of NASA who provided all of the flight test data and much of the inspiration for the simulation work. Also, a special note of thanks to Ms. Algyte Cabak who performed all of the software development involved in these efforts.

References

1. Edwards, F., and Hamlin, J., "Operation of a Single Channel Sequential Navstar GPS Receiver in a Helicopter Mission Environment", American Helicopter Society, May 1983.

2. Beser, J. and Parkinson, B., "Application of Navstar Differential GPS to Civil Helicopter Operations", NASA CR 166169, June 1981.

3. Cardall, J. et al., "Civil Application of Differential GPS using a Single Channel Sequencing Receiver", NASA CR 166168, May 1981.

4. Denaro, R. P., "Simulation and Analysis of Differential GPS", Proceedings of ION National Technical Meeting, San Diego, CA, January 1984.

5. Brogan, W. L., "Improvements and Extensions of the Geometrical Dilution of Precision (GDOP) Concept for Selecting Navigation Measurements", AFWAL-TR-81-1142, September 1981.

6. "Federal Radionavigation Plan", DOT-TSC-RSPA-81-12-1, March 1982.

7. Brady, W. and Jorgensen, P., "Worldwide Coverage of the Phase I Navstar Satellite Constellation", ION Proceedings, April 1981.

8. Kalafus, R. et al., "Navstar GPS Simulation and Analysis Program", DOT-TSC-RSPA-83-2, May 1983.

84-2621

PILOT COMMAND INTERFACES FOR DISCRETE CONTROL OF AUTOMATED NAP-OF-EARTH FLIGHT

S. Joy Mountford,*
Robin Penner,
Paul Bursch

Honeywell, Systems & Research Center
Minneapolis Minnesota
Ms. Mountford is currently a Member of the Technical Staff at M.C.C., Austin, Texas.

Abstract

Current helicopter design trends indicate a need to develop a single pilot crewstation. In turn, this creates a need to develop new pilot command interfaces. The advent of automatic terrain following, terrain avoidance algorithms may ultimately enable parts of the flight control loop to be performed automatically. Furthermore, such algorithms may be able to outperform the pilot, but the pilot will still need to have supervisory control to update or correct the flight path. This paper describes an experimental evaluation of different interface designs to transition a pilot in and out of an automated flight loop.

Three interface concepts were tested combining force positioning control, voice recognition and pointing positioning technologies to enable discrete control of continuous flight trajectories. Testing occured within an out-of-window gaming area simulation. Performance scores indicated that force positioning coupled with voice recognition, showed the best accuracy and speed of update to the automated flight trajectory. Traditional methods of continuous flight control are no longer sufficient to accommodate new automatic flight control techniques. This paper investigates some initial interface design concepts for control of automated continuous flight control.

Introduction

Workload levels are excessive in many crewstations, especially for the two-man crew of an attack helicopter performing nap-of-the-earth (NOE) flight. In order to reduce training time, weight and system costs, it would be beneficial to reduce the crew size to that of a single operator. Preliminary workload studies performed by Honeywell on single pilot mission scenarios of scout-attack helicopters for the Army's Advanced Rotorcraft Technology Integrator (ARTI) program (1) indicate that multi-task performance during continuous flight control can only be achieved by automating some tasks. One of the prime candidates for automation could be some features of flight control. During flight control activities, simultaneous attention demands to perform other system management tasks are high, so automation could facilitate these time-shared performances. Automation of NOE flight is a challenging engineering task that will lead to many system enhancements. In order to achieve this goal while ensuring safe workload levels for a single pilot, many integrated advanced technologies must be considered for interface design.

The current trend in crewstation designs is to place the pilot in the role of manager, where continuous monitoring and direct manual control of flight activities are replaced by the pilot making executive decisions. This type of trend is illustrated in a recent Honeywell contract to develop an automated terrain following/terrain avoidance (TF/TA) algorithm for the F-15. (2) This algorithm generates a real-time optimal 20 sec flight trajectory, calculating the best flight path between navigation points, which avoids the terrain while following the contours. These types of optimization algorithms can ultimately outperform the response speeds of the pilot and could be more effective than a pilot for continuous flight control.

The issue of how best to interface the pilot with such a flight control system is yet to be addressed. Clearly the complex-

*Member of IEEE

ity of developing similar algorithms for the control of helicopter NOE flight is much more difficult, since the helicopter is much closer to the ground than the F-15 and unexpected obstacle avoidance is critical. With the advent of such technological enhancements as automatic TF/TA algorithms, the possibility of automating some of the helicopter flight control activity does exist. However, even with such an automated system, a pilot will still need to have the capability of overriding automated flight trajectories to accommodate changes in direction, and to avoid unexpected obstacles. The utility of such algorithms to aid in reducing pilot workload will only be appreciated if the pilot can operate effeciently and cooperatively alongside them. The important human engineering design issue is how best to design the control interface to allow the pilot to reenter the flight control loop to update flight objectives, and then effectively exit the loop while ensuring speed and accuracy during flight conditions.

Approach

Currently, the most similar type of implemented automated flight control system is the terrain following system on the F-111 which uses a "dead-band" technique for pilot handoff where, outside the specified dead-band, the pilot overrides the automatic system and flies manually. Clearly such a system would be inappropriate for a flight control system that can outperform the pilot, since it would result in handing pilots a flight system beyond their control capabilities. In other words a new command language needs to be evaluated for bringing the pilot efficiently in and out of the control loop to enable refinement of the flight trajectory. Such a candidate interface must be designed to enable discrete control of the flight path, instead of actually controlling immediate continuous vehicle position. The design principles that must be adhered to for effective interfacing are to maintain natural stimulus-response compatability, reduction of workload, reduction of training time, provision of a continuous range of pilot involvement, while still being generalizable to other modes of control.

Our command interface design goal attempted to implement unconventional control technologies for discrete flight command updates. The necessary pilot interactions with such an automated TF/TA flight system would require two pilot associated control management tasks: (A) Flight Course Repositioning and (B) New Course Activation. The flight course repositioning task, (A), requires a cognitive assessment by the pilot that the existing flight trajectory direction needs updating based on new flight data, i.e., change in mission objective(s). The pilot must be able to input to the system the direction and position of this mission objective change, but still allow the algorithm to calculate the exact best path to achieve the ultimate goal. A location needs to be specified, not the path, a task analgous to positioning a navigational waypoint cursor flag. The second task, B, involves having placed the navigation marker in the correct place, activating the system (algorithms) to transition to the new course objective, an initiation of a modification to the flight management system. This implies that the interface must be able to receive pilot inputs both for position location objective, and discrete initiation of that update command.

Interface Designs

In this preliminary experiment three different interface methods were designed to transition a pilot in and out of an active flight control loop. The interface designs combined three main communication techniques; Force hand controller positioning, voice recognition commands and light pointer positioning. In order to accomplish task (A) repositioning could be performed using a rigid force hand controller, or by sensing light pointer position. Task B, course activation, used either voice recognition commands or manual discrete entry. These technology interfaces were combined to create 3 different interface design concepts, as shown in Table 1.

Pilot/ System Interface	PILOT MANAGEMENT TASK A) Flight Course Redirection	B) New Course Activation
1	Force Hand Controller	Voice Recognition
2	Force Hand Controller	Manual Discrete
3	Light Pointer Positioning	Voice Recognition

Table 1 Interface Designs

The manual force hand controller provided the natural compatibilities of a conventional joystick with coupled axes of control for exact positioning of a cursor. The Force hand controller consisted of rigid controller mounted on the right side of the pilot. The other positioning device used a hand-held light gun pointing device, which optically traced out cursor position while the device was held at arm's length. A VOTAN V-5000A voice recognition system was used for voice activation, as a speaker dependant isolated word voice system. Speech recognition has indicated some performance enhancements of discrete tasks in eyes and hands busy, timeshared task environments, and further facilitates the amount of available heads-up flight time (Mountford, Schwartz, 1983) (Mountford and North, 1980).

The three interfaces for automatic flight control interactions used the previous technologies as follows: Interface 1 used the right hand continuous force controller to position a cursor, and voice recognized commands to initiate the activation of the new desired location. Interface 2 coupled force hand control positioning with activation of new position positioning by a discrete manual selection switch on the force hand controller. Interface 3 used a light pointer positioning device for flight course redirection location, and voice recognized commands for new course activation. This creates an interface analogous to a car passenger providing desired directions by verbal commentary, and pointing gestures to the driver.

A standard-comparison interface was designed to allow continuous manual control entry in a conventional flight control mode. This was used to assess the effectiveness of the preliminary discrete control interfaces against traditional continuous manual flight modes. In order to perform experimental testing comparisons, an entirely manual flight condition was included. This interface used a springloaded, center-return joystick mounted centrally in the cockpit.

Simulation

A simulated nap-of-earth (NOE) environment was designed for pilot control. A real time vector graphics block world of geometric planes of rectangular object blocks was drawn in front of a pilot subject, between which the pilot had to steer. The design objectives for the visual simulation were to constantly demand pilot intervention into the control loop in order to avoid or hit unexpected targets of opportunity. This visual task had to push the limits of human speed and accuracy capabilities to be typical of NOE flight conditions, and the reaction times required to avoid unexpected obstacles.

A static frame of the out-of-world view to be navigated through is shown in Fig. 1. The ground is represented as a flat ground plane with equally spaced grid lines. The blocks or towers are assumed to be planear and are placed across the course as planes of tower presentations which vary in width and height, drawn as overlays on the world. The visual scene moved past the subject at a constant fast velocity, which necessitated steering through two of the three obstacles in the vicinity of the immediate flight path course. The best flight path

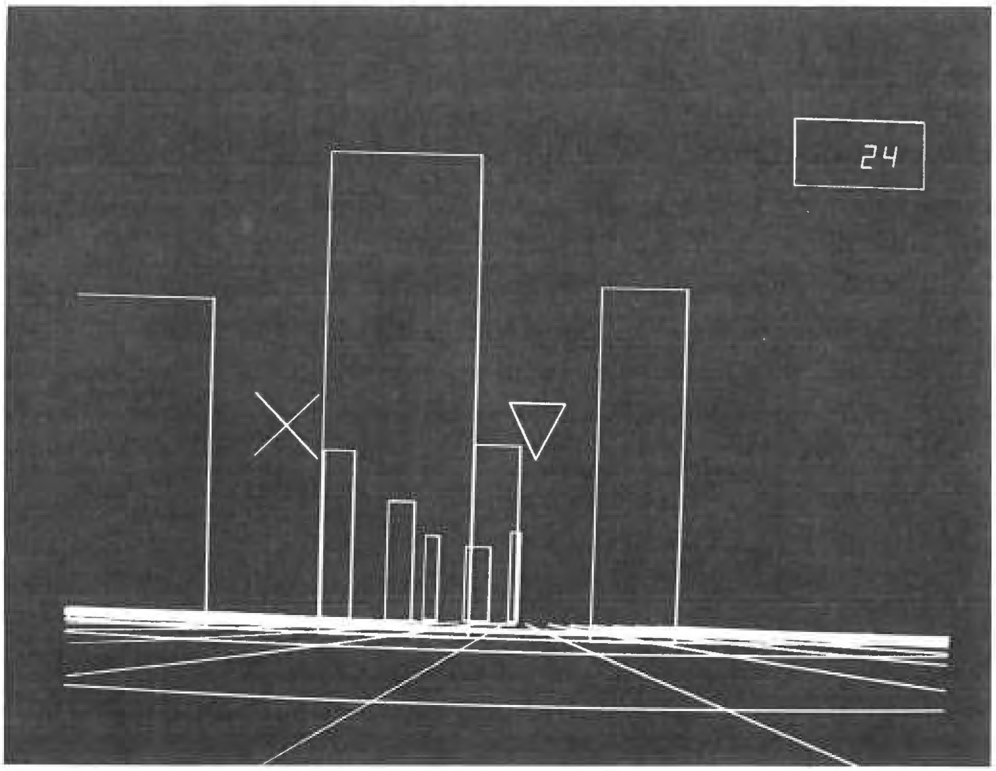
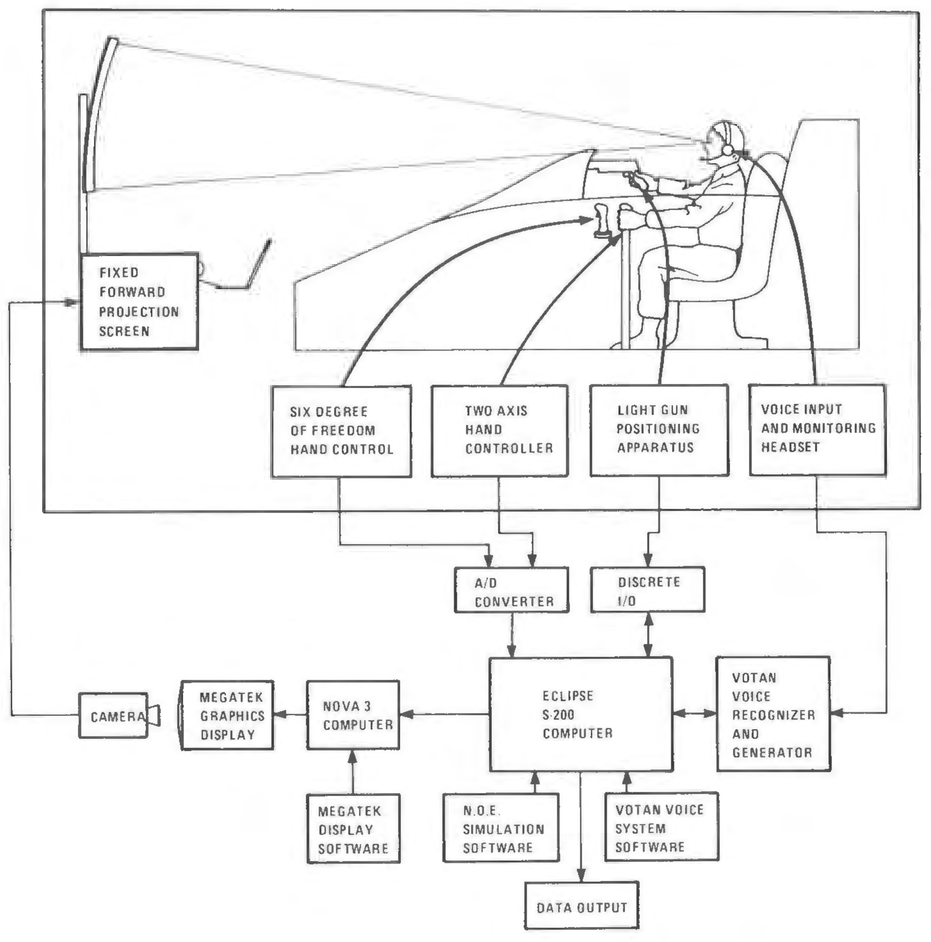


Fig. 1 Out-of-window simulated visual task environment



FIXED FORWARD PROJECTION SCREEN
SIX DEGREE OF FREEDOM HAND CONTROL
TWO AXIS HAND CONTROLLER
LIGHT GUN POSITIONING APPARATUS
VOICE INPUT AND MONITORING HEADSET
A/D CONVERTER
DISCRETE I/O
CAMERA
MEGATEK GRAPHICS DISPLAY
NOVA 3 COMPUTER
ECLIPSE S-200 COMPUTER
VOTAN VOICE RECOGNIZER AND GENERATOR
MEGATEK DISPLAY SOFTWARE
N.O.E. SIMULATION SOFTWARE
VOTAN VOICE SYSTEM SOFTWARE
DATA OUTPUT

Fig. 2 Single Pilot N.O.E. crewstation simulator

Results

Data were averaged from 10 subjects and illustrative graphs of the performance data are shown in Fig. 3. Formal statistical analyses were not performed due to the small sample involved, (only 240 observations per interface type for each subject). However, standard error bars are drawn illustrating the reliability of the data in Fig. 3. The light gun/voice interface 3 shows high standard errors with the largest distance off optimal path and highest percentage of defaults and incompatible responses. Clearly, there were some operational problems with the light gun. It was not as sensitive a device as needed to operate on a screen 15 feet away from the operator, many random noise inputs entered the system, causing the light gun to move somewhat erratically. This technique should be further explored with pointer operated against a fixed surface rather than as a free-movement extended arm motion. Modifications to the hardware need to be made before meaningful performance comparisons of this interface can be made. Data for this interface are included, but must be interpreted with caution.

Comparisons of the two new discrete command interfaces, using the force/voice, force/-maunual, and force controller positioner, showed best overall performance , voice initiated activation showed superior performance. There were no incompatible position selections by voice, and fewer positioning errors by voice. It seems that voice recognized initiation with the force controller is preferable to manual switching for course activation. Voice activation is more accurate than manual switching, but has a slight speed disadvantage, since it takes physically longer to say something quickly than to press a manual switch. The speed advantage for manual activation would be expected to break down during multi-task scenarios, where multiple manual operations are interfering and can further degrade manual performance.

along the terrain was calculated by a simplified TF/TA algorithm, which was insensitive to the "unseen" block obstacle courses, simulating "unexpected" obstacles. In order to steer around these obstacles the pilot had to position a cursor represented by the "X", to a desired optimal location, and fix the position using a discrete command.

The gaming area simulation was programmed with threats and targets of opportunity, within each plane of obstacles/towers. A different obstacle plane was encountered every 6 seconds continuously throughout a 2 minute trial.

The threats were represented by an inverted triangle ▽ which indicated that the path should pass to the immediate opposite side of the prompt. The targets were represented by an upright triangle △ , which indicated that the flight path should pass as close as possible to the middle of the prompt/triangle. Only one type of target of opportunity appeared at each obstacle plane viewing. The pilot's objective was to earn bonus points based on course redirection accuracy in a game-like scenario. Subjects earned bonuses up to 3, based on how close they were to the optimal cursor placement position, (left or right of the central obstacle), and lost 4 points if the path was redirected to the incorrect position. The overall mission was to avoid threats, while engaging targets. A running score throughout each 2 minute trial was presented on the screen, to provide performance feedback and incentive to the user, 24 points in Fig. 1.

Under the three different automation management interface conditions, the pilot was required to interrupt the automatic flight path trajectory and update it when either a threat lay in the computed path or a target was presented outside of the best computed path. Intervention into the automatic system required two steps: (1) indicating the redirection position, and (2) activating the new course. The automatic flight path algorithm then calculated the best flight path to the new desired course location indicated by the position of the cursor. The intervention technique used one of the three previously described discrete interface designs. In the traditional manual comparison mode, the subject pilot had to continuously control the flight path, using the center joystick to steer through the blocks, to avoid threats and hit targets, without using a discrete control cursor positioner.

The visual graphics information was presented via a fixed forward projection screen to the subject pilot sitting in a cockpit equipped with the four different control interfaces. The simulation hardware is shown in Figure 2. The Eclipse S200 computer is responsible for executing software which gathers the cockpit's signal data via digital to analog and discrete voltage converters. The Eclipse also executes the simulation software which generates, controls and records the subjects performance data. Graphics signals are sent from the Eclipse to a Nova 3 computer where they are processed by the graphics display software and relayed to the megatek graphics, display which interprets the signals to generate vector graphics lines on the screen. The controller and light gun positioning system are interfaced to the Eclipse S200 providing a closed loop simulation.

In summary, the subjects task involved updating the position of the automatic flight trajectory (TF/TA) based upon the prompts presented. The pilot did <u>not</u> have to fly the vehicle to the correct location, but merely position the cursor to the desired goal and initiate the path algorithm by a discrete input to steer the vehicle to the new path from the current location. Since the vehicle was moving at a constant velocity, the pilot was required to continually make discrete, quick, update decisions, otherwise the system would ultimately crash into a block. A crash could occur when a flight path was chosen too late or too far away from the current location to respond to the update within the standard velocity. Before a crash, a collision prompt was presented, then a destruction image shown, alongside a loss of 4 points, but the visual simulation program continued to another plane of blocks.

All 10 subjects tested were males in their mid-twenties and right-handed, selected from engineering professionals. They were tested on each of the four interface types over 2 separate testing periods. A repeated measures within subject design was used, blocking interface type trials following training and familiarization with the task simulation world. There were a number of different flight courses available with different obstacles and prompt placements, which were randomly presented across subjects and interface conditions. Two different constant velocity speeds were used to see if increasing velocity differentially affected command interface performance.

A variety of performance scores were collected during each 2 minute trial across the four interface types. Performance measures included: (1) overall cumulative course score earned during each trial (based on number of targets acquired and threats avoided), (2) distance from the optimal flight path, (3) time taken to make a course alteration, (4) numbers of incompatible responses/errors.

The force controller configuration showed additional positioning errors, compared with voice activation. It appears that the use of two manual modes of control on the same controller enters noise or cross-talk into the positioning system and increases position errors. It is important to note that subjects do not say the incorrect thing, as no imcompatible responses were made, which is an important design advantage for voice recognition implementation. The number of crashes (n) does not show an increase in the selection responses between voice over manual conditions during faster simulation times (solid lines in Fig. 3). In the slower condition, since voice recognition takes longer to be uttered, it does not tolerate last minute changes as well as manual switches.

Referring to the graphs in Figure 3 it is clear that the interfaces using the implementation of the automated TF/TA algorithm improved the accurancy of flight path movements, reduced errors, and enabled the pilot to have more time to respond than when performing continuous flight control. However, the overall score was highest for traditional manual continuous center stick control, where it seems that more aggressive responding occurred--if the response was correct it was a close or good score, but more incorrect responses (wrong direction) were made, although not bad enough to result in a crash to lose points. With conventional center stick control, 2-3 more seconds were used to respond to the desired path since the pilot was involved continually in the flight control loop until the object blocks were passed. Time is a highly priced commodity during high workload segments, which are predicted to occur in single pilot attrack operations. It appears that implementation of automatic TF/TA algorithms may help to reduce visual-manual pilot workload, offloading the operator to perform other activities. This particular advantage for automatic TF/TA algorithms is of considerable importance in impacting future crewstation designs.

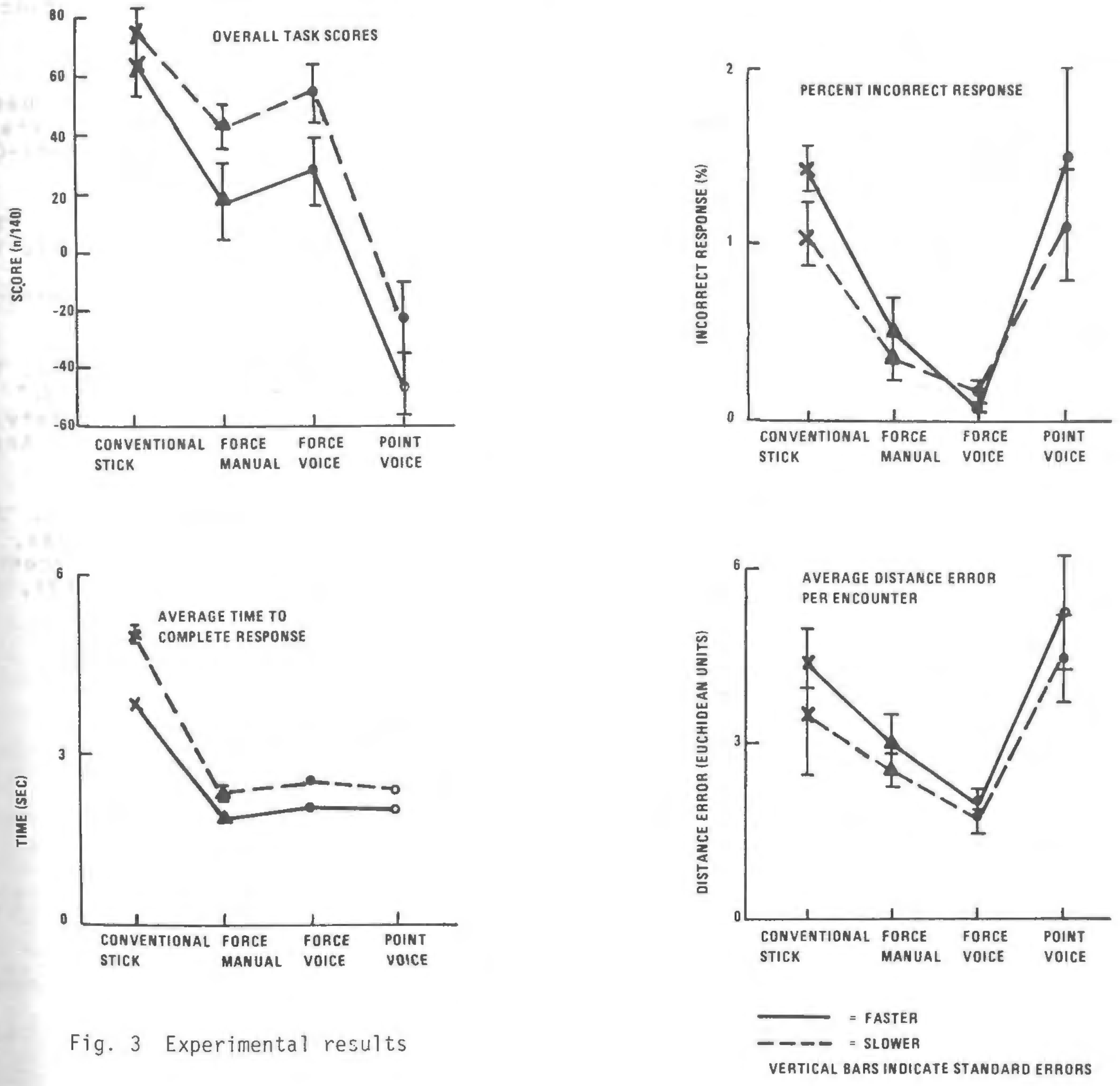


Fig. 3 Experimental results

CONCLUSIONS

The results of this study indicate that the integration of automated flight control algorithms may facilitate pilot performance, especially in time-shared task environments. The use of automated flight trajectory calculations enables a pilot to have more time and attentional resources to direct to other time -shared system management activities. It appears that the use of automated flight control may also facilitate overall system performance, and that an innovative design approach must be taken to create the best overall discrete command interface configuration. Handing a pilot a continuous flight control system beyond his control capabilities is clearly inappropriate, so a new discrete command language needs to be developed for effective interaction with an automatic flight system.

Of the discrete command interfaces investigated, the use of the force controller for positioning coupled with voice recognized commands for new course activation, showed the best overall performance. A natural interface metaphor was created, and produced fast response times, accurate distance positioning and high task scores with almost perfectly accurate responses. The use of the force controller with manual course activation was an alternative interface candidate, but less accurate due to potential cross-talk of manual operations. Many time-shared flight task activities rely on simultaneous multiple manual entries, which would even further exacerbate performance degradation of the manual modality. This experiment supports a cognitive model of multiple task activities that shows the more different two time-shared tasks are, the more easily they can be performed simultaneously. (Wickens, Mountford, Schreiner, 1981) (5)

It appears that the coupling of voice command and force positioning is an interface design concept worthy of further analysis, within a multi-task environment, which must be expanded to include secondary tasks to further workload the pilot. The utility of the light pointer positioning interface should be further explored using other technologies such as a touch screen surface, or the use of an oculometer for eye positioning instead of arm pointing. The use of the TF/TA algorithm needs to be futher investigated within the context of pilot transitions between discrete command control and the traditional manual continuous flight control.

The results of this study offer a preliminary attempt to develop some command language interface concepts for integration with an automatic flight control system. The integration of automated flight control may further facilitate pilot/vehicle performance to help achieve the goal of single pilot operability outlined in the Army's ARTI plans. The results clearly show that the discrete control update of an automated flight trajectory is feasible, and that it offers much potential for permitting pilot transitions in and out of an automated flight control loop. However automated or "smart" the flight control loop becomes, it will always need to receive some degree of pilot updates, otherwise the path will continue to run in a default mode when no other data is available. The design of a discrete command interface for a continuous flight systems needs to be addressed and quantitatively explored if crewstation designs are to keep pace with the automation trends of flight control.

Reference

1. Wendl, M. J., Wall, J. E., Young, G. D., "Advanced Automatic Terrain Following/Terrain Avoidance Control Concepts." American Institute for Aeronautics and Astronautics Guidance and Control Conference, August, 1981, Albuquerque, NM.

2. Wald, J. (Ed.) Refined Baseline Avionics Display and Control System for LHX-Armed Scout Contract DAAK50-81-C-0038 for AVRADCOM, May, 1982.

3. Mountford, S. J., Schwartz, J., "Speech Technology Enhancements of Automatic Target Recognition" in Proceedings of Fifth Digital Avionics Systems Conference, November, 1983, Seattle, WA

4. Mountford, S. J. & North, R. A., "Voice Entry for Reducing Pilot Workload." Proceedings of the Human Factors Society, 24th Annual Meeting, October, 1980, Los Angeles, CA.

5. Wickens, C. D., Mountford, S. J. and Schreiner, W., "Multiple Resources, Task-Hemispheric, and Individual Differences in Time-Sharing." Human Factors 23(2), 1981, pp 211-229.

SESSION 13 DATA BUS CONCEPTS AND PRACTICES

Chairman:

Roger M. Hartman
McDonnell Douglas Astronautics Co.

Manvel A. Geyer
Westinghouse Electric Corp.

This session presents new concepts in data bus implementation and application to digital avionic systems, as well as the effectiveness of existing systems.

84-2679

A STANDARD COMPUTER BUS FOR MIL-STD-1750A AVIONICS COMPUTERS

By David Penn, Shmuel Levy, and Elad Loker.

(David Penn and Shmuel Levy - Israel Aircraft Industries Ltd. Israel
Elad Loker - Elbit Computers Ltd. Israel).

ABSTRACT

While MIL-STD-1750A describes an instruction set architecture (ISA), the application of this ISA requires the usage of a data and address bus system which permits efficient communication between the cpu, memory, and application oriented input/output devices. The data and address bus system design and implementation is influenced by the design of the cpu and main memory since these two devices, in general, are the main users of the bus system. The Lavi avionics system utilizes a standardized data and address bus system (called L-BUS) for use in the MIL-STD-1750A computers which are embedded in the various components of the avionics system. The L-BUS is described and is proposed as a potential standard bus for MIL-STD-1750A implementations.

INTRODUCTION

Numerous standard computer interfaces have been defined over the past fifteen years to interconnect the major component modules of minicomputers, and later of microprocessor systems. These interfaces have usually been defined by commercial computer equipment manufacturers in order to increase the configuration possibilities of their own product range, and to promote upgrade capability and future expansion. Typically, a manufacturer introduced a new cpu module, with a choice of memory modules, and a small range of input/output devices, to be supplemented at a later date by additional modules. All of these modules were interconnected by means of his defined standard computer bus. Wide publication of the standard computer bus allowed other companies, and customers themselves to attach other devices thus popularising the product range to mutual advantage.

Standard interfaces like Unibus, Multibus, Z-BUS and many others, have gained wide acceptance today as standard computer busses. Although each of these busses has its own individual features, nevertheless, over the years a certain similarity has emerged and it would be reasonable to assume that the best features have surfaced, and have been universally adopted.

AVIONIC COMPUTER BUS

High performance aircraft of today are being equipped with totally integrated avionic systems employing numerous microprocessor based subsystems. These subsystems are often developed independently by subcontractors whose design brief from the main contractor may not specify the choice of a specific standard microprocessor. An awareness of the multiplicity of microprocessors, each with its own architectural features and instruction repertoire, prompted the USAF to specify the MIL-STD-1750A ISA. (Instruction Set Architecture), in an attempt to standardise on a single preferred cpu at the subsystem level. This would encourage not only standard software modules, but also standard support packages and standard hardware.

Thus, ideally, an avionic system could evolve where each subsystem could contain as building blocks, standard hardware components such as :-

a standard cpu module,
a standard memory module (or modules),
and perhaps a standard MIL-STD-1553B data bus module, etc.

It becomes clear that such a system would benefit from a standard computer bus to interconnect these modules, together with other non-standard modules. Indeed, the definition of a standard computer bus is but a further logical step from defining a standard instruction set architecture, which itself implies a standard cpu. The MIL-STD-1750A User Group has discussed the possibility of specifying such a computer bus, but until now no specific implementation has been proposed. This paper proposes the "L-BUS" as the standard computer bus for MIL-STD-1750A applications. The L-BUS is an avionic computer bus which has been specified by I.A.I. Ltd. and is currently being implemented to fulfill a major contract for the Israel Ministry of Defence.

FUNCTIONAL REQUIREMENTS

The initial consensus of opinion favoured choosing an existing standard computer bus. Preferably one which had featured in other avionic subsystems designs. Various standard busses were studied and reviewed against the following considerations :

A) The prime requirement over and above any other consideration, was for a bus which would incorporate all of the functions specified in MIL-STD-1750A, whether mandatory or optional.

B) It was also felt at this stage that the Fairchild F9450 microprocessor chipset was emerging as an attractive means of implementing the 1750A instruction set. It was therefore considered worthwhile ensuring that the standard bus be compatible with this chipset, while still remaining as flexible as possible within the constraints of 1750A.

C) An attempt would be made to design the standard bus to be equally efficient for the smallest possible 1750A system configuration, as well as the largest, without the small system bearing the burden of being a subset of the large system.

MIL-STD-1750A specifies certain optional hardware mechanisms for memory management, and for memory access and violation protection. When associated with sophisticated applications requiring large amounts of memory, these mechanisms can efficiently provide partitioning and protection of the available memory space. However, if employed in simpler applications these mechanisms could become very cumbersome. The mechanisms in question are inherently related to memory addressing and therefore cannot avoid influencing the computer bus.

D) Much thought was given as to whether data lines and address lines should be multiplexed on the same signal pins, or whether they should be totally independent. The attraction of saving sixteen signal pins on the module connector was finally outweighed by the loss of system performance due to the delays introduced by multiplexors and demultiplexors. It was also felt that a multiplexed configuration would greatly complicate the debugging of hardware, and considerably reduce the flexibility to monitor and stimulate the bus with external test equipment. These considerations finally determined that the address and data lines should be maintained separate.

E) Additional (non-essential) lines should be specified, to aid in system test and integration, and to allow monitoring of the computer bus in real time. The use of these lines should be optional and should not impose any limitation on performance when not in use.

F) An attempt would be made to define the control signals of the L-BUS as asynchronous signals, in order to ease the integration of new subsystem modules.

SELECTION OF A STANDARD

The familiar standard busses were reviewed against these considerations, and it transpires that criteria (A) and (B) above impose severe constraints.

MIL-STD-1750A is very specific about the implementation of expanded memory addressing and the concept of converting logical addresses to physical addresses.

Each of the busses reviewed was originally defined for a specific processor. Therefore, compatability with the F9450 chipset (or any other 1750A chipset) is impossible without compromising on performance, because of the specific nature of the signal timing relationships.

Once it has been established that it is not possible to conform exactly to an existing standard, there seems little point in being slightly standard or even mostly standard, and thus constraining the design. Rather it is preferable to use existing standard busses only as a framework in which to specify the best possible design of a new bus standard.

The L-BUS therefore is not based on one particular existing computer bus. Rather it is a purpose designed avionic computer bus, conceived within the framework of several popular commercial busses.

L-BUS CONCEPT

The L-BUS consists of a set of common signal lines that interconnect all the modules connected to the 1750A cpu. The signals on these lines can be conveniently grouped into the following five categories. Each of which will be discussed in turn :-

A). L-BUS arbitration.
B) address lines.
C). data lines.
D). interrupt lines.
E). test lines.

The L-BUS is structured upon the "Master - Slave" concept. The device which initiates and controls the data transfer is called the "BUS-MASTER", and the passive device with which the bus-master transfers data is called the "BUS- SLAVE". The control signals employed to effect the transfer perform a type of "handshake" operation. This handshake between master and slave devices allows modules of different speeds to use the L-BUS interface.

A typical bus master would be the 1750A cpu module, another might be an I/O device controller module which performs its data transfers via direct memory access (DMA).

A typical bus slave would be a random access memory module. (ie. the sub- system memory).

L-BUS ARBITRATION

The L-BUS interface can accommodate several bus masters on the same system, each one taking control of the bus as it needs, to effect information transfers.

Any device which requires to perform a data transfer via the L-BUS must first request access to the L-BUS by means of the appropriate signal line "BUSREQ(0-5)/".

At any instant in time a number of bus requests may be simultaneously asserted. A bus arbiter located on one of the modules (normally the cpu module) will determine which requesting device will be granted the subsequent bus cycle. It does so by responding to the highest priority request with the corresponding bus-grant signal "BUSGNT(0-5)/". A unique "BUSGNT/" signal corresponds to each unique "BUSREQ/" signal.

Normally an L-BUS cycle is restricted to a single data word transfer, each cycle being granted by the bus arbiter in conformance with the device's defined priority. A device which is granted an L-BUS cycle must acquire the L-BUS as soon as it becomes unbusy, and then perform a single data word transfer. If the device requires to transfer more than one data word it should continue requesting the L-BUS until it has obtained sufficient cycles.

The device which acquires the L-BUS is said to become "Bus Master" for the duration of that bus cycle. The device with which the bus master transfers data during the bus cycle is called the "Bus Slave".

L-BUS EXCHANGE SIGNALS

"BREQ/" - BUS REQUEST - There are six independent L-BUS request lines, which may be used by potential bus masters for requesting bus access. The cpu will normally be connected to the lowest priority request line "BREQ5/".

"BGNT/" - BUS GRANT - For each of the 6 bus request lines there is a corresponding bus grant line. Only one of the devices requesting the bus will be granted access to it, at any particular time.

"BCLK" - BUS CLOCK - This signal is used for synchronising bus masters with the bus arbiter, and with the bus slave.

"BUSY/" - BUS BUSY - This signal is driven by the bus master currently in control of the L-BUS. All other potential bus masters monitor "BUSY/" to determine the state of the bus. The signal is bi-directional and should be synchronised with "BCLK".

Note that a change in the signal state of "BUSY/" indicates the end of one L-BUS cycle and the beginning of a new one.

MASTER - SLAVE ENGAGEMENT

The bus master when beginning its L-BUS cycle must specify characteristics of the transfer to be performed.

MIL-STD-1750A defines two types of data transfers each with their own independent address maps.

A) XIO data transfer.
B) memory data transfer.

XIO transfers are normally transacted between the cpu as bus master and an I/O device as bus slave.
Memory transfers are normally transacted between either the cpu or dma device as bus master and the memory as bus slave.

The bus master shall specify, when beginning its L-BUS cycle, whether the transfer is an XIO data transfer or a memory data transfer, and whether it is a read or write operation. It shall also specify for access protection purposes whether the bus master itself is the cpu, and if the bus cycle will be used for fetching an executable instruction.

XIO ADDRESSING SCHEME

During an XIO transfer 16 L-BUS lines (ADR0 - ADR15) define the XIO address, thus permitting a theoretical maximum of 65,536 possible XIO addresses, in accordance with MIL-STD-1750A.

MEMORY ADDRESSING SCHEME

Conceptually a distinction is made between logical memory addresses and physical memory addresses. At the beginning of each L-BUS cycle the bus master shall define a logical memory address on the 16 address lines "ADR0 - ADR15". The memory management unit (MMU), normally situated on the cpu module shall interpret the 4 most significant bits "ADR0 - ADR3" together with additional mapping information contained in the MMU registers, to form the absolute physical address in the memory. It does this by generating the 8 extended address lines "EADR0 - EADR7". Thus a total of 20 address lines (EADR0-7 together with ADR4-15) define the physical memory address to access a maximum of 1,048,576 locations.

When a particular system application does not employ MMU, the physical memory address will be identical to the logical memory address. That is, the absolute physical address will be defined by the 16 lines (ADR0-15). In this case the maximum size of memory is restricted to 65,536 locations.

L-BUS ADDRESS SIGNALS

"ADR0 - ADR15" - Sixteen address lines used to transmit the address of the memory location or I/O function to be accessed. When the expanded memory option is implemented, the four most significant bits (ADR0 - ADR3) will define a "logical" 4k page of memory. These four bits will be input to a memory management unit which will produce the absolute physical address in memory.

"EADR0 - EADR7" - eight extended address lines used only when expanded address in memory is implemented. These extended address lines are normally generated by a memory management unit to define the absolute physical address in memory of the 4k memory page. The eight extended address lines, together with the twelve least significant address lines - (ADR4 - ADR15) form the physical memory address.

"LSTRBA/" - LOGICAL ADDRESS STROBE - This signal is asserted by the current bus master to indicate that the logical address is present on the bus. The logical address lines should be sampled with the trailing edge of "LSTRBA/".

"PSTRBA/" - PHYSICAL ADDRESS STROBE - This signal is asserted by the memory management module. When true it indicates that the physical address is present on the bus. Potential bus slaves should sample the address lines with the trailing edge of "PSTRBA/".

"BIOM/" - MEMORY OR I/O - Indicates when active that the address on the bus is to be interpreted as a memory address. When inactive the address is interpreted as an XIO address.

"BRDW/" - READ OR WRITE - Indicates when active that information is to be transferred from the bus master to the bus slave (write). When this signal is inactive, the opposite direction (read) is defined.

"BDI/" - DATA TRANSFER OR INSTRUCTION FETCH - This signal may be activated only by a cpu and indicates when active that the cpu is fetching an instruction from the memory location indicated by the system address bus. This bus signal is intended primarily for the use of memory protection schemes and bus monitoring test equipment.

L-BUS DATA SIGNALS

"DAT0 - DAT15" - Sixteen bi-directional data lines used to transmit or receive information to or from a memory location or I/O device.

"BSTRBD/" - STROBE DATA BUS - This signal is asserted by the bus master. Its trailing edge is used to sample the sixteen data lines.

"BRDYD/" - DATA READY - This signal is activated by the bus slave and indicates to the bus master that the bus slave is ready to complete the data transfer. This signal may be used by a slow bus slave for causing additional wait states in the bus master.

INTERRUPT OPERATION

MIL-STD-1750A defines 16 system interrupts. Certain of these interrupts relate specifically to cpu functions and are therefore internal to the cpu. Other interrupt conditions however, will be generated externally to the cpu and need to be transferred to the cpu by means of the L-BUS interrupt lines.

The L-BUS provides two types of interrupt lines, "edge" triggered and "level" triggered. The edge triggered lines shall be activated with a pulse of minimum duration 100 nanosec. The level triggered interrupt lines shall be activated until acknowledged by the cpu with the L-BUS signal "INTACK/".

The following MIL-STD-1750A interrupts are initiated from the L-BUS :-

INTERRUPT	L-BUS SIGNAL	TRIGGER
0. power fail	PFINT*	edge
2. user interrupt	UINT2*	edge
8. user interrupt	UINT8*	edge
10. user interrupt	UINT10*	edge
11. user interrupt	UINT11*	edge
12. I/O interrupt	IOLINT1/	level
13. user interrupt	UINT13*	edge
14. I/O interrupt	IOLINT2/	level
15. user interrupt	UINT15*	edge

L-BUS INTERRUPT SIGNALS

"PFINT*" - POWER FAIL INTERRUPT - This signal shall interrupt the cpu when a power failure is detected. It instructs the cpu to perform those emergency measures necessary to ensure an orderly shutdown, so that at least a limited recovery will be possible when power is restored. The power fail interrupt has the highest priority of all interrupts.

"UINT2*,UINT8*,UINT10*,UINT11*,UINT13*,UINT15*" - Six interrupt lines corresponding to interrupt priority levels 2,8,10,11,13,15, as specified in MIL-STD-1750A Table VIII. These interrupt lines may be masked and/or enabled by program.

"IOLINT1/,IOLINT2/" - INPUT OUTPUT INTERRUPT LEVELS - Two interrupt lines corresponding to interrupt priority levels 12 and 14 as specified in MIL-STD-1750A Table VIII. These lines when asserted should remain active until the interrupt acknowledge cycle occurs. These interrupt lines may be masked and/or enabled by program and are used in conjunction with the input output interrupt code registers specified in MIL-STD-1750A Section 4.4.2.6.

"INTACK/" - INTERRUPT ACKNOWLEDGE - This pulse occurs during the execution of the interrupt acknowledge instruction. It may be used in conjunction with the appropriate data bit to reset the acknowledged interrupt.

SYSTEM TEST AND INTEGRATION

Much thought has been given to providing additional signals which will facilitate testing and debugging the system during all phases of development, integration, manufacturing and maintenance. The additional "non-essential" test signals are intended to complement the essential signals in order to provide the engineer with the ability to :-

A) passively monitor the L-BUS activity in real time, or

B) externally control and stimulate L-BUS activity with either minimum delay, slow motion, or single step.

Examples of features provided by the test signals are as follows :-

* The basic system clock "BCLK" may be replaced with an external clock frequency "EXTCLK", supplied by the test equipment.

* The signal "DISMEM/" can be used by the test equipment to selectively inhibit the system memory, so that the test equipment itself can simulate memory responses. This capability is a particularly powerful one since it allows the tester to simulate external events, provide simulated data, cause program branching, provide a monitor program service, assist in program loading, and many other powerful features.

* The signal "EXTWAIT/" may be used to freeze the state of the L-BUS on a particular L-BUS cycle. This could be required for one of the following reasons :-

  1. Synchronously delaying each L-BUS cycle by a predetermined number of "BCLK" cycles.

  2. Inserting a fixed or variable delay on detection of a particular external event.

  3. Asynchronously halting the subsystem by freezing all L-BUS activity. (similar to a "HALT" and "CONTINUE" function from an operator control panel).

L-BUS TEST SIGNALS

"EXCLKEN/" - EXTERNAL CLOCK ENABLE - This signal when active replaces the cpu clock with the signal appearing on "EXTCLK" pin.

"EXTCLK" - EXTERNAL CLOCK - This line is used to feed an external clock to the system. "EXCLKEN/" must be active to select the external clock feature.

"DISMEM/" - DISABLE MEMORY - This signal may be asserted to disable memory operation. While this signal is asserted every memory device shall maintain its L-BUS line drivers in the high impedance state.

"EXTWAIT/" - EXTERNAL WAIT - This signal may be asserted in order to lengthen the data phase of a bus cycle. The bus master shall condition its receipt of "BRDYD/" with "EXTWAIT/" inactive.

"STOPCNT/" - STOP COUNTING - This signal when active inhibits the counting of the timers specified in MIL-STD-1750A (Timer-A and Timer-B). This signal would also be normally used in conjunction with the "EXTWAIT/" signal.

"INHWDOG/" - INHIBIT WATCHDOG - This signal when active inhibits the counting of the Trigger-go function specified in MIL-STD-1750A. This signal would also be normally used in conjunction with the "EXTWAIT/" signal.

"DMACYC/" - NON-CPU BUS CYCLE - Asserted by the bus arbiter to indicate that the L-BUS is currently granted to a non-cpu device. This signal is primarily intended to enable bus monitoring test equipment to distinguish between cpu memory accesses and dma activity.

"BSNEW" - START NEW - Indicates when active that a new instruction will start execution in the next cycle.

"CONREQ/" - CONSOLE REQUEST - This signal is intended for initiating the console function defined for the Fairchild F9450 microprocessor, or any other similar console function.

OTHER L-BUS SIGNALS

A number of additional signals are briefly described below, which do not satisfactorily fit into any of the above categories. It is not within the scope of this paper to discuss these signals in detail since an in depth familiarity with the requirements of MIL-STD-1750A, and/or the Fairchild F9450 chipset may be necessary for a complete understanding.

"SCR0 - SCR3" - SYSTEM CONFIGURATION REGISTER - Indicates the following system configuration parameters,

SCR0 -	MMU present
SCR1 -	BPU present
SCR2 -	console not present
SCR3 -	co-processor present

"BNMLPWRUP/" - NORMAL POWER-UP - This signal comes true following the application of power to the subsystem. It indicates that the cpu has successfully completed its power-up sequence and self test routine.

"RESET/" - INITIALIZATION SIGNAL - Resets the entire system to a known internal state. The cpu shall be initialized to the reset state as defined in MIL-STD-1750A Section 4.4.4.1.

"SUROM/" - START-UP-ROM ENABLED - Inhibits main memory devices from responding to the memory address on the system address bus for read operations, when the start-up-rom option is active.

"BGLOBPROT/" - GLOBAL PROTECT - Indicates that the entire main memory is write protected. This signal is intended primarily for write protection of main memory during cold-start, prior to the loading of the registers of the MMU and BPU with their initial parameters.

"BMPARERR/" - DATA PARITY FAULT - Indicates that information appearing on the accompanying data lines has at least one bit in error.

"CORRERR*" - CORRECTED ERROR - This signal is asserted by an error detection and correction scheme during a bus cycle in which a data error has been corrected.

"MEMVIOL/" - MEMORY VIOLATION - Indicates that the current bus cycle is addressing a memory location in violation of the memory protection scheme. This signal will be produced by the memory management unit or block protection unit.

"SYSFAIL/" - SYSTEM FAIL - This signal shall be made true on detection of any irrecoverable error which will result in failure of the system.

"SYSFLT1*" - SYSTEM FAULT - This signal may be asserted by appropriate devices to set ERROR BIT-15 in the processor FAULT REGISTER.

SUMMARY

This paper has indicated the benefits which can be derived from adopting a standard computer interface bus for avionic subsystems. The criteria were considered for choosing a MIL-STD-1750A compatible bus, and the process which led to the specification of the L-BUS was reviewed. The L-BUS itself was described briefly, but a full specification is available from :-

DEPT. 4711, ENGINEERING DIVISION,
ISRAEL AIRCRAFT INDUSTRIES LTD.
BEN-GURION INTL. AIRPORT,
ISRAEL.

This paper proposes that the L-BUS be considered as a standard computer bus for MIL-STD-1750A.

A WAVELENGTH DIVISION MULTIPLEXED (WDM) OPTICAL DATA BUS FOR FUTURE MILITARY APPLICATIONS

84-2681

K. K. Chow and W. B. Leonard
Advanced Electronics Laboratory
Lockheed Palo Alto Research Laboratory
3251 Hanover Street
Palo Alto, Ca. 94304
415/424-2167

T. C. Yang
Dept. of Information Engineering
Feng Chia University
Taichung, Taiwan, Republic of China

Abstract

The data bus for future military applications of the 1990's must have high performance, flexibility, expandability, EMI and EMP immunity, as well as volume and weight advantage. Fiber optics fills these needs admirably well, particularly when wavelength division multiplexing (WDM) is considered. This paper proposes a new bus topology, protocol and architecture, based on WDM, to achieve all these goals. Specifically, a segmented ring is proposed, in which a number of segments, each one employing specific wavelengths and under the control of a supervisor, are hooked end-to-end. Within a segment, reservation time division multiplexing is employed. This allows maximum bus utilization as well as dynamic reassignment. Repeaters are used only at segment supervisors; ordinary nodes are passively coupled to the ring. Communications within a segment and among segments can be carried out without interference from other segments. A separate wavelength is reserved for bus administration, priority request, interrupt control, etc. With present-day technology, at least twelve wavelengths can be used. Future improvement in technology will bring about added capabilities of such a bus.

1. Introduction

In considering a generic data bus concept for future (1990's) military applications, flexibility and expandability come immediately to mind. This is because the future systems under consideration can include a wide variety of different terminal node types and different data types. Therefore, we must first provide for the bussing of more numerous and complex signals, i.e., the future bus must provide greater throughput, EMI immunity, and ease of being reconfigured. Also, the minimization of bus system volume and weight as well as the elimination of unintentional rf radiation are desirable. All these factors point to the use of fiber optics as the transmission medium, a fact recognized since the heydays of fiber bundles and first demonstrated in the ALOFT program by the U.S. Navy. Later work included the use of fiber optics (FO) in data busses such as MIL STD 1553B, as well as various point-to-point data links and daisy-chains.

While these endeavors show that FO can be directly substituted into an existing architecture so that high data rate transmission on a given link can be obtained quite readily, nevertheless, the advantages of FO have not been fully utilized. This is due to the fact that, presently, work on FO data bus has been a "direct translation" from the low data rate, low throughput, copper wire systems, and thus carries with it the protocol inadequacies of low throughput systems. (We hasten to point out, at this juncture, that high data rate is not the same as high throughput; high throughput is the product of high data rate and long transmission time.) That is, when the protocol is such that most of the time is expended in requesting and granting the use of the bus, and the subsequent confirmation or request for retransmission, the available time for transmission will be correspondingly shortened. This problem was brought out painfully clearly in one of the studies on EW applications for the USAF.[1] This problem becomes even more serious when a large number of terminals (nodes) are tied to the bus. Therefore, to achieve a truly high throughput data bus, protocols based on all the attributes of FO must be developed.

A very interesting attribute of FO is that even with present-day single fiber technology, several transmission windows exist. That is, for a given physical bus installation, a number of optical wavelengths can be supported. This opens up a new dimension in bus architecture, for in addition to the traditional time division multiplexing (TDM), frequency division multiplexing (FDM), we can now use wavelength division multiplexing (WDM). For each optical wavelength λ, we can utilize the full electronic bandwidth to transmit data either in TDM or FDM. We also propose to reserve one wavelength for the administration of bus transactions so that data transfer efficiency will not be hampered by interruptions due to bus transactions. Included in this administrative bus will be priority assignment, interrupt, dynamic reconfiguration, etc. This is shown schematically in Fig. 1.

Referring to Fig. 1, it can clearly be seen that with this added dimension, the flexibility of the bus architecture is greatly increased. Not only a large number of terminals can be served, but also different data types and data rates may be simultaneously transmitted. As more terminals are added, different wavelengths may be brought in to serve them. As parts of the bus are degraded, different transmission and switching schemes may be brought to bear to ensure a graceful degradation. All these features can be implemented when a proper protocol is developed. Indeed, with the use of a good protocol, future modification of bus configuration and addition of nodes can be

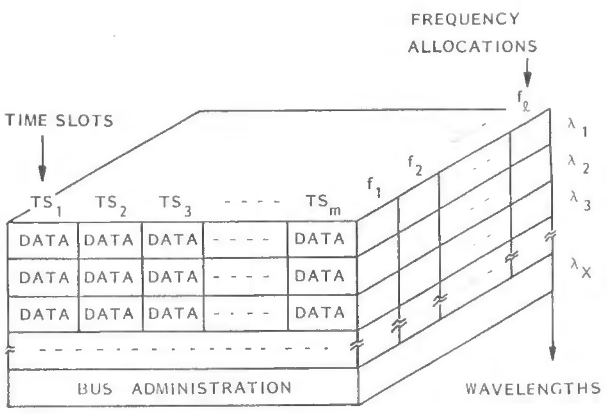

FREQUENCY ALLOCATIONS
TIME SLOTS
TS_1 TS_2 TS_3 - - - - TS_m
DATA DATA DATA - - - - DATA
DATA DATA DATA - - - - DATA
DATA DATA DATA - - - - DATA
BUS ADMINISTRATION
f_1 f_2 f_ℓ
λ_1 λ_2 λ_3 λ_X
WAVELENGTHS

Figure 1

Use of Combined WDM, TDM and FDM Schemes to Improve Throughput in a Data Bus

implemented easily and without detriment to bus performance.

In the rest of this paper, we shall first give a brief discussion of existing bus topology and protocol, and then suggest a new topology and protocol that are specifically based on WDM to achieve high throughput and flexibility.

2. System Topology

Presently, three major types of bus topologies are in common use. These are the straight bus, the ring, and the star, as shown in Fig. 2(a), (b) and (c) respectively. In all these configurations, the nodes may contain couplers that route messages in and out of these nodes. To select the topology that will take the most advantage of the properties of the FO link, the three possible candidates - conventional bus, conventional ring, and star - are evaluated according to the following criteria: (a) throughput, (b) reliability, (c) versatility-modifiability, and (d) amount of optical components. Practical issues such as the minimization of number and complexity of switches, the addition and deletion of nodes without modifying the rest of the system, the impact of single point failures, etc., are considered within this context. Results of this evaluation are summarized in Table 1.

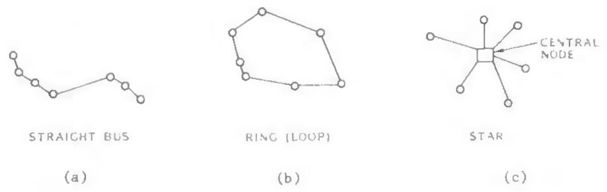


Figure 2

Conventional Data Bus Topologies: (a) Straight Bus; (b) Ring (Loop); and (c) Star

Although there are both advantages and disadvantages to each topology, the ring is generally favored over the straight bus and the star. One

Table 1

COMPARISON AMONG THREE TOPOLOGIES - CONVENTIONAL BUS, RING, AND STAR

Item	Bus	Ring		Star
		Passive	Active	
1. Signal strength limits number of terminals	Yes	Yes	No	Yes
2. Large dead gap waste for long distances	Yes	No	No	Yes
3. Cable length requirements for n stations, R radius	$\sim 2\pi R$*	$2\pi R$	$2\pi R$	$2\pi R$
4. Potentially large number of optical connectors per terminal	No	No	No	Yes
5. Node failure halts all transactions	No	No	Yes	No
6. Delay at each node	No	No	Yes	No
7. Double couplers required at each node	Yes	No	No	No
8. Lack of provision for signal termination	No	Yes	No	No

* Of the existing common bus structures, some require more cable per installation than a loop for the same site.[2]

of the more important factors contributing to this is that the wasted dead gaps that are inherent to the bus and the star increasingly degrade throughput for higher data rates over longer distances. A ring with an active repeater at each node has a further advantage that signal strength does not become a factor in limiting the total number of nodes since each needs only to send data to the next, regardless of the total number of nodes. Two disadvantages of this active ring, however, are that each node adds propagation delay to the system, and a failed node would block all transactions (unless redundancy is employed). In practice, on the other hand, both the star and the bus tend to require more optical components per terminal than the ring topology. This is true for the star because the extra fiber cabling required to directly connect all nodes to a central point often means that multiple optical connectors will be used in the fiber for each node. For the bus, the fact that light must be sent out both directions from a node makes it necessary to have two couplers at a node since each coupler can only inject light onto the bus in one direction. Of course, the bus requires less total fiber length than the star since nearest neighbor distances are usually much less than distances to one central point.

For the ring, then, the advantages generally outweigh the disadvantages when all are considered. However, neither the purely passive ring nor the ring with an active repeater at each node becomes a very strong winner when compared to the bus and the star. The main disadvantages of the passive ring are that limited signal strength places too small an upper limit on the number of nodes for some systems, and there is generally a lack of provision for termination of messages that have circulated around the ring. The ring with an active repeater at each node has the disadvantages of additional delay associated with each node and that a failed node potentially halts all transactions of the system. What is desired, then, is an architecture which effectively avoids all these disadvantages.

For this reason, we propose a segmented ring approach as shown in Fig. 3, to combine the best features of the passive and active rings. In this architecture, active repeaters divide the ring into segments, and within each segment ordinary nodes couple passively to the ring. With just a few active repeaters, not only is the capacity for a very large number of nodes achieved, but message termination becomes convenient. Since the number of repeaters is small, not only is the accumulated delay kept small, but the cost of fault tolerant redundancy is far less than that for the conventional active ring. The segment architecture also provides a basis for taking advantage of WDM, as will be described later.

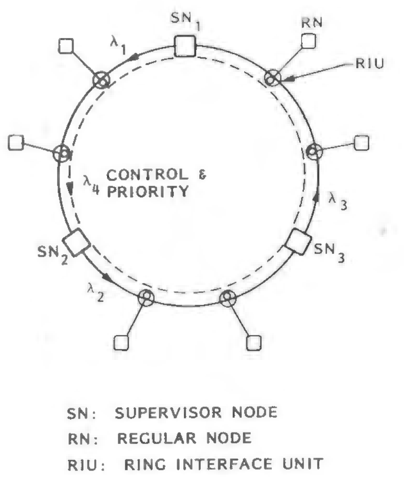


Figure 3

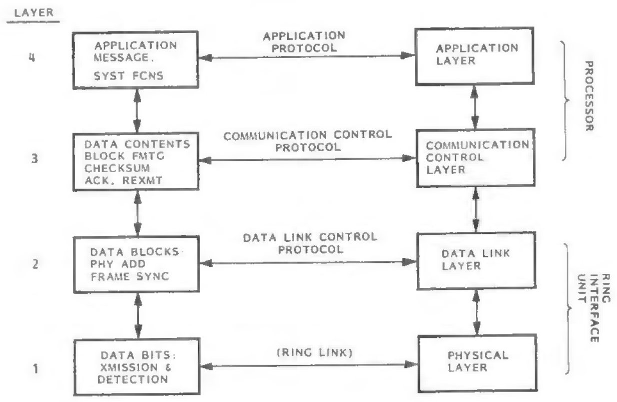
Basic Concept of a Segmented Ring

Each segment employs the intrasegment wavelength λ_0 and is under supervision of a supervisor node, which uses its characteristic wavelength λ_n for intersegment communications. Control and priority signals are broadcast to all nodes using a special wavelength, λ_x.

3. Protocols

To reduce their interdependency and to enhance flexibility, most data communication network systems are organized as a series of layers or levels, each one built upon its predecessor. The purpose of each layer is to offer certain services to the higher layers, and to shield these layers from the details of how the offered services are actually implemented. A well known example is the seven layer ISO protocol for open connections.[3] In this manner, network reconfiguration can be more readily achieved at the local level with a minimum impact to the overall system. Our architecture is conveniently described in four layers, as shown in Fig. 4. In this structure, the physical layer (the lowest layer) is concerned with the transmission of raw data bits over the FO link. The design issues at this level have to do with making sure that when one terminal sends a "1" bit, it is received by the intended receiver as a "1" bit and not as a "0" bit. Therefore, design issues here largely deal with mechanical, electrical, and procedural interface to the link. Going up to the next higher level, the task of the data link layer is to synchronize movement of data blocks over the bus, and above that the communication control layer formats data into blocks (or frames), transmits the frames sequentially, and processes the acknowledgements.

Since the physical layer merely accepts and transmits a stream of bits without any regard to meaning or structure, it is up to the data link layer to create and recognize frame boundaries. This is generally accomplished by attaching special bit patterns to the beginning and end of a frame. These bit patterns can also accidentally occur in the data, therefore special care must be taken to avoid confusion. It is also at this layer that the physical access to the link is controlled. To ensure the most efficient use of the data bus, the use of time slots is proposed. That is, the available time for each segment is divided into time slots. The assignment of time slots is made in accordance with the load level of the nodes: heavy-load nodes are allocated dedicated slots and light-load nodes are assigned sharable slots and compete for the slots using a contention scheme.



LAYER
4
3
2
1
APPLICATION MESSAGE. SYST FCNS
DATA CONTENTS BLOCK FMTG CHECKSUM ACK. REXMT
DATA BLOCKS PHY ADD FRAME SYNC
DATA BITS: XMISSION & DETECTION
APPLICATION PROTOCOL
COMMUNICATION CONTROL PROTOCOL
DATA LINK CONTROL PROTOCOL
(RING LINK)
APPLICATION LAYER
COMMUNICATION CONTROL LAYER
DATA LINK LAYER
PHYSICAL LAYER
PROCESSOR
RING INTERFACE UNIT

Figure 4

The Four-Layer Protocol Structure

The lower two layers reside in the ring interface unit, the upper two layers reside in the data processor.

Or, the light-load nodes can be assigned sharable slots but different frequencies so that they can get onto the bus within these time slots without using contention. By keeping track of slot usage and priority interrupt, time slots can be reassigned as the load changes. This overall scheme is shown in Fig. 5 and is known as the reservation time division multiplexing (RTDM) scheme. Within the dedicated time slots, FDM can be used to multiplex signals from within a given terminal, if such a scheme can be used to advantage.

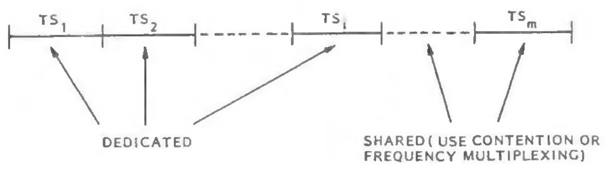


Figure 5

Reservation TDM

It is at the level of the communication control layer that transmission errors are checked and acknowledgement (ACK) sent. If, however, the data content does not check out, a request from retransmission (REXMIT) will be sent and the sender will retransmit at the next opportunity. On the other hand, if the frame has not been received at all (e.g., destroyed by a noise burst in the system), the sender will neither receive an ACK or a REXMIT, in which case the sender will automatically retransmit after a suitable time interval. However, multiple transmissions of the same frame introduce the possibility of duplicate frames. A duplicate frame could be sent, for example, if ACK from the receiver back to the sender was destroyed. It is up to the communication control layer to solve the problems caused by damaged, lost, and duplicated frames, so that layer four (the application layer) can assume that it is working with an error-free signal.

In this protocol, therefore, the data link layer controls the operation of the ring network. The major design issues here are channel allocation and flow control. The application layer is the user's interface with the network - it is with this layer that the user must negotiate to establish a connection with a node. This layer will not be discussed further in this paper because its content is application dependent. For example, when two users on two different nodes communicate, they alone determine the set of allowed messages and the action taken upon receipt of each. Therefore, in this protocol only the first three layers - physical, data link, and communication control, are defined in detail. However, it is not possible to present all these details in this paper; rather, a summary is presented in the following by the use of an example.

Referring to Fig. 6, one of the application nodes (e.g., Node #1) wants to send a message to another node (e.g., Node #4). To accomplish this, the events that must take place in time sequence are:

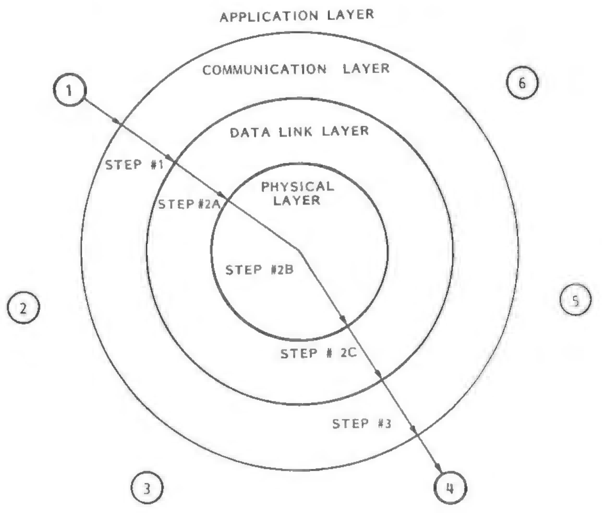


Figure 6

Layer View of Link System

(1) Format data into frame including error check bits.

(2) Move frame. This is accomplished by:

(2a) Add sync pattern and, if necessary, stuff bits. Stuffing depends on the multiplexing scheme; it is not needed in some schemes, e.g., fixed time slot.

(2b) Enter data into the link by TDM, FDM, or WDM, and move data to the intended destination(s).

(2c) Detect data bits, strip off sync pattern and de-stuff bits _if_ necessary.

(3) Check for errors, send back acknowledgements, strip off check bits and unpack frame to give original message before formatting.

4. The Segmented Ring Architecture

As discussed in Section 2, a segmented ring using WDM appears to best utilize FO characteristics and provide performance, expandability, and reliability. This segmented ring consists of a number of segments; each node connected to a specific segment feeds a segment supervisor node which in turn originates and absorbs only one main wavelength as shown in Fig. 7. That is, a terminal node in a segment (e.g., Segment 1) transmits data on the feeder wavelength λ_0 to the supervisor node at the end of the segment (e.g., Supervisor 1). The supervisor node absorbs all the transmissions on λ_0 within its segments and repeats the messages for sending onto the entire ring by using its unique wavelength reserved for main data transmission. This unique wavelength is known as the characteristic wavelength for the segment (i.e., λ_1 for Segment 1, λ_2 for Segment 2, etc.) and is unique in the sense that the supervisor originates all the main data transmissions at that wavelength and absorbs that wavelength after one complete cycle around the ring. In addition to retransmitting on this characteristic wavelength, a supervisor node also generates timing signals for the reservation time division multiplexing on λ_0 for the segment following; that is, Supervisor 1 generates timing signals for Segment 2, Supervisor 2 generates for Segment 3, etc. This is where dynamic reassignment of time slots occurs.

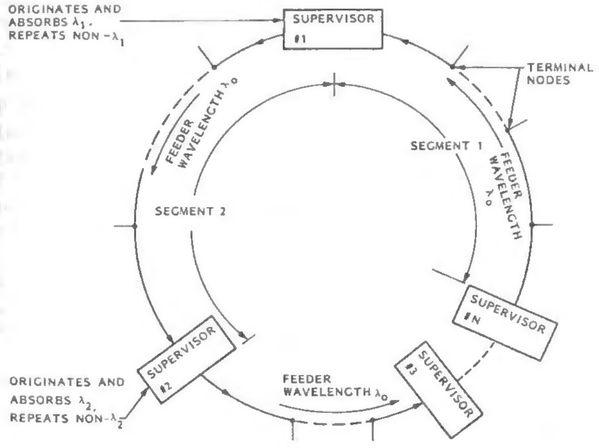

ORIGINATES AND ABSORBS λ_1. REPEATS NON-λ_1
SUPERVISOR #1
TERMINAL NODES
FEEDER WAVELENGTH λ_0
SEGMENT 1
FEEDER WAVELENGTH λ_0
SEGMENT 2
SUPERVISOR #N
SUPERVISOR #2
SUPERVISOR #3
ORIGINATES AND ABSORBS λ_2. REPEATS NON-λ_2
FEEDER WAVELENGTH λ_0

Figure 7

Main Data Flow of an N-Segment Ring

Finally, the supervisor node also serves as a repeater (i.e., detects and retransmits) all other characteristic wavelengths associated with other supervisors, as shown in Fig. 7. In order to permit any terminal node to transmit to any other terminal node, each terminal must be capable of transmitting on the feeder wavelength (λ_0) and receiving each of the main wavelengths ($\lambda_1 \ldots \lambda_n$). The availability of n main wavelengths means that up to n data transmissions may be performed simultaneously as long as there is only one active data source in any one segment. The receiver of a given node can detect all the wavelengths so that one message from each segment can be received simultaneously. If several destinations within a given segment are to be reached, these must be specified in a multi-destination address field. This field will be at the beginning of a transmitted block so that only the desired destination will have the need to read the entire block from the bus. Addition and elimination of nodes can be simply accomplished by a change of addresses. To maintain flexibility, the bus interface is envisioned to consist of a standard link interface which interfaces with the fiber optic data link and a programmable node interface which gives the desired interconnection flexibility.

It should be noted that each main data wavelength makes exactly one complete trip around the ring before it is absorbed. Therefore, there is no possibility of a message colliding with itself even if the message block length is much larger than the physical ring size. Each message will, therefore, pass over each potential destination only once - after which the bus will be available to the next sender. The feeder wavelength, however, only resides within the segment, i.e., it is detected and converted to a main wavelength at the supervisor node. Thus the same wavelength may be used as the feeder on each segment.

Thus far, we have only discussed the main data bus - namely, the bus for high throughput data transmission. In actuality, there has to be a mechanism to accommodate bus transactions such as request, acknowledge, interrupt, assignment of priorities, etc. For this application, we envisage a logically separate, low data rate bus, so that the transmission of high rate data does not have to be interrupted for these transactions. This "administrative bus" is not an independent ring, but rather, is one utilizing a separate wavelength (e.g., wavelength λ_X in Fig. 1 is reserved for this purpose) to transmit the required messages among all the terminals and supervisor nodes.

Using this architecture, the number of wavelengths to be used can be expanded as system demand increases. Each added wavelength is a new logical bus to be used either as an added segment in the ring, or as an all-encompassing ring similar to the administrative bus. Even with today's technology, each optical wavelength propagating within a fiber can handle a wide modulation bandwidth ($>$ 1 GHz) and support a number of nodes ($\sim$ 32). A good single fiber can have three transmission windows (0.8, 1.3 and 1.5 μm), each of which can accommodate at least four wavelengths in terms of WDM. Therefore, there are currently at least twelve possible logical busses for a given fiber installation. Future technology will bring about additional capabilities.

5. Conclusions

Our motivation for this study is to provide a high throughput optical fiber ring so that elements of a distributed system can be linked together efficiently. The planning and preliminary design considerations of such an FO system have been presented. The most important feature of the proposed segmented ring system is the use of combined WDM, TDM, and FDM schemes. A separate bus-transaction and priority/interrupt control bus is an integral part of this system to ensure efficient ring usage.

Specifically, the major characteristics of the proposed segmented ring system are:

(1) A combined WDM, TDM, and FDM scheme is applied to optimize the ring utilization to give much increased throughput.

(2) The control is decentralized to enhance fault tolerance - supervisor nodes are provided to manage a segment of the bus.

(3) Priority/interrupt control is provided on a "logically" separate channel to achieve fast response.

(4) A four layer communication protocol structure is used to give modularity and flexibility.

(5) A dual mode time slot assignment scheme (reserved TDM) is used to improve system throughput.

(6) Multiple destination is allowed to increase bus efficiency.

(7) Dual interface approach is used in the design of the ring interface unit to facilitate reconfiguration.

Some of the details relating to bus operation and fault tolerance have also been worked out, but have not been reported here as they do not pertain to the principles of this segmented ring.

In conclusion, it is our firm belief that in considering a generic family of data busses for future military applications, a WDM optical bus as described here presents a number of very attractive features.

References

[1] "EW Applications of Fiber Optics", Final Report, Contract AFAL-TR-79-1165, Air Force Avionics Laboratory, WPAFB, Ohio, 1979.

[2] D. Y. Farber, "A Ring Network", Datamation 21(2), pp 44-46, February 1975.

[3] H. Zimmerman, "OSI Reference Model - The ISO Model of Architecture for Open Systems Interconnection", IEEE Trans. on Comm., Vol. Com-28, No. 4, April 1980.

DATA BUS INTERFACE MIL-STD-1553B / ARINC 561

James E. Spieth*

Electronics Engineer
Systems Engineering Avionics Facility
Aeronautical Systems Division, Air Force Systems Command
United States Air Force
Wright-Patterson Air Force Base, Ohio

Abstract

This paper describes interface circuitry that converts the data structure and transmission frequency of either MIL-STD-1553B or Aeronautical Radio Inc. (ARINC) Characteristic 561 to that of the other. This circuitry was developed in the Systems Engineering Avionics Facility (SEAFAC) in support of an Air Force program to update the avionic (aviation electronic) systems on board the KC-135 tanker aircraft. Specifically, it is incorporated in an engineering Hot Bench which is being used to evaluate the new avionics. By modeling major aircraft subsystems in a computer and utilizing hardware interface circuits, the Hot Bench provides realistic electrical stimuli to the avionics. The subsystem models communicate with the interface circuits over a MIL-STD-1553B Data Bus. The interfaces transform model data to whatever electrical format the aircraft subsystems normally produce; and data from that format back into MIL-STD-1553B format. In the case of the aircraft's Inertial Navigation System, the electrical format needed is the ARINC 561 Data Bus. The MIL-STD-1553B / ARINC 561 Interface has three major parts: a MIL-STD-1553B remote terminal, an ARINC 561 transmitter, and an ARINC 561 receiver. Both the transmitter and receiver are designed with common digital and analog electronic components. The Interface has been constructed, tested, and has sucessfully undergone integration testing with its subsystem model.

The FSA/CAS Program

The KC-135 is the Air Force's primary aerial refueling tanker aircraft. The first version flew in 1956 and its airframe is basically similar to that of the Boeing 707. Various improvement programs are in progress to update the aircraft, thereby enabling it to remain operational well past the year 2000. The Fuel Savings Advisory / Cockpit Avionics System (FSA/CAS) Program updates the present KC-135 cockpit with new avionics integrated via a MIL-STD-1553B Data Bus. The new avionics provide fuel saving advisory information based on mission to the pilots in the form of target engine pressure ratios, airspeeds, and altitudes. This allows fuel efficient operations to be conducted. Delco Systems is the system contractor to the Air Force for the FSA/CAS Program.

What is a Hot Bench

A Hot Bench is specialized hardware and software simulating an aircraft's electrical environment so avionics may be operated on the ground. The software is used to model the aircraft and its subsystems in a computer and the hardware is used to provide the electrical signals the avionics expect to see. The degree to which the software accurately models the aircraft and its subsystems can vary for trade-offs in cost and development time versus the type of testing to be accomplished. The Hot Bench allows individual equipment or system testing without requiring expensive flight time and an actual air vehicle. Also, areas involving safety which could not be accomplished in flight can be investigated with the Hot Bench.

The Data Buses

MIL-STD-1553B

MIL-STD-1553B is a Defense Department military standard that sets the requirements for aircraft internal time division command/response multiplex data buses. It is a serial digital data bus with a bit rate of 1MHz. There can be a total of 32 units attached to the bus. Aircraft subsystems are interfaced to the bus by remote terminals. Remote terminals can be separate units or be imbedded within the subsystem. The remote terminals receive and transmit data only when commanded to do so by the bus controller which initiates all information transfers on the bus. All information transfers are acknowledged by the participating remote terminals by means of a status word.

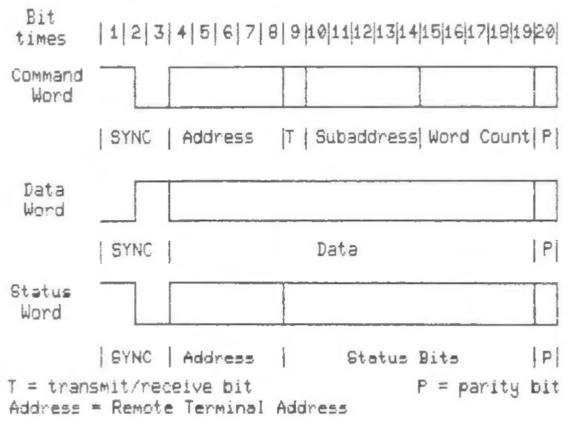
Messages. There are three types of messages which are used. The first is a normal information transfer message, the second is a mode command message which allows the bus controller to communicate with the bus related hardware in the remote terminal, and the third is a broadcast message. With the first type, data can be transferred from the controller to a remote terminal, from a terminal to the controller, or between two terminals. With the second type, one data word can be transferred from the controller to a remote terminal, from a terminal to the controller, or an acknowledgment message with no data word between the controller and terminal. With the third type, data is sent by the bus controller to one or more remote terminals or from a remote terminal to one or more other remote terminals, but there is no acknowledgment. Broadcast messages are not allowed in Air Force systems.

Words. Messages are made up of three types of words: command, status and data. All types contain a sync waveform which is three bit times in length, 16 bits of data, and a odd parity bit for a total of 20 bits. There are two different types of sync waveforms. The first one identifies the word following to be either a command or status word. The second identifies the word following to be a data word. The formats of the words are shown in

*Member IEEE

Figure 1. A normal message which transfers data from the controller to a terminal would consist of a command word, followed by data words, and then the status word from the terminal acknowledging the transfer. A maximum of 32 data words can be transferred in one message. The remote terminal must respond with its status word to the bus controller within 4 to 12 microseconds. There must be a minimum 4 microsecond gap between the completion of one message and the start of another.

Figure 1 MIL-STD-1553B Word Formats

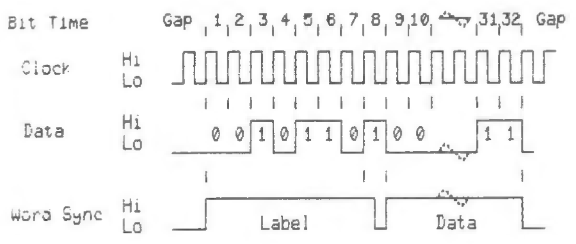


Cable. The bus is physically made up of a cable with two twisted, shielded conductors. It has an impedance of 70 to 85 ohms and is terminated with a resistance having a value equal to the characteristic impedance of the cable. Remote terminals are coupled to the bus by a short length of cable called a stub. For Air Force systems, the stub is coupled to the bus by a transformer.
Data Code. The data is transmitted over the bus in Manchester II bi-phase code. A logic one is transmitted as a postive then negative pulse and a logic zero is transmitted as a negative then positive pulse. The peak-to-peak signal on the stub varies between 1 and 14 volts line-to-line. The most significant bit is transmitted first. This standard was first published in 1973 and the B revision was published in 1978.

ARINC Characteristic Number 561

The ARINC 561 characteristic defines an air transport Inertial Navigation System (INS). Within the characteristic, a six wire serial digital data bus for navigation units to communicate with each other in a dual or triple INS installion is detailed. This is a broadcast bus with one sender and one listener. It is a one-way bus with information flowing from the sender to the listener and there is no acknowledgment by the listener. Communications between the navigation unit and its control/display unit were not defined in the characteristic but an uncommitted group of wires were provided for this purpose. The 561 Bus is physically made up of three twisted-shielded pairs of wire. One wire in each pair, called the reference line, is connected to ground. The other wire in the first pair carries a continuous 9.6 kHz clock signal. The other wire in the second pair carries a synchronization signal which denotes the start and end of a word, and the other wire of the last pair carries the serial data signal. These signals are shown in Figure 2.

Figure 2 ARINC 561 Six Wire Timing Diagram

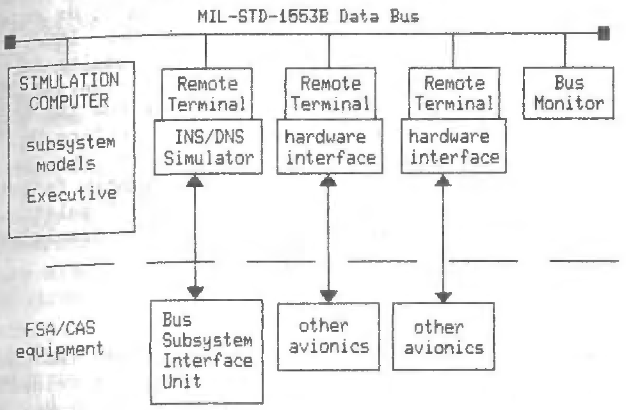


Words. Words are 32 bits long. The first 8 bits are a predefined label and the other 24 bits are data. The information the data word contains and its label are defined in the ARINC 561 characteristic. There must be a minimum gap of 1 bit time or 104 microseconds between words.
Data Code. The sync and data signals are transmitted on the bus in a nonreturn-to-zero bi-level code. A logic 1 is transmitted as 12.5 volts line-to-line and a logic 0 as 0 volts line-to-line. The least significant bit is transmitted first.
Discretes. The 561 characteristic also defines two 28 VDC discrete signals between the navigation unit and its control/display unit. The discretes go from the navigation unit to the control/display unit and inform the operator if the navigation unit is operating on battery power (Battery) or if a fault condition (Warning) exists in the navigation unit. This characteristic was first published in 1967.

The FSA/CAS Hot Bench

The purpose of the FSA/CAS Hot Bench is to provide an integrated test bed for Air Force engineering evaluation and validation of the FSA/CAS's performance. In the Hot Bench, three major aircraft subsystems are modeled by software in the simulation computer. They are: the aircraft Inertial Navigation System/Doppler Navigation System (INS/DNS), the fuel system, and various aircraft sensors and controls. The simulation computer is a Digital Equipment Corp. (DEC) VAX 11/782 running under a VMS operating system. The subsystem models are written in the FORTRAN programming language. A block diagram of the Hot Bench is shown in Figure 3. The INS modeled is a Delco Systems Carousel IVE unit which was designed to the ARINC 561-6 1970 revision of the characteristic. Delco Systems uses the ARINC 561 defined inter-navigation unit bus as the means for the navigation unit to communicate with its control/display unit. Additionally, a lamp test discrete signal was defined by Delco from the control/display unit to the navigation unit to indicate the operator wants a lamp test to be performed. Aircraft sensors and controls modeled include engine pressure ratio values and flap position, engine anti-ice and air conditioning switches, and differental pressure and total air temperature values. In the Hot Bench, each of these areas have a corresponding hardware interface circuit. The models exchange data with the hardware interface circuits over a MIL-STD-1553B Data Bus. The hardware interface circuits are used to transform this data into whatever electrical format (analog, digital, syncro and discrete) the modeled aircraft subsystems normally produce, and from that format back into MIL-STD-1553B format.

Figure 3 Hot Bench Block Diagram



The hardware interface this paper describes is the interface between the INS/DNS model in the simulation computer and the Bus Subsystem Interface Unit, one of the avionic units in the Fuel Savings Advisory/Cockpit Avionics System. This interface is called the Inertial Navigation System/Doppler Navigation System (INS/DNS) Simulator. A block diagram is shown in Figure 4.

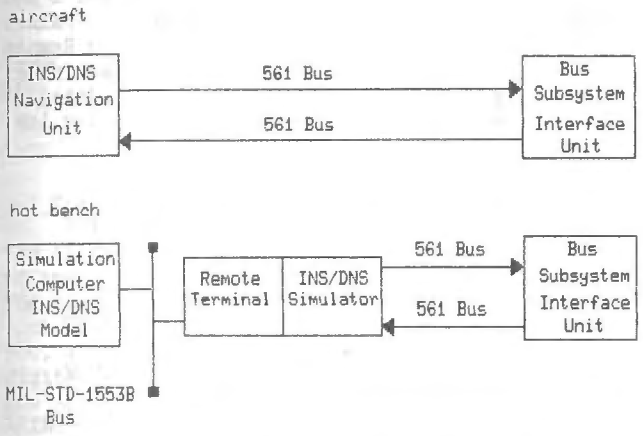


Figure 4 INS/DNS Simulator Block Diagram

The INS/DNS Simulator

Referring to Figure 4, the INS/DNS software model in the simulation computer sends data to the INS/DNS Simulator (hardware interface) over the MIL-STD-1553B Bus. The INS/DNS Simulator reformats the data and transmits it on the ARINC 561 Bus to the Bus Subsystem Interface Unit. The Bus Subsystem Interface Unit transmits its data over the other ARINC 561 Bus and it is received by the INS/DNS Simulator. The INS/DNS Simulator reformats the data and it is sent over the MIL-STD-1553B bus to the INS/DNS model.

Implementation. A software executive program in the simulation computer acts as the bus controller for the MIL-STD-1553B Bus. The program uses the normal information transfer type message to transfer data from the INS/DNS Model (bus controller) to the INS/DNS Simulator (remote terminal). The message transfers 31 data words from the Model to the INS/DNS Simulator and is shown in Figure 5. The information contained in

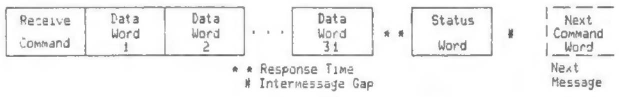


Figure 5 INS/DNS Simulator MIL-STD-1553B Receive Message

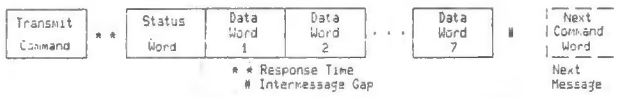


Figure 6 INS/DNS Simulator MIL-STD-1553B Transmit Message

the data words are navigation data calculated by the Model; such as present position in latitude and longitude, heading, altitude, ground track, ground speed, time to go, cross track, track angle error and distance to go. This data represents the Carousel IVE Navigation Unit sending data to be displayed on the cockpit control/display unit. To get data from the INS/DNS Simulator back to the Model, another normal information transfer type message is used to transfer the data from the INS/DNS Simulator (remote terminal) to the INS/DNS Model (bus controller). This message transfers seven data words from the INS/DNS Simulator to the INS/DNS Model in the simulation computer and is shown in Figure 6. The information contained in the data words is the status of the cockpit control/display unit and any information the operator wants to input to the Carousel IVE Navigation Unit, such as a new waypoint. The INS/DNS Simulator consists of three major parts: a Remote Terminal, a ARINC 561 Transmitter Unit and a ARINC 561 Receiver Unit. Before these major parts are described, the ARINC 561 Bus messages and their transmission rates are described.

ARINC 561 Data Bus Messages

ARINC 561 Transmitted Message. The ARINC 561 message the Carousel IVE navigation unit transmits to the Bus Subsystem Interface Unit is shown in Figure 7. This message is constructed by the INS/DNS Simulator from the data passed to it by the INS/DNS Model. The message is divided into Frames that are 600 milliseconds long. The Frames are divided into Subframes that are 50 milliseconds in length. There are 12 Subframes in every Frame. Within every Subframe there is time for 12 ARINC 561 words to be transmitted. Delco Systems, in designing this message, chose to have an eight bit gap between words. A word is then 40 bits long (8 label + 24 data + 8 gap = 40) and takes 4.16 milliseconds to be transmitted. There are two data words, Ground Velocity, and Heading that are sent in every Subframe. In the first five Subframes, five data words are transmitted. This is shown in Figure 8. The first three words are various navigation type data as listed before. The last two

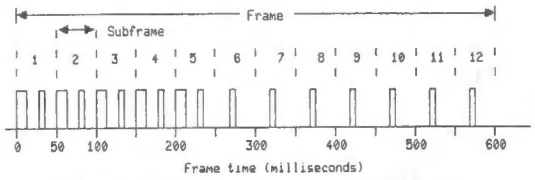
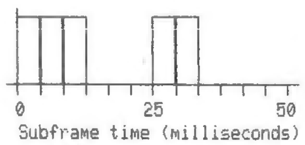
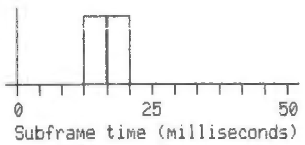


Figure 7 INS/DNS Simulator ARINC 561 Transmitted Message



Subframes 1 - 5
Figure 8



Subframes 6 - 12
Figure 9

words are Ground Velocity and Heading. In the sixth through twelfth Subframes, two data words are transmitted. This is shown in Figure 9. The words are Ground Velocity and Heading.

ARINC 561 Received Message. The ARINC 561 message the Bus Subsystem Interface Unit transmits to the Carousel IVE navigation unit is shown in Figure 10.

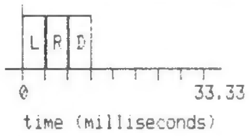


Figure 10 ARINC 561 Received Message

The message is 33.33 milliseconds in length which is enough time for eight 561 data words to be transmitted. A message can contain one, two or three words but must always contain the Discrete word. The different combinations of words that the message can have are shown in Figure 11. The Discrete word, denoted by a D in Figures 10 and 11, contains the status of the cockpit control/display unit. The other two words, called Left and Right and denoted by a L and R respectively in Figures 10 and 11, contain information input by the operator. Whichever message combination is transmitted, the Left, Right and Discrete words are always in their assigned time slot as shown in Figure 11.

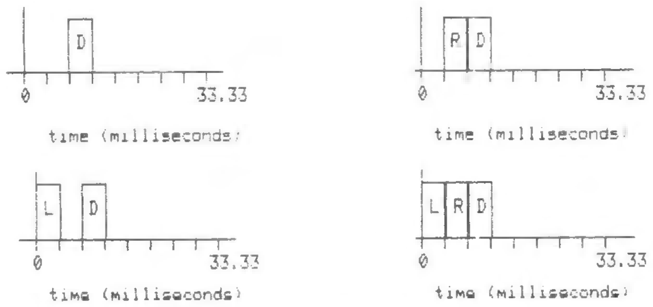


Figure 11 ARINC 561 Received Message Combinations

Transmission Rates

The software executive program in the simulation computer is designed to initiate MIL-STD-1553B bus information transfers (messages) according to a schedule. Transfers can be initiated at a 2 Hz or multiple thereof rate (2, 4, 8, 16, 32, 64 Hz etc.). For transferring data from the INS/DNS Model to the INS/DNS Simulator, a 2 Hz rate was chosen. This results in the data required to form one ARINC 561 Transmitted message (one Frame) arriving at the INS/DNS Simulator every 500 milliseconds. Since a ARINC 561 Frame is 600 milliseconds in length, new data to form the next Frame is always available on time. A 32 Hz rate was chosen to transfer data from the INS/DNS Simulator to the INS/DNS Model. This results in a request for data by the bus controller every 31.25 milliseconds. New data is received by the INS/DNS Simulator from the Bus Subsystem Interface Unit every 33.33 milliseconds. Note that the bus controller is requesting data at a slightly faster rate than it is received by the INS/DNS Simulator. Therefore, the hardware of the INS/DNS Simulator had to be designed to keep pace.

Remote Terminal

The Remote Terminal portion of the INS/DNS Simulator is a separate unit previously developed by SEAFAC. It meets all MIL-STD-1553B requirements and uses the encoder/decoder integrated circuit designed by Harris for the MIL-STD-1553B Bus protocol. The SEAFAC Remote Terminal is based on an Advanced Micro Devices 2910 bit-sliced sequencer. It is completely contained on an 8 inch by 9.5 inch wire-wrap board. It requires voltages of +5 VDC, +15 VDC and -15 VDC. The SEAFAC Remote Terminal can communicate with up to 13 subsystems, although in this application only one subsystem is addressed. Sixteen parallel data lines and various control lines are used for this communication. The data transfers are based upon a strobe scheme which is summarized in Figure 12. Both 4 MHz and 2 MHz clock signals are provided by the Remote Terminal for synchronization purposes. When the Remote Terminal wants to pass data to a particular subsystem or have the subsystem provide it data, it holds the subsystem's associated enable line low.

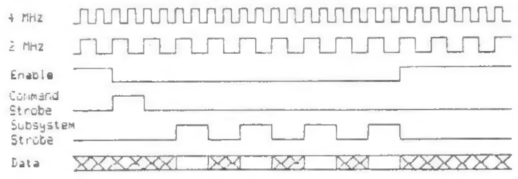


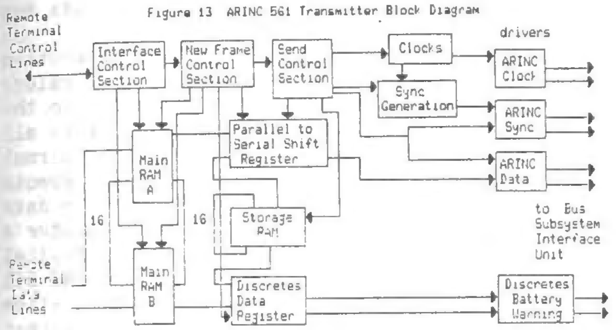
Figure 12 Remote Terminal Data Transfer Timing Diagram

A strobe line, called the command strobe, is used to note when the data lines contain the 16 bits of data from the MIL-STD-1553B command word. Another strobe line, called the subsystem strobe, is used to note when the data lines contain the 16 bits of data from the MIL-STD-1553B data words. There are as many subsystem strobe pulses as there are data words. Another line, called the Transmit/Receive line, is used to tell the subsystem in which direction the data is to be transferred. The SEAFAC Remote Terminal can transfer data to and from the subsystem at a fast (1 MHz) and slow (50 kHz) rate depending upon the subsystem needs. The data rate is automatically indicated by the subsystem to the Remote Terminal.

Design Approach

Aspects of the FSA/CAS Program dictated certain requirements in addition to the actual performance specifications for the Hot Bench. These, in turn, became design guidelines for the hardware and software. These requirements were: there was to be only one ground based Hot Bench built, the Hot

Bench development schedule was tightly coupled to the actual avionics production schedule, and it was to be completed at a minimum of cost. Environmental and cost considerations dictated the use of the 74 series of integrated circuits with the commerical temperature range of 0^{o} C to 70^{o} C rather than the 54 series with the miltary temperature range of -65^{o} C to 150^{o} C. Also, since high volume production procedures were not required, the wire-wrap technique of circuit interconnection was chosen. Wire-wrap techniques are also more suited to an engineering development environment such as the Hot Bench, where design changes can be quickly incorporated. A discrete digital logic design was chosen as opposed to a microprocessor or sequencer based design because it could be designed and tested in a short period of time, at a minimum of cost and since most of the intelligence for the simulation of the INS/DNS would be in the INS/DNS Software Model.


Remote Terminal Control Lines
Interface Control Section
New Frame Control Section
Send Control Section
Clocks
Sync Generation
drivers
ARINC Clock
ARINC Sync
ARINC Data
Parallel to Serial Shift Register
Main RAM A
Main RAM B
Storage RAM
16
16
to Bus Subsystem Interface Unit
Remote Terminal Data Lines
Discretes Data Register
Discretes Battery Warning
Figure 13 ARINC 561 Transmitter Block Diagram

561 Transmitter

The block diagram for the 561 Transmitter portion of the INS/DNS Simulator is shown in Figure 13. The 561 Transmitter is contained on a 8 inch by 9.5 inch wire-wrap board. It requires voltages of 5 VDC, 12.5 VDC and 28 VDC. The 5 VDC is for the digital integrated circuits; the 12.5 VDC is for the ARINC 561 drivers and the 28 VDC is for the ARINC 561 discrete drivers. The 561 Transmitter uses small scale integration (SSI) and medium scale integration (MSI) digital integrated circuits of the Low Power-Schottky Transistor Transistor Logic (LSTTL) type family. A total of 93 integrated circuits are used in the 561 Transmitter along with 41 resistors and 2 capacitors.

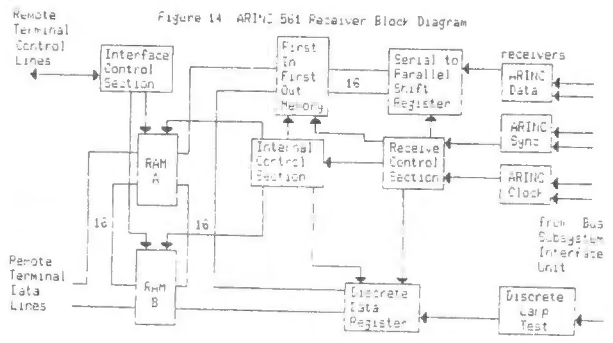
Operation Overview. Initially, the 561 Transmitter is in a wait state until it receives the first transfer of 31 data words from the Remote Terminal. Upon completion of the first transfer from the Remote Terminal, the 561 Transmitter leaves the wait state and enters its normal operating state. It initializes its circuits and starts transmitting the 561 message. It uses the contents of the 31 MIL-STD-1553B data words for the contents of the 561 Data Words. During the non-transmitting time of Subframe 12 (Figure 7), it takes the next 31 data words transferred from the Remote Terminal, and the initialization and transmission process repeats. If the Remote Terminal should stop providing data to the 561 Transmitter, it continues sending 561 messages using the last valid set of data words it received. If the Remote Terminal starts transferring data to the 561 Transmitter again, the 561 Transmitter returns to its normal operating state.

Operation. Data transferred from the Remote Terminal to the 561 Transmitter is alternately stored in one of the two main Random Access Memories (RAMs) which are shown in Figure 13. This double-buffering scheme is used so the newest data received by the 561 Transmitter from the Remote Terminal is always used. Also, since the incoming data rate (500 milliseconds) and the outgoing data rate (600 milliseconds) are not the same, as previously explained, the double-buffering allows the Remote Terminal to be storing data in one RAM while the 561 Transmitter is accessing the other RAM to construct the 561 message. There is a separate Interface Control Section of the 561 Transmitter which controls all the interaction of the 561 Transmitter with the Remote Terminal. For example, when the Remote Terminal's subsystem strobes are occurring, indicating the data words are on the parallel data lines, the Interface Control Section enables one of the main RAMs so the data words are written (stored) in the RAM. The New Frame Control Section controls the initialization for the next ARINC 561 Frame that will be transmitted. First, it decides which of the main RAMs has the newest MIL-STD-1553B data words to use for the next Frame. It reads the first MIL-STD-1553B data word from the main RAM selected and loads the data word, which contain the 561 discrete values, into a register where they are fed to the discrete drivers. These drivers take a Transistor Transistor Logic (TTL) level signal and output a discrete voltage level, in this case 28 VDC, via an on chip output transistor with a uncommitted collector. The New Frame Control Section then loads the next MIL-STD-1553B data word into two 8 bit parallel to serial shift registers in preparation of the start of a new Frame. Since the data words are received and stored in a 16 bit parallel format, the parallel to serial shift registers are used to convert the data from a parallel to a serial format. The Send Control Section, as its name implies, controls the sending or transmission of the signals on the 561 Bus. This is done through the use of Programmable Read Only Memories (PROMs). The Send Control Section is made up of a "Frame PROM" and a "Subframe 1 PROM", "Subframe 2 PROM", "Subframes 3-5 PROM" and a "Subframes 6-12 PROM". The ARINC 561 clock and sync signals are continuously generated by the clock and sync generation circuits. The Frame PROM enables the appropriate Subframe PROM during its subframe. The Subframe PROMs generate enable signals which allow the 561 clock, sync and data signals to be transmitted at the appropriate time during the Subframe. Controlling the message format generation with the PROMs allows changes to be made to the format simply by programming the new format into new PROMs. The Storage RAM is used to store the Ground Velocity and Heading data words since they are only transferred once per Frame from the Model to the INS/DNS Simulator, but are needed in every Subframe. The ARINC drivers are the same as the discrete drivers, except they change TTL level signals to the 12.5 volts DC level of the 561 bus signals.

561 Receiver

The block diagram for the 561 Receiver portion of the INS/DNS Simulator is shown in Figure 14. The 561 Receiver is also contained on a 8 inch by 9.5 inch wire-wrap board. It requires voltages of 5 VDC, +15 VDC and -15 VDC. The 5 VDC is used for the digital integrated circuits and the + and - 15 VDC for the analog integrated circuits. The 561 Receiver uses SSI and MSI digital integrated circuits of the Low Power-Schottky TTL (LSTTL) type family. The 74LS series of integrated circuits are used. In addition, analog integrated circuits consisting of voltage comparators and operational amplifiers are used to receive the 561 bus signals. A total of 79 integrated circuits along with 52 resistors and 2 diodes are used in the 561 Receiver.

Figure 14 ARINC 561 Receiver Block Diagram


Remote Terminal Control Lines
Interface Control Section
First In First Out Memory
16
Serial to Parallel Shift Register
receivers
ARINC Data
ARINC Sync
ARINC Clock
RAM A
Internal Control Section
Receive Control Section
16
16
from Bus Subsystem Interface Unit
Remote Terminal Data Lines
RAM B
Discrete Data Register
Discrete Lamp Test

Operation Overview. The 561 Receiver is in a wait state until it receives the first 561 message sent. If the Remote Terminal requests data before this happens, the 561 Receiver will transfer the random data that is in its RAMs. The wait state is left for the normal operating state as soon as the first 561 message is received. The 561 Receiver will always provide the last complete 561 message received to the Remote Terminal when it requests data. If, after receiving 561 messages for a period of time the 561 messages stop, the 561 Receiver will keep transferring to the Remote Terminal, if requested, the last complete 561 message it received. If 561 messages start being received again, the 516 Receiver will return to its normal operating state.

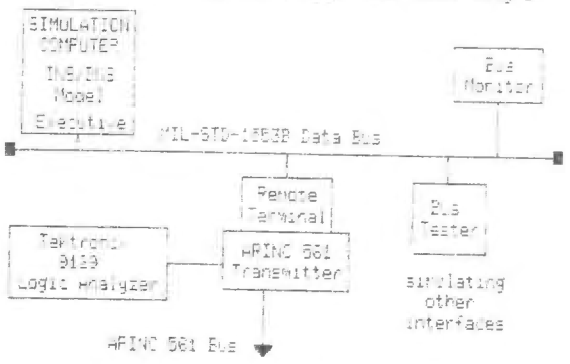
Operation. 561 Bus signals transmitted by the Bus Subsystem Interface Unit are received by the 561 Receiver. The 561 Receiver knows when a message is being sent by monitoring the 561 sync line for activity. The receivers transform the 561 Bus voltages to a TTL level signal. The 561 Data signal is then sent to two 8 bit serial to parallel shift registers (Figure 14), where it is changed to a 16 bit parallel format. From there the data is loaded into a fall-through First-In First-Out (FIFO) memory integrated circuit. Data loaded into a FIFO falls through to the outputs and is available for use. The FIFO is similar to a RAM as data can be stored and then retrieved later. But the designer does not have to worry about the storage addresses of the data as in a RAM, since the order they are loaded in, is the order they are retrieved (read out). The 561 discrete signal, which is the Lamp Test signal, is received and also changed to a TTL level signal. Its value is stored in a register. After the 561 Discrete Data Word is received, signifying the end of the data words in that message, the Internal Control Section of the 561 Receiver first loads the Lamp Test bit and then the received data into a RAM from the register and FIFO. When the Remote Terminal requests the 561 Receiver provide it data, the Interface Control Section reads the data out of the RAM and sends it to the Remote Terminal. A double-buffer RAM scheme is used so data can be provided to the Remote Terminal while new data is stored in the other RAM by the Internal Control Section.

Integration

Integration of the INS/DNS Simulator and the INS/DNS Model was completed in a two step process. The 561 Transmitter and the Remote Terminal were integrated first, followed by the 561 Receiver and the Remote Terminal. During this integration testing, the other hardware interfaces were simulated with a MIL-STD-1553 bus tester, either a Loral Serial Bus Analyzer or Fairchild Data Bus Monitor/Controller.

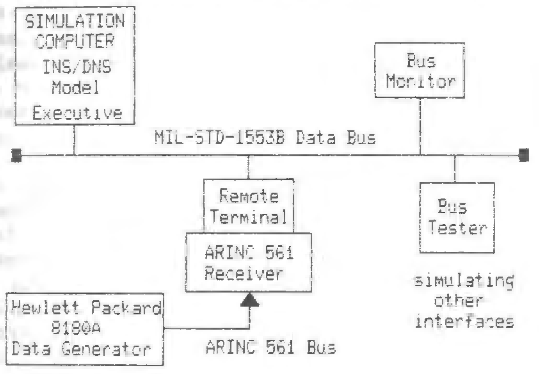
561 Transmitter. Using the software executive program in the simulation computer, various values for the different data words were inserted into the INS/DNS Model. The executive program started all the models and the INS/DNS Model began its normal operation of transferring data words to the Remote Terminal. The Remote Terminal then passed the data on to the 561 Transmitter. The 561 Transmitter's output was checked using a Tektronix 9129 Digital Analysis System. A diagram of the ARINC 561 Transmitter, Remote Terminal, and Model integration test is shown in Figure 15. The 561 message format was checked as well the contents of the data words for agreement with the inserted values.

Figure 15 ARINC 561 Transmitter Test Block Diagram



561 Receiver. For the 561 Receiver to be tested, typical 561 Bus signals had to be generated and sent to the 561 Receiver. This was accomplished by generating simple 561 messages representing each of the four possible combinations with a Hewlett Packard 8180A Data Generator and sending them one at a time to the 561 Receiver. The executive program started all the models and the INS/DNS Model began its normal operation of requesting data from the INS/DNS Simulator. The contents of the MIL-STD-1553B data words in the message between the INS/DNS Model and Simulator were checked with the Loral or Fairchild MIL-STD-1553 bus tester in a bus monitor mode. A diagram showing this integration testing is shown in Figure 16. The executive program was used to interrogate the INS/DNS Model for the data received as another check of the contents of the data words for agreement with the values originally loaded into the Hewlett Packard Data Generator.

Figure 16 ARINC 561 Receiver Test Block Diagram



Conclusions

Due to the requirements of the Hot Bench, the approach selected for the design of the INS/DNS Simulator proved to be acceptable. The choice of a discrete digital logic design resulted in a simple, low cost design. However, the use of PROMs for message formatting in the Simulator gives it flexibility without the extra complexity and cost associated with being completely programmable. The basic concept of changing parallel data format words to a serial format and then transmitting them over a serial digital data bus can be applied to other ARINC buses. Different drivers could be used to achieve the different voltages and encoding required by the other buses.

Improvements. Two improvements that could be made to the INS/DNS Simulator are as follows. The first would be to utilize Programmable Array Logic (PAL, which is a registered trademark of Monolithic Memories Inc.) integrated circuits in place of AND and OR gate circuitry used to allow two separate circuits to both access RAMs in the double-buffering schemes. The PALs would be a cost effective improvement because of their high ratio of replacement of the AND and OR gates, ease of programming, and that these access circuits are all similar and used in four different places within the INS/DNS Simulator. The second improvement involves the counter used to count the number of words transmitted in a Subframe in the 561 Transmitter. The counter is presently set to count 12 words. With a small circuitry addition, the counter could be able to count any number of words in a Subframe, thus making the 561 Transmitter completely changeable as to words in a Subframe and Subframes to a Frame.

Summary

Even though the MIL-STD-1553B and ARINC 561 Buses were developed for different applications and with different philosophies, the successful development and integration of the INS/DNS Simulator proves that a data bus interface for the Buses was feasible. The feasibility is further reinforced since the INS/DNS Simulator was achieved using common electronic components and at a low cost.

References

(1) Air Transport Inertial Navigation System - INS, Aeronautical Radio Inc. Characteristic Number 561-11, 17 January 1975

(2) Digital Data System Compendium, Aeronautical Radio Inc. Report 419-1, 1 December 1975

(3) Delco Systems Operation, -135 FSA/CAS Data Bus Interface Control Document, EE-75-S-649, 21 September 1983, Revision 11

(4) MIL-STD-1553B, Aircraft Internal Time Division Command/Response Multiplex Data Bus, 21 September 1978

84-2757

AI CONCEPTS IN MUX BUS CONTROL: LAYERED RESERVATION ACCESS

Brian J. McNamara*

Principal Engineer
ARINC Research Corporation
2551 Riva Road
Annapolis, Maryland 21401

Abstract

This paper begins with a review of common non-deterministic CSMA/CD and deterministic token passing high speed bus techniques. The less studied Reservation Access bus allocation approach is then examined. The Reservation Access approach is then expanded into layers of topological trees based on a system's functional distribution. This leads to the concept of using dependency analysis to generate the topological tree. Such dependency analysis techniques have been successfully employed to develop system fault isolation trees. The paper then suggests combining such dependency paths with processing time data bases. This could create an expert system, capable of inferring rulesets. These rulesets could be used for generating reconfigured systems to perform new functions, using existing system resources. Under these circumstances, the bus controller becomes an important system processing element.

Introduction

The evolution in signal processing systems has resulted in deployment of more highly distributed and parallel processing machines. The elements of these machines require a high speed busing concept that provides speed and bandwidth and takes advantage of system slack time. Traditional CSMA/CD and token passing buses are not suited to all situations and standardization has been difficult.

Reservation Access is a less studied approach that tries to offer the best of both worlds. By expanding the approach to capitalize on system slack time, an efficient bus approach could be developed. The development of the required topological tree might be eased by using dependency analysis techniques already developed in the testability field.

The recent advances in Artificial Intelligence also suggest that a modified Reservation Access approach might be useful in configuring next generation machines through bus allocation. This paper raises the issues and suggests possible areas for further research.

*Member IEEE

The Non-Deterministic Contention Approach (CSMA/CD)

Traditional approaches to developing a high speed multiplex data bus have been divided between deterministic and non-deterministic bus implementations. These implementations are based on Local Area Network (LAN) technology which accommodates traditional Von Neumann machines. Non-deterministic approaches are also known as contention busses. Each node contends for access and control of the bus whenever the node has a message to be transmitted. Because a node must compete with other nodes for bus access and control, at any point in time a node has a finite probability less than 1 of actually gaining access to the bus. Hence, the bus is non-deterministic because a node is never guaranteed access to the bus.

The most common non-deterministic contention bus approach is the Carrier Sense Multiple Access with Collision Detection (CSMA/CD) bus[1]. Each node on the CSMA/CD bus monitors the energy level or average voltage on the bus in order to detect the presence of bus activity. A node can begin sending its message as soon as it detects there is no other message activity on the bus. Beginning a message transmission places activity on the bus and locks out the other nodes until the transmitting node detects an acknowledgment from the destination node. This acknowledgment signals that the message transfer is complete and the bus is available for other nodes requiring access. However, message collisions can occur. Assume node 1 attempts to transmit a message to node 5. As a result of propagation delay, node 4 may attempt to transmit a message before it senses that the message going from node 1 to node 5 already occupies the bus. The resulting message collision destroys the integrity of both messages and further raises the energy level or average voltage on the bus (called a "jam").

To accommodate such message collisions, the transmitting nodes must monitor the bus energy both before message transmission (to gain access to the bus) and during message transmission (to detect message collision). When transmitting nodes detect a collision, they then execute a backoff and re-try scheme. Both nodes cease transmitting for some time period and then attempt retransmission. The time period between jam detection and retransmission can be random or programmed. The random backoff risks another message collision. The programmed backoff has the advantage of allowing high priority nodes to retransmit before lower priority messages.

The advantage of CSMA/CD lies in a node's nearly instantaneous bus access when the bus is unoccupied. However, as the number of messages to be transmitted on the bus increases, the probability of collision increases. The overhead associated with collision resolution and backoff schemes reduces effective usage of the bus. The problem compounds if message strings are short, because more messages are likely to be transmitted. Thus the CSMA/CD bus access approach is more appropriate for systems using a relatively low number of long message strings.

The Deterministic Approach

Unlike CSMA/CD approaches where bus access is only a probability for each node, deterministic approaches guarantee a node bus access at regular intervals. The most common deterministic approach allocates bus access and control via a bit pattern, or token, passed along the bus from station to station[2]. The node possessing the token controls access to the bus. The token holder can transmit a message and release the bus when the node receives an acknowledge signal from the destination device. A time-out feature prevents bus hang-up caused by a destination node incapable of acknowledging message receipt. A token possessing node with no message to transmit forwards the token to the next node in the sequence.

Nodes can receive the token in sequence around a ring, along a bus or in some other programmed order. Programming the token passing sequence has the advantage of allowing more frequently needed nodes, or higher priority nodes, to be serviced with the token more often.

The primary advantage of token passing is that it guarantees a node access to the bus at regular intervals depending on the node's priority and the number of times the node is needed during a sequence cycle.

The main disadvantage of token passing is that a node's messages wait to be transmitted while other nodes have possession of the token. Since the token must propagate through the remainder of the node sequence, the message at the first node is delayed until the token returns. A token propagation delay occurs even if each remaining node in the sequence transmits no message. Moreover, message transfer efficiency drops, since the bus is idle even though a node needs to transmit. The problem compounds and messages accumulate in a node's transmission queue, if intervening nodes transmit long, time consuming data strings. Thus, unlike CSMA/CD, token passing appears to be most appropriate for networks whose nodes can be expected to transmit a large number of relatively short messages. The large number of messages maximizes bus usage (minimizes idle time). Short data strings cause the token to revisit nodes more often, preventing message accumulation in transmission queues.

The Reservation Access Approach

The Reservation Access Approach attempts to combine the features of CSMA/CD and token passing to obtain the best of both worlds.[3] The Reservation Access Approach uses a topological tree search to allocate bus access. In figure 1, higher priority nodes, which transmit first, are located down and to the left. At t_o (1 in the tree) nodes 0, 1, 4 and 7 all have messages to transmit. The bus controller immediately inhibits all nodes to the right of point 1 in the tree because their tree position indicates they are lower priority messages. Nodes 4, 5, 6 and 7 inhibit. Messages from terminals 4 and 7 transmit later in the sort cycle. The sort then moves to 2 . Messages from terminals 0 and 1 collide and the hierarchical sort process repeats, inhibiting terminal 2 and 3 and advancing the sort to step 3 . Another collision between node 0 and node 1 messages occurs and the sort moves down and left to node 0, location 4 in the tree. At this point no collision occurs and node 0 transmits.

The sort then moves right to location 5 . No messages collide and node 1 transmits. The sort moves up and right again to location 6 , detects no messages and moves up and right to 7 where node 4 and 7 messages collide. Using the same approach, the sort moves down and left allowing node 4 to transmit, location 8 . The sort then moves right and node 7 transmits (location 9) completing the cycle.

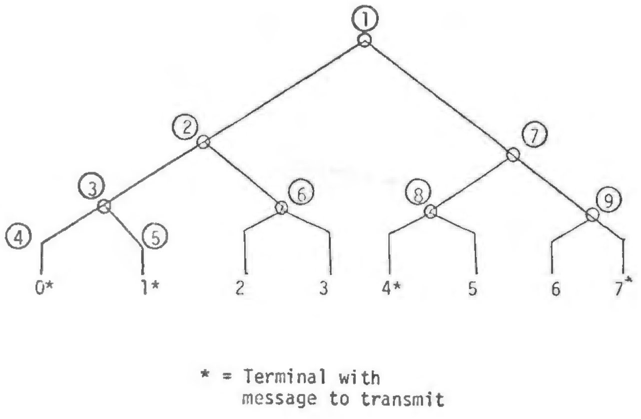


Fig. 1 Reservation access topological tree.

Several features of reservation access are immediately obvious. When there is no other bus traffic, a sending node acquires access to the bus immediately, without having to wait for a token. Thus, under light loading, Reservation Access offers the efficient immediate bus access of CSMA/CD. As message traffic increases, the number of collisions increases. When all nodes have a message to transmit, bus access occurs in sequence from left to right. This approximates token passing. Thus, if heavily loaded, the Reservation Access method resembles token passing.

The topological tree sort can be arranged to permit high revisit rates for very high priority nodes or nodes frequently requiring access. For example if node 0 requires a high revisit rate, node 0 can be placed in the tree several times. Figure 2 shows a case when node 0 is revisited 3 times in a heavily loaded situation. Since Reservation Access approaches CSMA/CD under light load and token passing under heavy load, Reservation Access appears to offer the best of both worlds for managing message traffic.

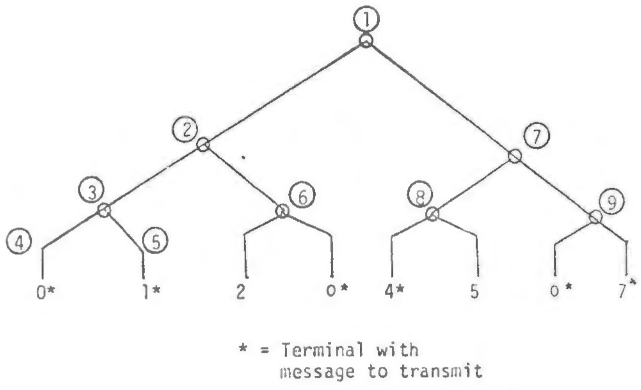


Fig. 2 Reservation access with revisit of terminal 0.

Layering Reservation Access

The Reservation Access approach can be expanded to a layered approach to increase message transfer efficiency. This approach involves placing any number of layers of topological tree sorts within each other, to allow simultaneous resolution of bus access conflicts. The approach is most easily described assuming subsystems with local buses communicating with each other over a global bus. In Figure 3, layer 1 collision resolution is fast, since the network need allocate access only between subsystems 1, 2 and 3. While subsystem 2 transmits its message over the bus, subsystems 1 and 3 resolve localized conflicts. Subsystems 1 and 3 then transmit their highest priority messages when they regain control. The reduced conflict resolution overhead in layer 1 approaches CSMA/CD, allowing subsystems near instantaneous access under light loading. Frequent access under heavy loading occurs because there are relatively few major subsystems.

Frequent updating is important in aircraft and EW systems where current environmental and performance data are required. In this configuration, subsystem 1 resolves a localized layer 2 conflict between A and B in favor of message A. When subsystem 1 obtains bus access, it transmits message A and passes control to subsystem 2 or 3. Message B is not transmitted until subsystem 1 again gains control. If subsystem 2 responds with a message eliminating B's need to transmit, no time was lost transmitting B's useless message. If no other subsystems seek control, or if it was not overcome by events in the other terminals, B's message will transmit. All the other

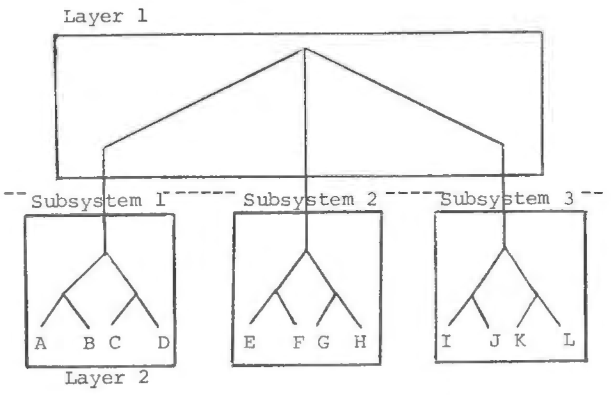


Fig. 3 Layered reservation access.

Reservation Access rules apply. High priority subsystems which require revisits before completion of a worst case cycle can be inserted in the tree at any level, any number of times. The problem, of course, is that this approach requires careful subsystem choice. Improper choice of revisit rate and sample rate may result in data lockout of some messages for long periods.

The Impact of AI - Reconfigurable Layered Reservation Access

The proliferation of distributed processing concepts and the coming of next generation data flow machines, suggest an opportunity for using a reconfigurable layered reservation access bus as a processing control element. This approach involves allocating bus access and control among functions rather than subsystems.

Artificial intelligence concepts are useful in this context. A truly distributed processing system transmits information to the processor who can make the best use of it at the time the information becomes available. This is a difficult real time decision and traditionally the bus controller plays no role in deciding where information is sent. However, the bus controller can be a powerful system asset.

Performing system functions requires consuming and allocating system resources in an efficient manner. Determining a function's most efficient processing path begins with defining data flow dependencies. For any function to be performed, the system resources can be allocated by calculating higher order dependencies from first order dependencies. The object is to optimize parallel processing of information in distributed processing systems. Determining first order dependencies is a matter of analyzing the higher level function performed, and identifying the sequence and timing of inputs and resources required to perform the function.

In figure 4 assume that equal message length resources R1, R2, R3 and R4 are all available at t_0 and that subfunction operations are performed in equal time intervals. Figure 4 shows that in performing the function F, at time t_o, subfunction f_1 requires inputs from resources R1 and R2 and subfunction f_2 requires an input from resource R4 only. Although both processes could proceed simultaneously the bus can carry only a single message. The dependency of $f_1(t_1)$ on the $f_2(t_2)$ output indicates that f_2 should be allocated resources first. Therefore, the bus is allocated to f_2 to receive input from R4. While f_2 processes information from R4, f_1 can occupy the bus. Since $f_1(t_1)$ requires information from R1 and R2, the most efficient processing is to allocate the bus to transmit R1 information first. (When f_2 is ready to receive input from R2, f_1 can also pick up that data. This eliminates the need to transmit R2 data twice.) At t_2, no collision occurs because f_2 does not require the bus. The bus can be allocated to send R3 data to f_1. At t_3, f_2 data is transmitted to f_1 and the output to E is transmitted next.

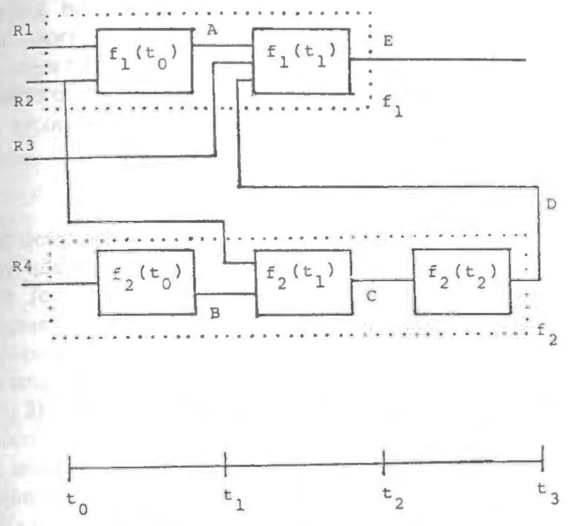


Fig. 4 Example processing system.

The resulting topological tree that maximizes parallel processing in resolving bus conflicts is shown in figure 5.

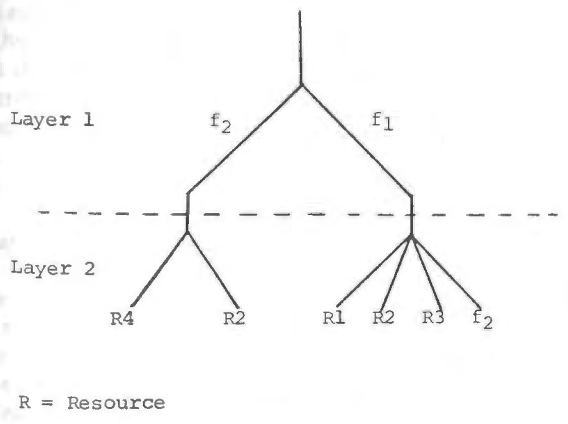


Fig. 5 Topological tree for example system.

In layer 1 subfunctions f_1 and f_2 collide. The competition for the bus is resolved in favor of f_2. In layer 2, no collision occurs, since $f_2(t_0)$ does not need R2 data, and R4 transmits. The bus is then allocated to f_1 where the layer 2 potential collision between R1 and R2 is resolved in favor of R1. Back in layer 1, f_1 and f_2 both want the bus. f_2 gets control to receive R2 data which f_1 also receives. f_1 then has control of the bus which it allocates to receive R3, while f_2 processes R2 and $f_2(t_1)$ results. f_1 then receives the f_2 subfunction result. The layering of the tree enhances processing speed by allowing R1 data to transmit prior to R2, since R2 data is required by both subfunctions later in the cycle.

The simplicity of the example makes the processing sequence, if not the topological tree, almost obvious. In this case, the designer can see the interrelationship of subfunctions at various points in time. The designer has deduced the higher order dependencies. As a system's complexity increases the mental bookkeeping becomes impossible. A model, in which the designer need only input the first order dependencies is required. Similar dependency models have been used in designing for testability.[4]

In this case the first order dependencies can be represented as a series of inputs and subfunctions performed in time. The first order dependencies are:

A depends on R1, R2, $f_1(t_0)$
B depends on R4, $f_2(t_0)$
C depends on R2, B, $f_2(t_1)$
D depends on C, $f_2(t_2)$
E depends on A, R3, D, $f_1(t_1)$

Higher order dependencies result from combinations of first order dependencies. Using these inputs, a dependency model calculates the higher order dependencies. This higher order dependency set can form the knowledge base for an expert system to develop initial topological trees. Such a dependency model could be coupled with a data base containing processing times. The resulting expert system could then be used to define processing rules for the subfunctions. Rules and dependencies can be evolved to whatever layer of detail is required and new topological trees developed. Such models do exist. Dependency modeling has been shown effective in the testability field.[5] Further developments in artificial intelligence may soon put such models within the reach of reconfigurable bus architectures.

An important advantage of the above approach is that a bus controller now becomes an active processing element that can be programmed to reconfigure the system. Since the bus controller determines when functions get access to their required resources, the bus controller could be used to implement new functions by redefining resource allocation. This could be done off line or in parallel to data processing with minimum overhead. All reservation access rules apply, except that layering the approach would enhance the ability to perform parallel processing.

Conclusion

This paper suggests that a bus controller can be used as a programmable element to reconfigure a system by allocating resources on the bus based on functional requirements. The approach would involve hierarchically defining first order dependencies among functions and subfunctions and utilizing a data base containing typical processing times. Although dependency models have proven effective in the testability arena, further research is required to implement the above approach to bus control.

References

(1) Introduction to Local Area Networks, Digital Equipment Corporation, 1982

(2) J. Bondy and R. Weaver, "Bus Schedule Change Issues in the On-Board Distributed Data System (ODDS)," 1983 Proceedings of the IEEE/AIAA Digital Avionics Systems Conference, Seattle, Washington, November 1983

(3) Bart Stuck, "Which Local Net Bus Access Is Most Sensitive to Traffic Congestion," Data Communications, Jan. 1983

(4) W. R. Simpson and H. S. Balaban, "The ARINC Research System Testability and Maintenance Program (STAMP)," 1982 Proceedings of the IEEE AUTOTESTCON Conference, Dayton, Ohio, October 1982

(5) W. R. Simpson, J. R. Agre, "Experience Gained in Testability Design Tradeoffs," 1984 Proceedings of the IEEE AUTOTESTCON Conference, Washington, D.C., November 1984

84-2756

HIGH-SPEED BUS STRUCTURES FOR MULTIPROCESSING

Quentin E. Dolecek

The Johns Hopkins University Applied Physics Laboratory
Laurel, Maryland 20707

Abstract

Parallel processing — the application of several processors to a single task — imposes stringent performance requirements on the bus structure used to connect the processors. This paper surveys several common bus architectures and presents an operational 64 megabytes per second synchronous bus supporting both data and control communications between processors. The bus also supports "party lines," which allow messages to be sent to multiple destinations. An asynchronous version of the bus that supports data rates of 100 megabytes per second is also described.

Introduction

The designer of a high-speed distributed parallel processor must address four difficult issues: data transfer rates, provision for interunit communications, coordination of the processing between units, and network reliability. The bus structure used in the system affects all these issues. Parallel systems have used crossbar switches and separate control buses (1 to 3) to communicate between units. However, substantial system time overhead is incurred in those networks (4) with the selection of a ready task through a scheduler, the initialization of processors, communication delays when one task waits for the results from another, and system cleanup when a task finishes. The SPAN system (5,6) uses a single unidirectional ring bus to transfer software, data, and control between processor units. In addition, the bus structure permits the use of a unique software structure that allows the system to control all units, each operating at full capacity if required.

A brief description of several of the more common bus structures provides an introduction to the options available to the system designer. The SPAN bus structure is then described as the starting point for understanding a unique asynchronous bus developed for systolic and wavefront array processors operating with data transfer rates of up to 100 megabytes per second.

Bus Architectures

Shared Bus Network

Bus networks use a common shared data channel for communication. Any processor may transmit a message or data to any other processor by means of the shared bus (Fig. 1). In a bus grant mode of operation (7), a single bus controller grants bus access to the various users. For interconnecting a number of relatively high-speed processors, the contention allocation approach (8), in which all users have free access to the bus, is popular. In that case, the processor transmits to the bus and monitors the bus to verify successful transmission. If another processor was transmitting at the same time, both must retransmit the messages.

Only one processor at a time may use the bus successfully. This is a serious limitation because a single processor may require 12 to 25 million messages per second to utilize its capacity fully.

Star Network

The star network consists of a central switching hub with data channels from the hub to the processing elements (Fig. 2). Data paths can be established between two or more of the processing elements via the switching and routing structure in the hub. Although the hub may be a crossbar structure, there is usually a matrix switch or data multiplexer at the hub input and a programmable multiple-tap demultiplexer at the hub output. This configuration allows multiple destinations to receive

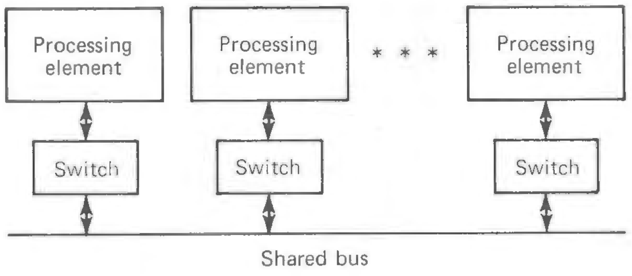


Fig. 1 Shared bus network.

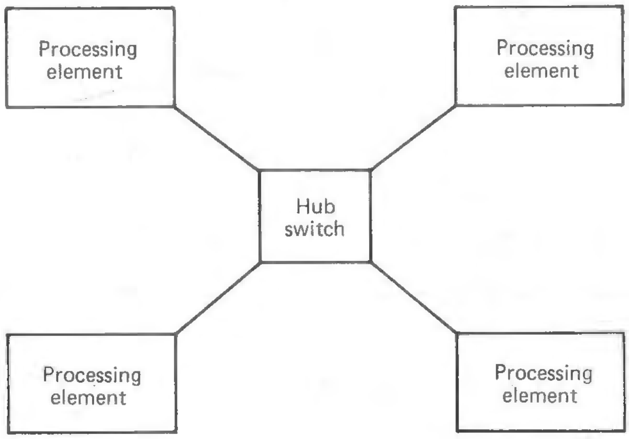


Fig. 2 Star network.

a data stream. In either case, only a single data source may be active at any one time. Thus, star networks are unsuitable for multiple simultaneous sources, and they have the same accumulative communication bandwidth problem as the preceding bus structure. Generally, the hub also contains a special processor that receives requests for service and programs the hub data path, leading to complex mechanisms for controlling the network.

Fully Connected Network

Fully connected networks (Fig. 3) establish direct point-to-point data communications among all nodes in the network. Although too complex for most large networks, fully connected networks are applicable to small very-large-scale integration (VLSI) systems that have high communication requirements. Although the complexity of fully connected networks (9) can be reduced by using a sparse network in which an irregular topology uses data links only as required for a specific application (10), the complexities of these networks is still roughly proportional to the number of processors in the system.

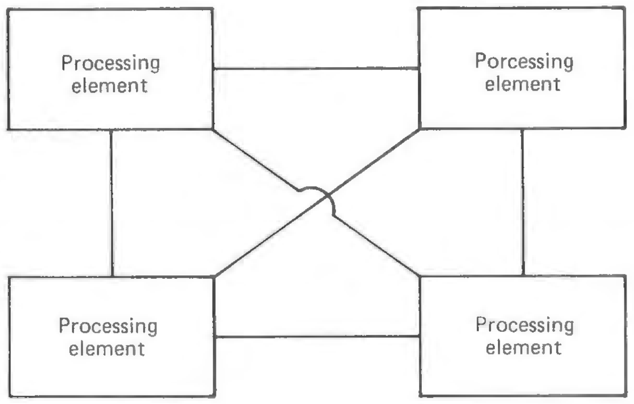


Fig. 3 Fully connected network.

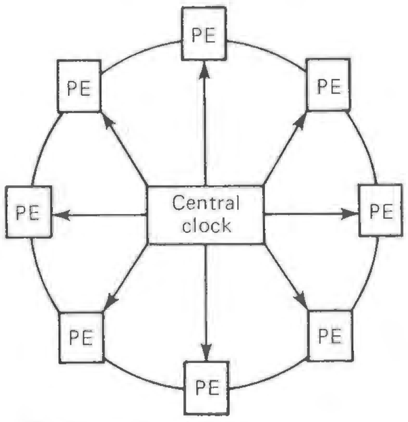
Ring Networks

A ring network is a sequential connection between processors, with the output of processor j connected to the input of processor j + 1 and so on until the output of processor N is connected to the input of the first ring node (Fig. 4). In the usual configuration, one node processor is used for input/output to the ring and a central clock is used for synchronous transfer of data around the ring. For large (long) rings, clock skew can be significant. One network structure for reducing the effect of clock skew is the distributed clock ring (Fig. 5). In that network, each processing element uses the clock to its left for communication with the element to its left and generates a new clock for communication to the right. Thus, data transfers are resynchronized with the clock at each processing element.

The addition of command/data message packets to a ring network enables each processing element to communicate with all other processing elements in the network without direct communication links. The command is a high-level processor operation (such as "load a program" or "read data from the bus"); the address is either for a particular processor element or for a subsystem in a processor element. Each processor reads input messages, retransmitting only those with destination addresses other than its own. Thus, as messages are used they are stripped off the bus; as long as each processor sends messages only to its adjacent processor, communication bandwidth does not accumulate along the bus.

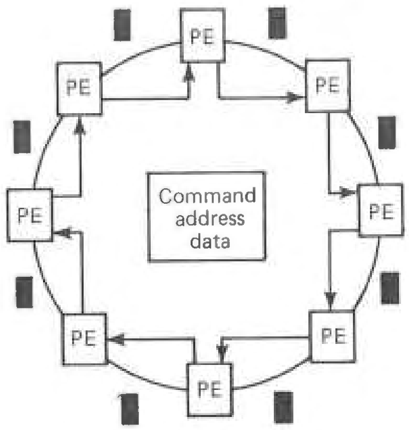
SPAN Bus Structure

The SPAN ring bus interfaces all system resources to the bus via a processor. Thus data input, output, bulk memory, computational hardware, etc. are all accessed through a processor element. This approach provides several useful system features: (a) very high communication rates using an interface that can check data and request validity for a node, implement special protocols, and route communication through a node; (b) programmable interfaces that may be modified via software to provide the flexibility needed to support different subsystem requirements within the bus structure; and (c) separation of system processing software from the software of individual nodes.



PE = Processing element

Fig. 4 Synchronous ring bus.



PE = Processing element

Fig. 5 Command/data packet driven distributed clock ring bus.

The synchronous SPAN-I bus described in Ref. 5 supports eight million messages per second, and each message contains 4 bytes of data. Thus, the bus can provide data rates of 32 million bytes per second between each pair of processors on the bus. A loop synchronization module provides clocks for the bus, clears the bus of illegal messages, and fills the bus with blank messages (identified by a zero in the message code field) upon power-up.

The bus is designed to handle three kinds of data: real-time or continuous data streams, blocked data such as arrays of data from a bulk memory, and control messages that initiate action and synchronize processing units. All three types of data can be transferred simultaneously and independently on the bus. In addition, the bus supports simultaneous multiple data transfers without interference. This is necessary in order to achieve high efficiency in the use of a bus with a large number of processors. When a processor is ready to output data to another unit on the bus, it can do so immediately without waiting for the bus to shift control from one unit to another. This capability eliminates the need either for large buffer memories at the output of each processor or for the complex scheduling of data transfer events on the bus.

A message on the bus consists of five fixed blocks: data, address, source code, destination code, and a message type code. The destination code is the address of the processor to which the message is being sent; the source code is the address of the processor sending the message. Combining this information with the ring structure of the bus permits the detection of messages that were not received without having to use acknowledge messages. The address is the local address destination within a processor. It may be the location to read or write into in the case of a memory operation, the location to begin program execution within a processor, or the location of a cross-mapping function in the case of a multiple-destination message. The message type code determines the function of the message such as read data from the bus, write data to the bus, begin a processor function, etc.

If a unit wishes to output data, it replaces a blank message with its own message for output and sends it, instead of the blank message, to the next unit on the bus. When a unit receives a message, it compares its position code with the destination code of the message. If the two codes match, the unit accepts the message into its memory and substitutes a blank message or an output message for passing on to the next unit. If they do not match, the unit simply passes the message on to the next unit.

Each processor is connected to the bus through a bus interface unit (Fig. 6). The input interface examines each message on the bus; if the message is for that unit, it is stored in a first-in/first-out (FIFO) memory in the input interface for transmission to the attached processor under processor control. The output interface may do one of three things: pass the received bus message to the next unit, strip the message and send a blank message to the next unit, or strip the message and send a message from the attached processor to the bus. The output interface also contains a FIFO memory for buffering the data between the attached processor and the bus. The interface contains decoders and logic for all bus operations. Thus, the processor receives input data and formats data for output; all bus control overhead is handled in hardware by the interface.

Each interface unit may also contain an address generator section that can interrupt the processor or a control processor (via a message type code) when a specified transfer is complete and can insert addresses into the address block of an outgoing message.

Control messages on the bus can set up all input and output interface control registers to identify which data streams are to be accepted and stripped from the bus (final destination) and which are to be accepted and not stripped (party-line input). This feature allows the same data stream to be accepted and processed by several units on the bus, if required.

Each unit on the bus is set to output at some rate that is less then the bus rate. In this manner, each unit passes along data messages and some blank messages for use by succeeding units. Each unit then has some blank messages coming along at all times for its use. As long as the peak bus load at any one point does not exceed the bus rate, data transfer occurs smoothly. The total bus capacity typically is much greater than the peak bus rate because most messages travel only a short distance on the bus before being stripped off. If there are N units on the bus and each unit sends messages only to its adjacent unit, the bus capacity is N times the peak bus rate.

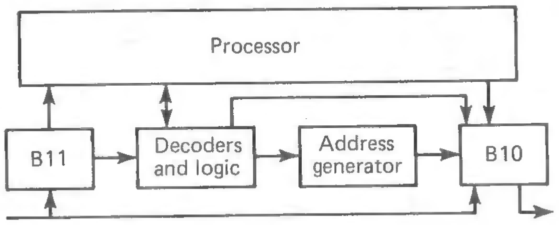


Fig. 6 Ring bus interface.

Asynchronous Ring Bus

The previous ring network can also be implemented with an asynchronous data transfer structure in which each processing element has a bidirectional buffer, independent status, and control flags for handshaking with adjacent elements (Fig. 7). This structure permits message transfers without clocking between elements, providing data transfer rates of up to 100 million data bytes per second, using 25 million messages per second, and permitting local synchronous and asynchronous processing elements on the bus. Interface circuits are simplified in that blank messages are not required. Messages are stripped off the bus simply by not being forwarded to the next unit.

Because asynchronous buses and protocols are often poorly specified and difficult to understand, they typically are viewed as being inferior to synchronous buses. However, a properly defined and correctly applied asynchronous bus and protocol actually result in a superior bus structure. Asynchronous buses can achieve the same communication speeds as synchronous buses and are not constrained by a system clock, so that system speed is not fixed, which means that processors that have been designed with today's technology will work with tomorrow's faster systems. The older technology module will slow

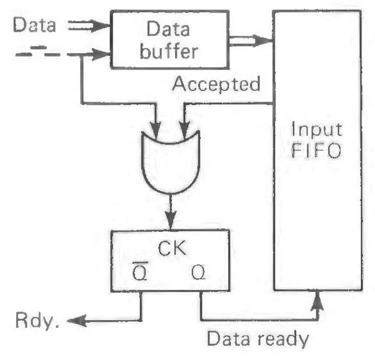


Fig. 7 Asynchronous handshaking.

a transfer only when it is actually participating in operations. Even then, it will still work. In addition, as there is no centralized clock or bus elements, modules can be plugged into any position along the bus. There are no rules for bus layout to keep clock skew under control.

With the decentralized architecture of an asynchronous protocol, the bus is more reliable because there are no central control components to fail and bring the bus down. In their basic form, ring bus structures are vulnerable to failures in the communication links between the processor nodes. One approach for improving the fault tolerance of rings is to braid the network. Each processor output node is routed to the input of its immediate and its second successor. Thus, a link or processor failure can be bypassed, permitting performance to degrade gracefully or redundant processors to be placed in the ring. However, the approach does increase the pin count at each processor node.

Ring nodes can be implemented with about 50 to 100 SSI/MSI chips. This total includes handshaking control or clock buffers, line drivers, and FIFO buffers in each node. Alternatively, ring ports can be implemented in VLSI with less than 5000 gates. Achievable ring data communication rates range from 10 MHz for commercial TTL, to 50 MHz for commercial ECL, to over 100 MHz for custom on-chip VLSI designs.

Conclusion

This paper has examined bus structures for parallel processing. The use of synchronous versus asynchronous buses depends on the network size. Asynchronous timing incurs a fixed time overhead because of the handshaking process, whereas synchronous time delay is due primarily to clock skew, which increases with bus length. This leads to the concept of globally asynchronous bus structures supporting synchronous processing elements to merge the merits of both timing schemes. Asynchronous ring bus structures have been shown to be attractive for large systems of processors and for mixing processors implemented in different technologies.

Acknowledgment

Research funding has been provided by the Applied Physics Laboratory Independent Research and Development Program.

References

(1) J. S. Thompson, "Digital Signal Processor Architecture for Voice Bank Communications," National Telecommunications Conference Record, pp. 501-506: 1974.

(2) R. Steinmetz et al., "Realization of Digital Filter Algorithms by Use of a High Speed Parallel Processing Architecture," ICASSP '83 Proc., IEEE 83CH1841-6, Piscataway, N.J., pp. 1188-1191: 1983.

(3) M. J. Knudsen, "MUSEC, A Powerful Network of Signal Microprocessors," ICASSP '83 Proc., IEEE 83CH1841-6, Piscataway, N.J., pp. 431-434: 1983.

(4) C. Weitzman, Distributed Micro/Minicomputer Systems, Englewood Cliffs, N.J., Prentice-Hall: 1980.

(5) W. D. Ashcraft and H. M. South, "Architecture and Control of a Distributed Signal Processor," IEEE International Conference on Acoustics, Speech, and Signal Processing, pp. 44.8.1-44.8.4: Mar., 1984.

(6) H. M. South, "Digital Recording and Signal Processing Systems for Hydrophone Arrays," Johns Hopkins APL Technical Digest, Vol 4, No. 3: 1983.

(7) J. V. Levy, "Buses, the Skeleton of Computer Structures," Chapter 11 in Computer Engineering, Bedford, Mass., Digital Press, pp. 269-299: 1978.

(8) R. M. Metcalfe and D. R. Boggs, "Ethernet: Distributed Packed Switching for Local Networks," Comm. ACM, Vol. 19, pp. 395-403: 1976.

(9) D. Katsuki et al., "Pluribus—An Operational Fault-Tolerant Multiprocessor," Proc. IEEE, Vol. 66, pp. 1146-1159: 1978.

(10) E. E. Swartzlander, Jr., and D. J. Heath, "A Routing Algorithm for Signal Processing Networks," IEEE Transactions on Computers, Vol. c-28, pp. 567-572: 1979.

SESSION 14 CREW SYSTEMS - SYSTEMS DEVELOPMENT AND TEST

Chairmen:

Wayne D. Smith
Boeing Commercial Airplane Co.

14

This session considers advanced man-vehicle interface systems, including the design, development, and testing of such systems. Actual equipment and the use of simulation are addressed

84-2686

AN ADVANCED MEDIA INTERFACE FOR CONTROL OF MODERN TRANSPORT AIRCRAFT NAVIGATIONAL SYSTEMS

Denise R. Jones
Russell V. Parrish
Lee H. Person Jr.

NASA Langley Research Center
Hampton, Virginia 23665

and

Joseph L. Old

Research Triangle Institute
Research Triangle Park, North Carolina

Abstract

As part of the effort to improve information management systems, an advanced concept for managing the navigational tasks of a modern transport aircraft, which emphasizes the simplification of the pilot interface, was studied at the NASA Langley Research Center. The system incorporates automation and intelligence into the pilot interface. Two multimode devices, a Bowmar multifunction keyboard and a Video/Graphics Terminal, were chosen to function in concert as a Navigation Control Display Unit (NCDU). The devices eliminate the need for dedicated switches in the cockpit since they can be programmed to display various menus. The navigational system is menu driven through the multifunction keyboard. The Video/Graphics Terminal is used mainly for displaying data and entering new flight plans. From preliminary evaluation it was found that certain solvable human factors problems exist in the design of the system. Also, the current implementation of the system is somewhat slow. However, the pilot who performed the preliminary evaluation was pleased with the general concept of the navigational system and feels it will greatly simplify the navigational tasks of the pilot.

Introduction

As modern transport aircraft become increasingly more sophisticated with the advent of digital avionics, pilot workload to manage the newly available information becomes much heavier. In order for advanced crew stations of the future to become more efficient they will require technologies that allow for multi-purpose, consolidated controls and displays; highly reliable operations; reconfigurable I/O crew/systems interface; intelligent automation and decision aiding techniques; and enhanced information storage capacities. Some of these enabling technologies are flat-panel displays, programmable display generators, multifunction controls, distributive processing, video/data busing, and system integration design techniques to provide for modularly-expandable stations capable of multi-location operation. These technologies have the potential to help reduce pilot workload; reduce training/cross-training requirements; declutter the workstation; improve safety; reduce cost, weight, and complexity of the electronic systems; and improve crew/systems performance.

The need for a more efficient means of pilot data entry evolves from the increase in digital avionics functions implemented in present transport aircraft. With the traditional method of data entry, this increase in functions requires additional dedicated controls and displays to be integrated into the remaining limited amount of cockpit area. Therefore, consolidated controls and displays using multifunction control devices are urgently needed for modern transport aircraft (Ref. 1).

As part of the effort to improve information management systems, an advanced concept for managing the navigational tasks of a modern transport aircraft, which emphasizes the simplification of the pilot interface, was studied at the NASA Langley Research Center. The advanced navigational system provides a simple method for a pilot to enter new waypoints to change his flight plan due to heavy traffic, adverse weather conditions, etc. Waypoints are geographical locations used to comprise the flight plan the aircraft must follow from take-off until the destination airport is reached.

The navigational system was designed with the aid of a NASA research pilot. The system incorporates automation and intelligence into the pilot interface. Two multimode devices were chosen to function in concert as a Navigation Control Display Unit (NCDU). The devices eliminate the need for dedicated switches in the cockpit since they can be programmed to display various menus and can "query" the pilot for various inputs. This should reduce the amount of training from that which is currently necessary for a transport's navigational system. It should also minimize the detailed information which the pilot would be required to remember.

The navigational system was implemented and evaluated in a flight simulator representative of a modern transport aircraft.

Simulator

The flight simulator (Fig. 1) used for this project represents a modern transport aircraft. It is a part-task simulator that is suitable to test the concept of the navigational system. The pilot (left) side of the cockpit consists of two cathode

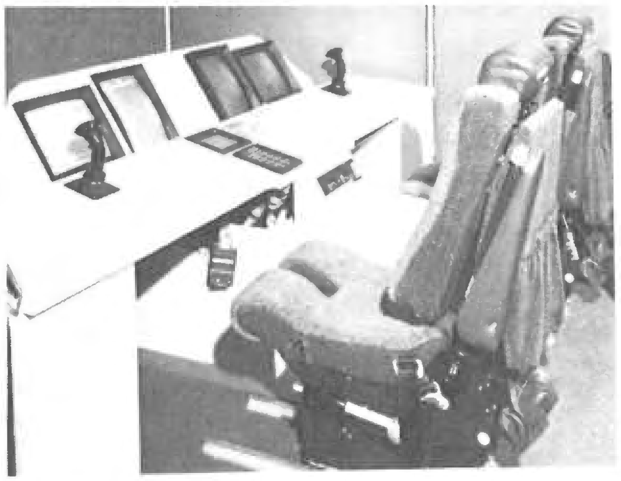


FIGURE 1. Console of Flight Simulator.

ray tubes (CRTs), a side arm controller, and a set of rudder pedals. The throttle is shared between pilot and copilot (seated on right side of cockpit). One CRT is mounted horizontally to display the Electronic Attitude Director Indicator (EADI), also called the Primary Flight Display (PFD), and one is mounted vertically to display the Electronic Horizontal Situation Indicator (EHSI) or Navigation Display. The EADI is an attitude director indicator (ADI)-like display which contains the altitude; ground speed; pitch indices; roll scale and pointer; glide slope deviation scale; localizer deviation scale; perspective runway; and the aircraft symbol. The EHSI is a map-like display which includes time to go (TTG) to next waypoint; distance to go (DTG) to next waypoint; ground speed; vertical deviation scale; aircraft symbol; current flight plan; and track tape, bug, and box. Also installed in the console on the pilot side of the cockpit are two multi-mode devices. These devices are used as a NCDU and are described in detail below.

Flight Simulation

Aircraft Model

The advanced navigational system was incorporated into an existing aircraft model. This model is a linear representation of the NASA-Advanced Transport Operating Systems (ATOPS) Boeing 737-100, a research aircraft that was modified to incorporate electronic displays and all-digital flight-control computers (Ref. 2 to 8).

Geographical Simulation

The simulation is generated around the Denver, Colorado and Atlanta, Georgia terminal areas. The geographical data base developed to test the concept contains six flight plans from Atlanta to Denver (Fig. 2). The flight plans consist of a combination of two Standard Instrument Departure routes (SIDs) fromAtlanta (Ref. 9) and three Standard Terminal Arrival Routes (STARs) to Denver (Ref. 10). In this instance, the enroute distance between the two terminal areas has been shortened to approximately 50 nautical miles due to the length of time required to fly from take-off to landing in the flight simulator. As the SIDs and STARs are standard routes that are usually unalterable, the pilot's navigational management

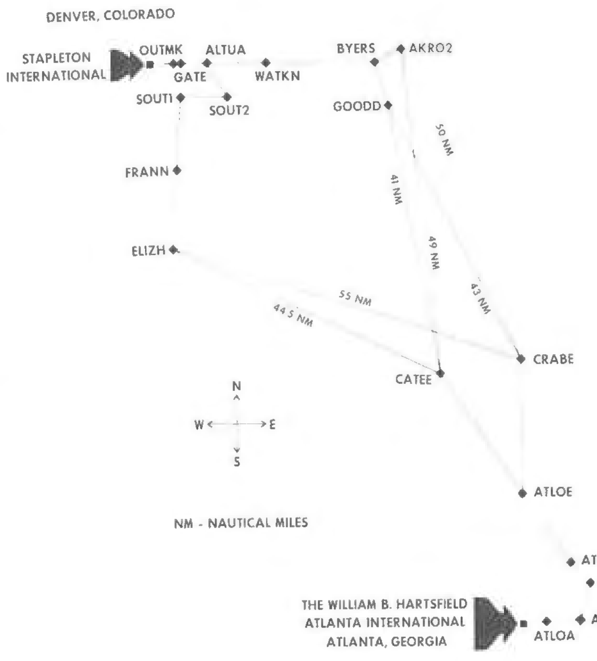


FIGURE 2. Geographical Database.

task involves choosing among alternative routes and introducing/changing the intervening waypoints between the SIDs and STARs as conditions or air traffic controllers dictate. The pilot also manages desired speed and altitude information for each waypoint on the entire route.

Computer Implementation

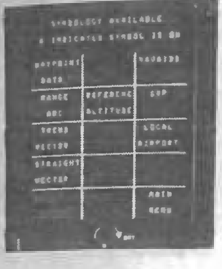
The aircraft model is hosted on a Digital Equipment Corporation (DEC) VAX-11/780 minicomputer. The model is programmed in VAX-11 FORTRAN. The graphic displays (EADI and EHSI) are produced on an Adage 3000 programmable display generator using Ikonas Display Language (IDL).

Multimode Devices

Multifunction Keyboard

One of the two devices used for the pseudo NCDU is the Bowmar multifunction keyboard (MFK), shown in Figure 3 (Ref. 11). The MFK was developed by Bowmar Instrument Corporation under a NASA/Air Force contract. The programmable keyboard consists of a scratchpad area and fifteen light emitting diode (LED) tactile feedback keystations. The scratchpad can display a maximum of 48 alpha-numeric characters while each keystation can display a maximum of sixteen. VAX-11 FORTRAN is used to program the scratchpad and keystations for this project.

One advantage of the MFK is that the LED displays are sunlight readable. The MFK also allows consolidated controls which means less wiring will be necessary than with dedicated controls. The MFK as a single unit would also be easier to replace than a multitude of dedicated controls. A problem could arise with the consolidation of controls in the level of paging that may be necessary to accomplish certain tasks. An effort must be made



- o LED ALPHA-NUMERIC DISPLAYS
  - SUNLIGHT READABLE
  - WIDE FIELD OF VIEW
- o SCRATCHPAD
  - 2 ROWS OF 24 CHARACTERS
- o KEYSTATIONS
  - TACTILE FEEDBACK
  - 16 CHARACTERS PER KEY
- o BRIGHTNESS CONTROL
- o RS-232 AND 1553-B INTERFACES

FIGURE 3. Multifunction Keyboard.

to allow returning to the upper-most level from any menu depth. A human factors guideline is to allow a maximum depth of three levels.

Video/Graphics Terminal

The Video/Graphics Terminal (VGT) is the second device used in the pseudo NCDU (Ref. 12) and is shown in Figure 4. The VGT was developed by Hycom,

- o THIN-FILM ELECTROLUMINESCENT FLAT-PANEL DISPLAY
  - SHALLOW DEPTH
  - ULTRA HIGH MTBF
  - UNIFORM RESOLUTION
  - GRACEFUL DEGRADATION
- o DISPLAY MODES
  - VIDEO
  - GRAPHICS
  - GRAPHICS-OVER-VIDEO
- o SCREEN
  - MEASURES 6 INCHES DIAGONALLY
  - 240 ROWS X 320 COLUMNS PICTURE ELEMENTS
  - TOUCH SENSITIVE OVERLAY
- o RS-232 INTERFACE

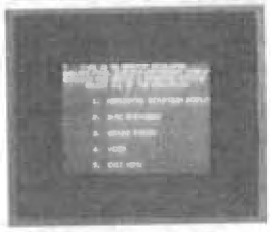


FIGURE 4. Video/Graphics Terminal.

Incorporated under a DOD/NASA contract. The device contains a thin-film electroluminescent (TFEL) flat-panel display with 240 rows by 320 columns of picture elements. The screen measures six inches diagonally and has a touch sensitive overlay which can be used for pilot inputs to the navigational system. The VGT can operate in three display modes - video, graphics, and graphics-over-video. In the video mode, a real-time display can be generated through an external video signal. In the graphics mode, built-in graphics commands allow vector and character drawing. For this project, the VGT operates in the graphics mode. A set of FORTRAN subroutines that were obtained from the U. S. Army Electronic Research and Development Command, Fort Monmouth, New Jersey were used to produce the graphics. Otherwise, assembly language programming would have been necessary. Some advantages of the VGT are uniform resolution, an ultra-high mean-time-between-failures (MTBF), graceful degradation, and shallow depth which permits easy installation into the cockpit. Current disadvantages are poor luminous efficiency, matrix addressing complexity, and high cost.

Device Interfaces

The MFK communicates to the VAX-11/780 minicomputer through a DZ32 RS-232 interface. Eventually, a 1553-B interface will replace the RS-232. This will dramatically increase the rate of data transfer between the MFK and host computer. The present implementation is somewhat slow for this application. The VGT communicates to the VAX-11/780 through a DZ11 RS-232 interface. Both the MFK and VGT are driven by VAX-11 FORTRAN from a detached process. This allows the devices to run seperately from the aircraft model so flight will not be interrupted when input and output are performed, thus allowing the model to run in a real-time environment. The detached processes and aircraft model communicate with each other through global sections which are installed in the computer's memory.

Navigational System

General

The navigational system is menu driven through the MFK. The VGT is used mainly for displaying data and touch entry of new flight plans.

Initially, the MFK displays a main menu which has five selections from which the pilot can choose. The VGT is concurrently displaying zoom in and zoom out areas which, when pressed, make the EHSI larger or smaller respectively (Fig. 5). The zoom areas are always displayed on the VGT unless it is otherwise in use. The following paragraphs will describe in detail what occurs when each selection is made on the main menu.

Main Menu Selections

'SYMBOLOGY' Key. Pressing the 'SYMBOLOGY' key causes a new menu to be drawn on the MFK (Fig. 6). This new menu allows symbology and geographical data to be displayed on the EHSI. An 'X' will appear in the lower right corner of the key if that data is currently being displayed. The information that can be added to the EHSI is waypoint data (waypoint ground speeds and altitudes); an altitude range arc feature which predicts where the aircraft will reach a preselected altitude with the current flight conditions; a trend vector which predicts where the aircraft will be located in 30, 60, and 90 seconds with the current flight conditions; a straight vector to display indicated distances in a straight path from the aircraft; geographical reference points (GRP); navigational aids (NAVAIDS); and local airports (Figure 7 illustrates the EHSI with the straight vector and GRPs added). The 'REFERENCE ALTITUDE' key causes a numerical keypad menu to be drawn on the MFK allowing the pilot to enter a new altitude for the altitude range arc feature. The 'MAIN MENU' key returns the MFK to the main menu. The VGT continues to display the zoom areas during these operations.

'MAP SCALE' Key. Several map scale choices are printed on the MFK when the 'MAP SCALE' key is specified (Fig. 8). When a new scale is selected, the EHSI is changed accordingly. This is comparable to the zoom in and zoom out areas on the VGT. The current scale of the EHSI can be seen in the lower left corner of that CRT. The VGT remains unchanged during this menu also.

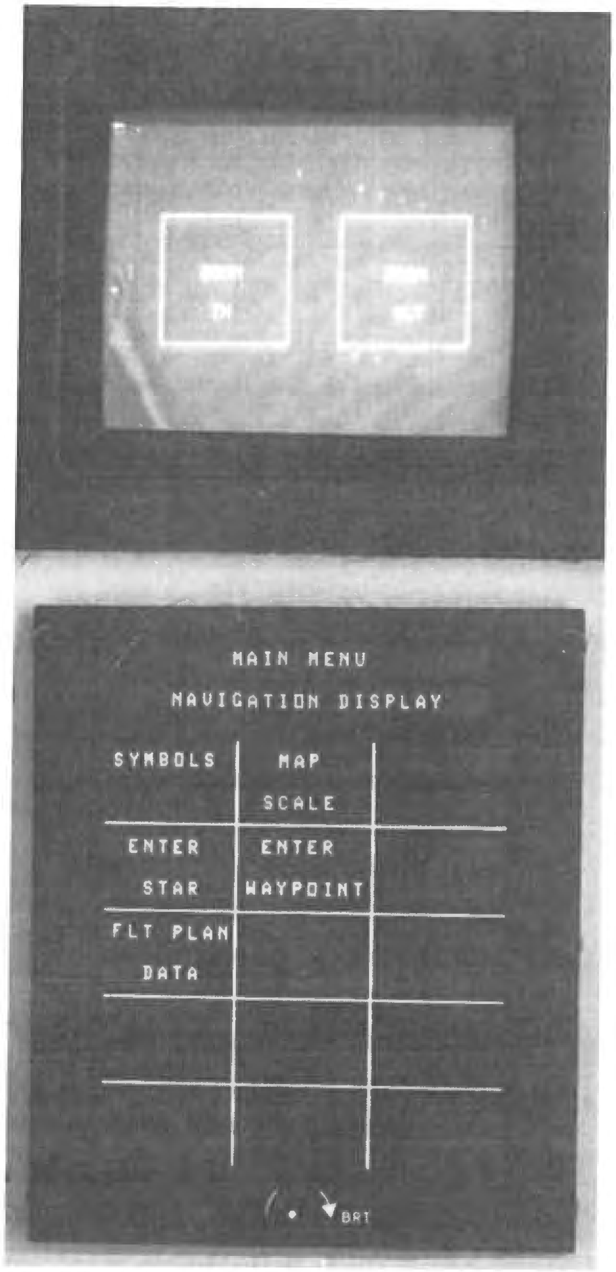


FIGURE 5. Main Menu of Navigational System and Zoom Areas on VGT.

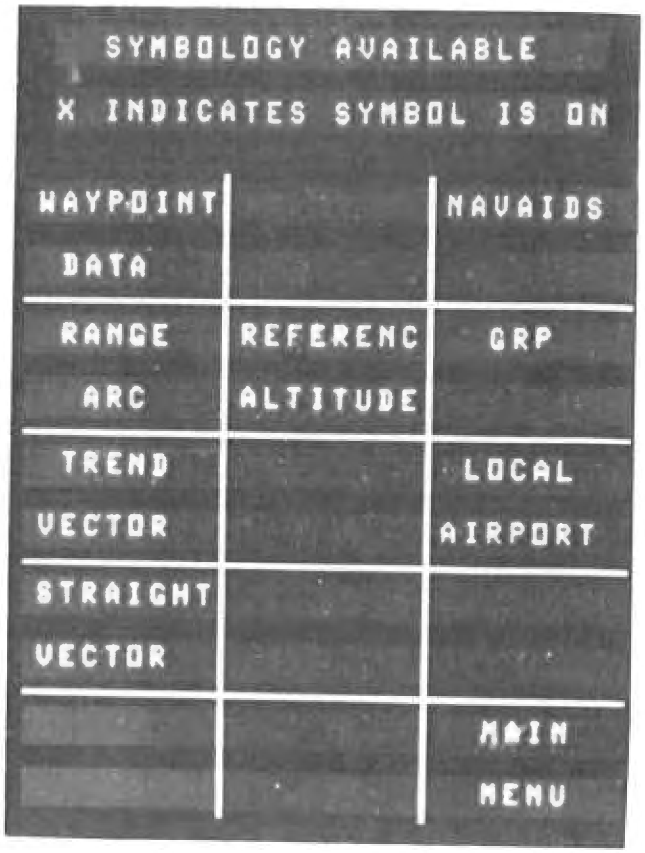


FIGURE 6. 'Symbology' Menu.

'ENTER STAR' Key. Choosing the 'ENTER STAR' key prints the name of each STAR available for the destination airport on the MFK (Fig. 9). When its key is then pressed, that STAR is drawn on the VGT. This allows the pilot to view the STAR before deciding to incorporate it into his flight plan. The 'ACCEPT' key incorporates the STAR that is displayed on the VGT into the current flight plan. The EHSI is then updated to reflect this change (Fig. 10).

'ENTER WAYPOINT' Key. When the 'ENTER WAYPOINT' key is selected, a new menu is printed on the MFK. At the same time, the current flight plan and a row of data manipulation areas are printed on the VGT (Fig. 11). The data manipulation areas allow the pilot, by touch, to scroll left, scroll right, scroll up, scroll down, zoom in, and zoom out the flight plan on the VGT. The 'GRP' and 'NAVAIDS' keys on the MFK allow these two types of geographical data to be displayed on the VGT. When a pilot enters new waypoints, certain guidelines must be followed. The first and last waypoints entered must be waypoints from the current flight plan. The first waypoint entered could also be the present position (PPOS) of the aircraft. The remaining new waypoints can be GRPs, NAVAIDs, or pilot entered points (PEP) which are referenced from existing geographical locations. The methods used to enter these waypoints are by touch on the VGT, by name on the MFK, or by bearing and distance. When entering by touch, the 'ENT WPT BY TOUCH' key is pressed; then the desired location of the new waypoint is touched on the VGT. After this, the pilot is asked to enter the ground speed (Fig. 12) and altitude for the new waypoint on the MFK. These questions may be bypassed and a default value accepted if desired. When 'ENT WPT BY NAME' is pressed, the name of the new waypoint is entered through alphabetic menus on the MFK (Fig. 13). The remainder of this entering process is the same as for entering by touch. When entering by bearing and distance, first the 'ENTER BEAR/DIS' key is pressed. The waypoint location used as the reference is then entered either by touch or by name. After this, the pilot is querried from the MFK to enter the bearing and distance for the location of the new waypoint . The process then proceeds as before. The new waypoints are drawn on the VGT as they are entered so the pilot can see his new flight plan compared to the current one (Fig. 14). The pilot also has the opportunity from the MFK menu to reject the last wayoint entered, reject the entire new flight plan, or accept the new flight plan. When the flight plan is accepted, the EHSI is changed accordingly.

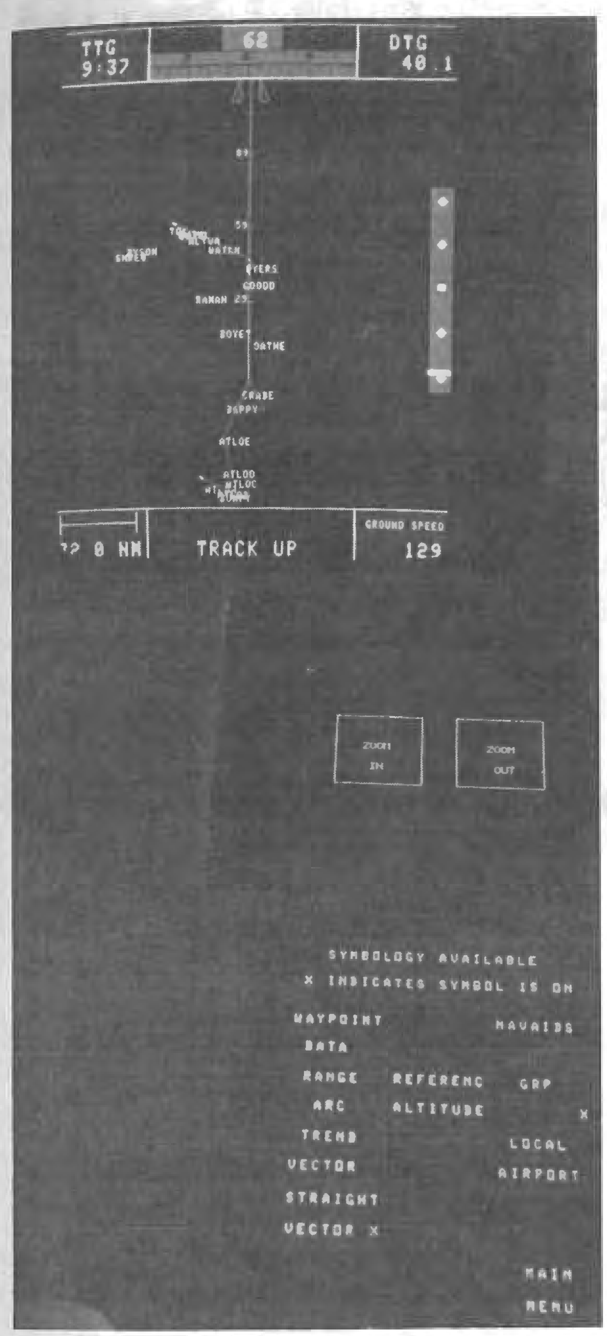


FIGURE 7. EHSI with Straight Vector and GRPs Added, VGT with Zoom Areas, and MFK with 'Symbology' Menu.

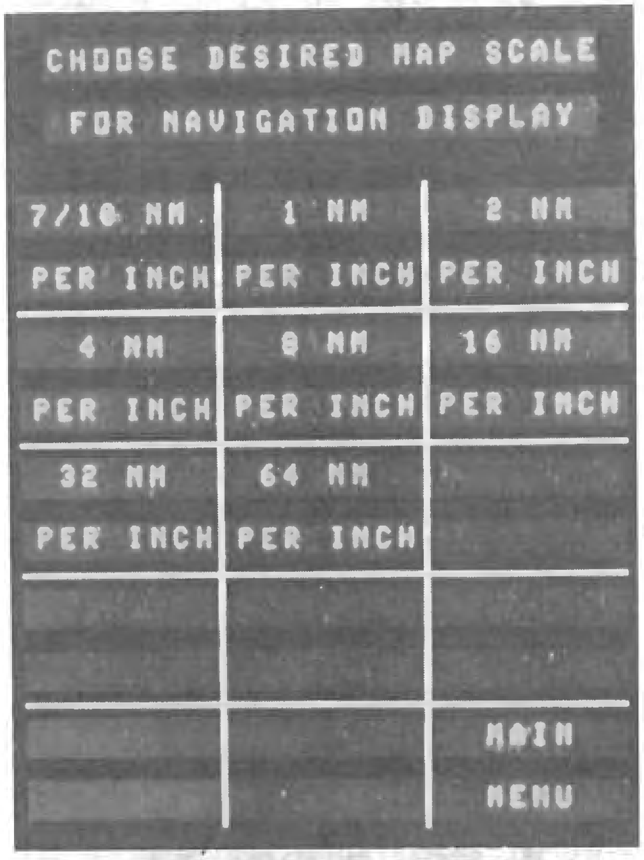


FIGURE 8. 'Map Scale' Menu.

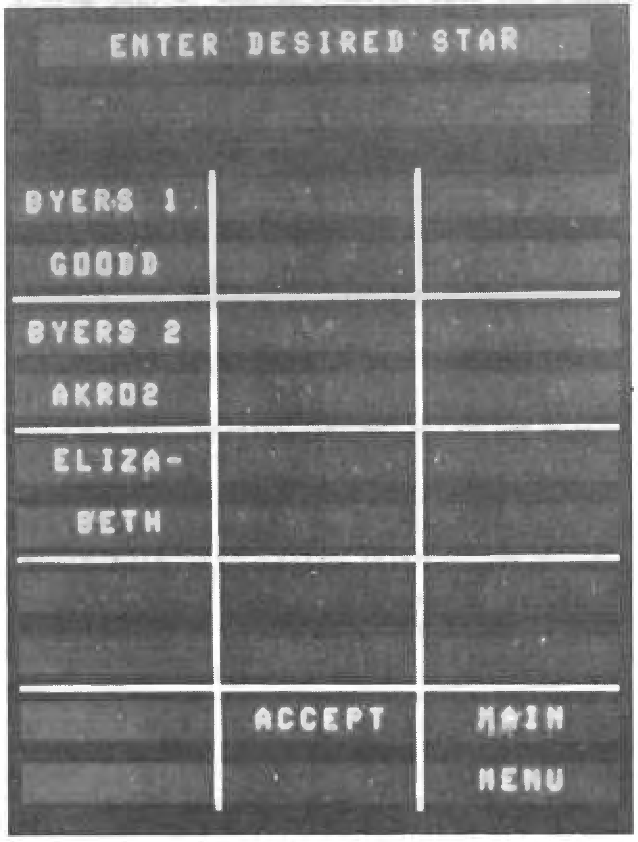


FIGURE 9. 'Star' Menu.

'FLT PLAN DATA' Key. The last key ('FLT PLAN DATA'), when pressed, causes the current flight plan data (waypoint names, ground speeds, and altitudes) to be displayed in tabular form on the VGT.

Concluding Remarks

From preliminary testing, it was found that the devices and navigational system have promise. The pilot who performed the preliminary evaluation was pleased with the general concept of the navigational system. He especially liked the ability to enter waypoints by touch on the VGT and

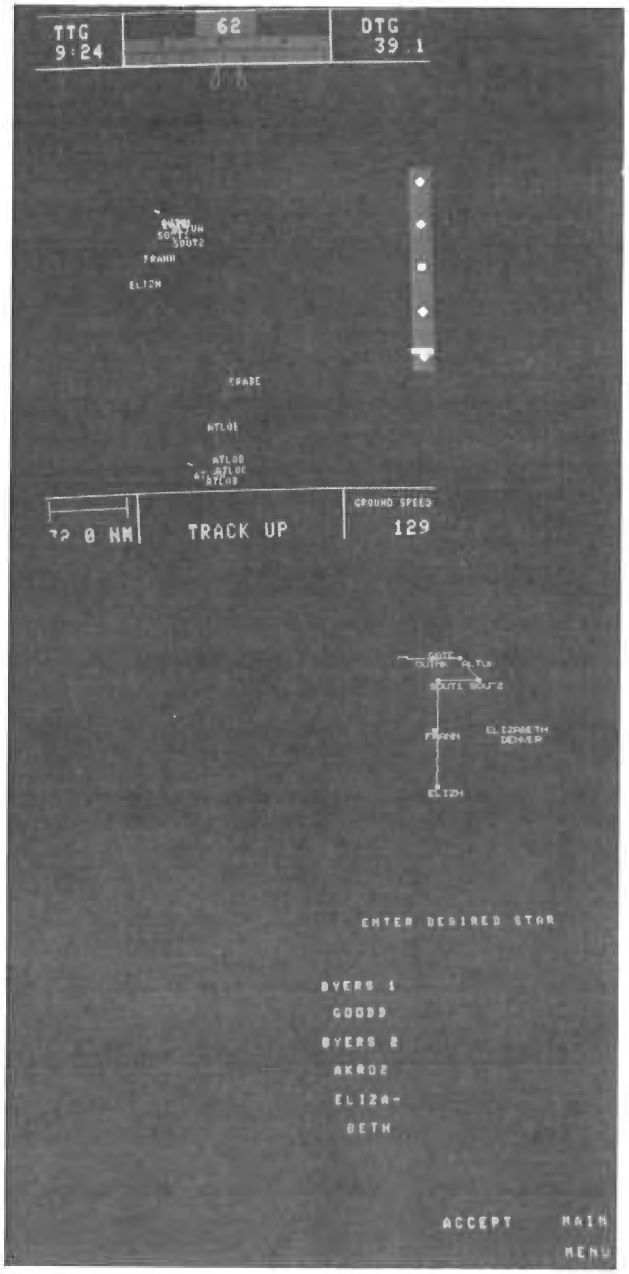


FIGURE 10. Updated EHSI, VGT with Elizabeth Star, and MFK with 'Star' Menu.

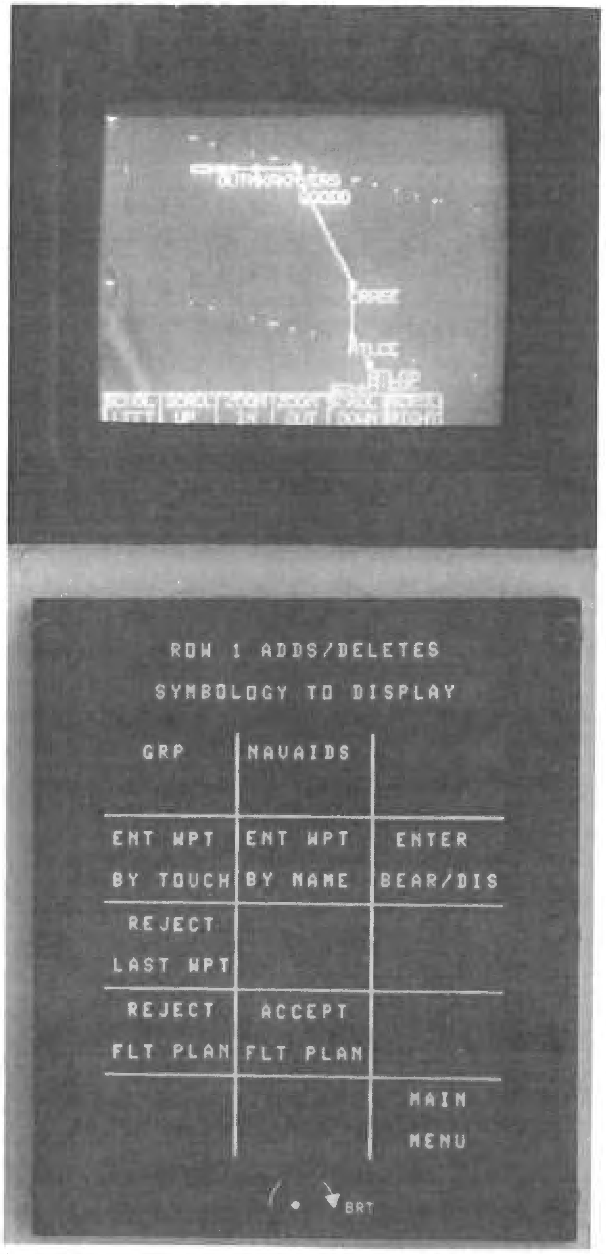


FIGURE 11. VGT Displaying Current Flight Plan and MFK with Menu for Altering Current Flight Plan.

the zoom in and zoom out areas on the VGT to enlarge and shrink the EHSI display. In fact, he predicts that pilots will not use the 'MAP SCALE' MFK function, as it is much easier to hold down the zoom area until the desired map scale is obtained, rather than guessing at a map scale. He also felt that the entire system may be much simpler for pilots to use since it is menu driven and queries the pilot; therefore, less memorization will be required of the operator.

There were some problems noted, however. It was determined that there are several trivial human factors issues involved with the system's design. These include the name selection for the zoom in and zoom out areas on the VGT; the direction of display movement with the scroll up, scroll down, scroll left, and scroll right areas on the VGT; and the format choice for the alphabetic keypad menus on the MFK.

The current implementation of the navigational system is somewhat slow. The use of a 1553-B interface between the MFK and VAX-11/780 will greatly increase the rate of I/O with the MFK. This in turn will improve the performance of the navigational system. Formal evaluation of the updated system will be conducted in the future.

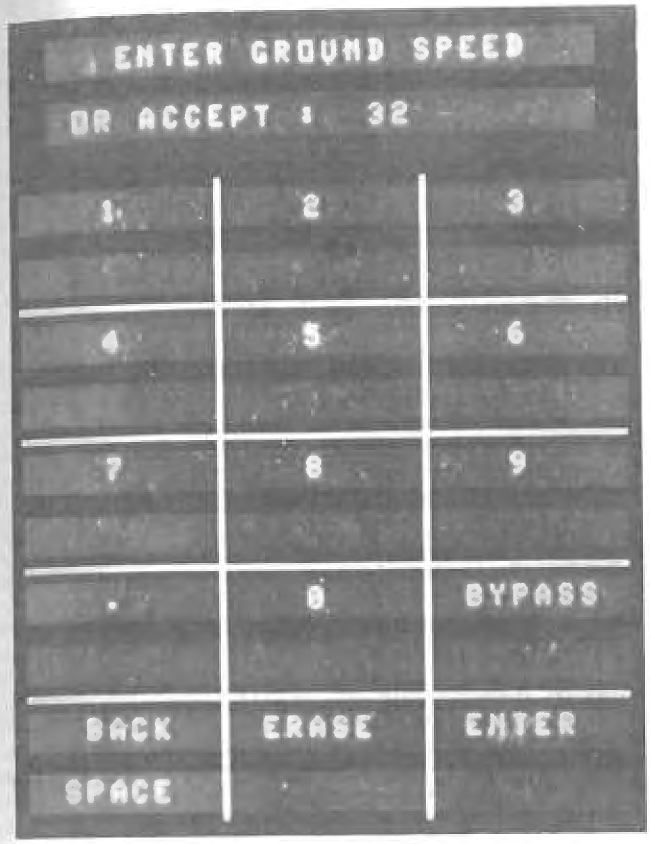


FIGURE 12. MFK Query to Pilot to Enter New Ground Speed or Accept a Default Value.

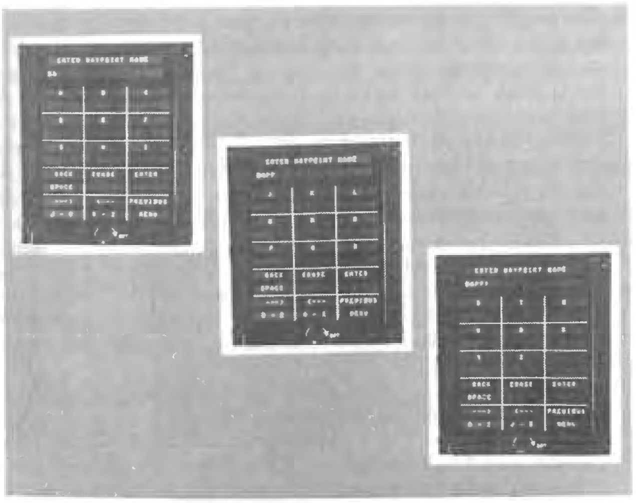


FIGURE 13. Alphabetic Menus for Entering Waypoint Names.

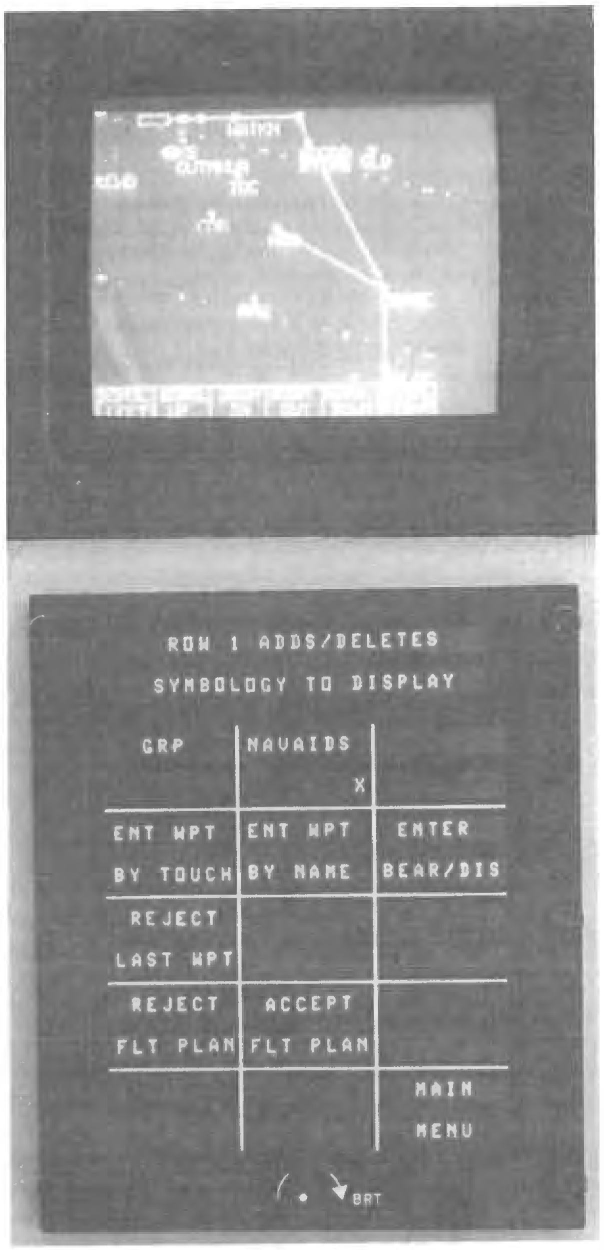


FIGURE 14. VGT Displaying Newly Entered Waypoints with Current Flight Plan

References

(1) Hatfield, Jack J.: Crew Station Technology. Tutorial at 5th Digital Avionics System Conference, p. 88: October 31, 1983.

(2) Reeder, John P.; Schmitz, Robert A.; and Clark, Leonard V.: Operational Benefits From the Terminal Configured Vehicle. NASA TM-80046, 1979.

(3) Steinmetz, George G.; Morello, Samuel A.; Knox, Charles E.; and Person, Lee H., Jr.: A Piloted-Simulation Evaluation of Two Electronic Display Formats for Approach and Landing. NASA TN D-8183, 1976.

(4) Morello, Samuel A.; and Person, Lee H., Jr.: Terminal-Area Flight Experience With the NASA Terminal Configured Vehicle. NASA paper presented at the Flight Safety Foundation 23rd Annual Corporate Aviation Safety Seminar (Washington, D.C.), April 9-12, 1978.

(5) Dieudonne, James E.; Grove, Randall D.; and Steinmetz, George G.: A Simulation Study of Curved, Descending, Decelerating, Landing Approaches for Transport Aircraft. NASA TN D-8190, 1976.

(6) Steinmetz, George G.: A Simulation Investigation of Cockpit Display of Aircraft Traffic During Curved, Descending, Decelerating Approaches. NASA TM-80098, 1979.

(7) Dieudonne, James E.: Description of a Computer Program and Numerical Technique for Developing Linear Perturbation Models From Nonlinear Systems Simulations. NASA TM-78710, 1978.

(8) Morello, Samuel A.; Knox, Charles E.; and Steinmetz, George G.: Flight-Test Evaluation of Two Electronic Display Formats for Approach to Landing Under Instrument Conditions. NASA TP-1085, 1977.

(9) Research Triangle Institute: Development of Simulation Techniques Suitable for the Analysis of Air Traffic Control Situations and Instrumentation. NASA CR-112195, p. 32: December 1972.

(10) Houck, J. A.: A Simulation Study of Crew Performance in Operating an Advanced Transport Aircraft in an Automated Terminal Area Environment. NASA TM 84610, pp. 42-47: May 1983.

(11) Bowmar Instrument Corporation: Operating Instructions, Programmable Keyboard System ES1003-2. June 10, 1982.

(12) Hycom, Incorporated: VGE Programming Manual (rough draft). December 21, 1983.

84-2687

A SIMULATOR APPLICATION OF A "HANDS-ON THROTTLE AND STICK" CONCEPT TO A TRANSPORT PILOT/AUTOPILOT INTERFACE

Anthony M. Busquets*
Russell V. Parrish
Thomas W. Hogge

NASA Langley Research Center
Hampton, Virginia

Abstract

This paper presents the initial experiences garnered in applying a multifunction control strategy, based on the Air Force's "Hands - On -Throttle - And - Stick" (HOTAS) concept of the fighter aircraft world, to a transport aircraft simulator. The multifunction control strategy involves the activation of various flight system/subsystem operations (such as guidance and control, communication, and navigation functions) by use of menu displays and throttle and stick switches. The initial application of this multifunction control (MFC) concept was developed around a pilot/autopilot interface, contrasting a conventional, dedicated autopilot interface to a MFC implementation.

The simulator characteristics and autopilot functions, as well as the conventional interface and MFC hardware/software, which were utilized in the application, are described herein. Initial pilot reaction and suggested improvements to this particular implementation are discussed. The paper terminates with a glance at plans for improvements and future applications based on the outcome of this initial study.

Introduction

The growth in complexity of modern transport aircraft cockpits is paralleled by the increase of specialized equipment necessary to control the vehicle. This induces an increase in the number of dedicated controls/switches needed to provide the human interface to such devices. This effect is most noticeable when one graphs the number of dedicated controls and switches versus time as seen in Figure 1 (Reference 1-2). Even though the number of displays in the cockpit has slowly decreased, as they have become more "integrated", the number of dedicated controls has drastically increased. Inherently associated with this is the increased crew workload.

Almost all segments of the flight management research community are concerned with reducing crew workload within the modern aircraft cockpit. These reductions are being sought in various ways, including the reduction of cockpit clutter through use of multipurpose, integrated displays and consolidated switches/controls which utilize programmable-legend multifunction switches. These concepts are also intended to improve pilot performance and efficiency, particularly during critical operations. The Air Force has been actively pursuing a concept known as "Hands - On - Throttle - And - Stick" (HOTAS) for the high

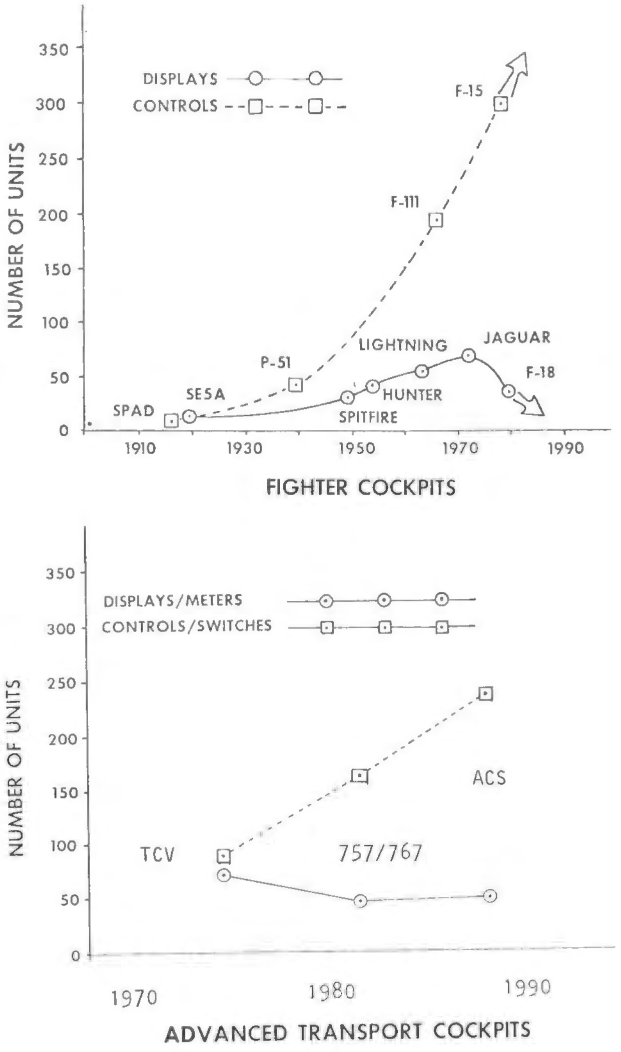


Fig. 1 Trends in controls/display consolidation

workload arena of the fighter cockpit (Reference 3). This concept involves the activation of various flight and weapons systems/subsystems by the pilot without the necessity of releasing the flight controls. This is accomplished by the use of switches and buttons built into the throttles and stick handles and whose functions are computer controlled.

A similar concept for transport aircraft applications was evaluated within the Crew Station Technology program at the NASA Langley Research Center. The idea behind the concept for transport aircraft is not necessarily to enable the pilot to keep his hands on the throttle and stick, but rather to reduce, without increasing task

----
*Member AIAA, IEEE

difficulty, the number of dedicated controls that are crowding the available cockpit space. These goals were to be accomplished by use of a multifunction control (MFC) technique implemented through menu displays and throttle and stick switches. The initial application of this MFC concept was developed around a pilot/autopilot interface, contrasting a conventional, dedicated control/display unit (Automatic Guidance and Control Unit, i.e. AGCU) for an advanced autopilot to a MFC implementation of the same. The MFC implementation allows the pilot to select the various combinations and levels of automatic flight control assistance by activating a cursor - selectable menu choice. Other aircraft systems operations, such as communication and navigation functions, could be controlled with the same throttle and stick switches and different menu displays.

This paper describes both the AGCU and MFC hardware as well as the software functions and simulator characteristics used in the application. It is deemed appropriate to mention here that the intention of this initial effort was to focus mainly on the hardware/software implementation of this concept rather than attend to human factor design considerations. Initial pilot reaction is discussed as well as some of the suggested improvements for this particular application of the MFC concept. Comments on future implementations and plans for more formal simulator evaluations complete the paper.

SIMULATOR CHARACTERISTICS

The simulator characteristics are discussed in terms of the portions common to both implementations of the autopilot interfaces, and then in terms of those portions specific to each interface.

Airplane Mathematical Model

The simulation study centered about the autopilot of an advanced flight-control configured transport airplane, in this case, the NASA Terminal Configured Vehicle (TCV) B-737-100 (Reference 4). This airplane is used to research advances in avionics flight systems and incorporates on-board all-digital flight-control computers, electronic displays, and fly-by-wire control features. The equations used to describe the motions of the simulated airplane were linearized, six degree - of-freedom, rigid-body equations referenced to a body-fixed axis system. The equations were linearized about the landing approach configuration. The autopilot equations, engine models, and other portions of the airplane model were the full nonlinear equations utilized in other Langley Research Center simulations of this particular airplane (References 5 - 8).

Computer Implementation

The mathematical model of the airplane was implemented on a Digital Equipment Corporation (DEC) VAX-11/780 minicomputer in VAX-11 FORTRAN. The electronic displays, an Electronic Attitude Direction Indicator (EADI) or Primary Flight Display (PFD), and an Electronic Horizontal Situation Indicator (EHSI) or Navigation Display (ND), were produced on an Adage 3000 programmable display generator using Ikonas Display Language (IDL).

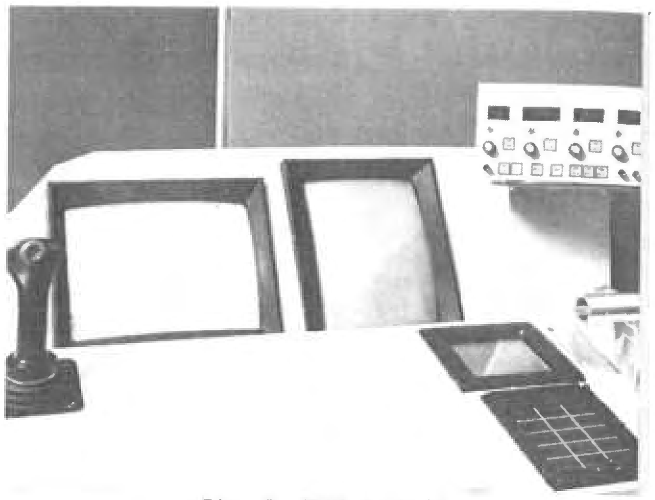


Fig. 2 ADEC console

Simulator Cockpit

Figure 2 shows the fixed-base Advanced Display Evaluation Cockpit (ADEC) console used in this study. The ADEC, as well as the VAX-11/780 minicomputer and Adage 3000 display generators, are part of a Crew Station Systems Research Laboratory (CSSRL, Figure 3) currently under development by the Cockpit Systems Branch of the Flight Control Systems Division of the Langley Research Center. The primary instrumentation for the pilot's side of the cockpit (the co-pilot station was not used in this study) were the electronic displays. The PFD contained altitude, ground speed, pitch, roll, glide slope error, and localizer error indicators, as well as a perspective runway and mode-specific indicators (such as flight path angle indicators, track scale, and mode annunciators). The ND displayed a flight plan map with current airplane position, waypoints, track scale, and various other sophisticated navigational aids (trend vector capabilities, distance and time to next waypoint, etc.).

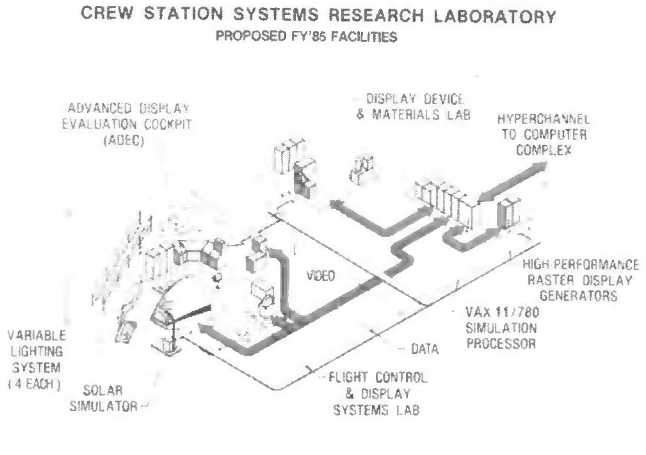

CREW STATION SYSTEMS RESEARCH LABORATORY
PROPOSED FY'85 FACILITIES
ADVANCED DISPLAY EVALUATION COCKPIT (ADEC)
DISPLAY DEVICE & MATERIALS LAB
HYPERCHANNEL TO COMPUTER COMPLEX
VIDEO
HIGH PERFORMANCE RASTER DISPLAY GENERATORS
VAX 11/780 SIMULATION PROCESSOR
DATA
FLIGHT CONTROL & DISPLAY SYSTEMS LAB
VARIABLE LIGHTING SYSTEM (4 EACH)
SOLAR SIMULATOR

Fig. 3 CSSRL under development

Description Of The Autopilot

The autopilot of the NASA TCV was chosen as the instrument of comparison for the interface evaluations, since it is at least representative of futuristic flight-control systems. This autopilot incorporates several modes/submodes of various sophistication, ranging from control wheel steering (CWS) concepts to total four dimensional path automation. This section will describe in some detail the functions of the lower-level automatic modes (those which involve some continual pilot interaction) and superficially describe the higher levels of automation.

Control Wheel Steering. There are two control wheel steering (CWS) modes available with this system, these being the more conventional Attitude CWS and the more advanced Velocity Vector CWS. The purpose of both modes is to provide stabilized airplane responses to pilot control inputs from either the hand controls (or wheel and column) and throttles or from the autopilot interface (regardless of the method of implementation). More importantly, CWS acts to hold certain flight variables constant upon release of the pilot inputs. In the Attitude CWS Mode, the autopilot will maintain the pitch attitude present when the pilot released the pitch control stick and likewise the roll attitude present when the roll command was nulled. This mode does not maintain constant heading, constant vertical velocity, constant altitude, or any other path variable, but only pitch and roll attitudes when no pilot inputs are present (essentially an attitude hold system).

In the Velocity Vector CWS Mode, the hold functions are provided on flight path angle and track angle, again when no pilot inputs are present. In the roll axis of this mode, if the bank angle at time of release is greater than +/- 5 degrees, then that bank angle is held. If bank was less than +/- 5 degrees, then the existing track angle is held (small adjustments in the held track angle can be made with stick inputs that result in banks of less than 5 degrees, however, to provide fine tuning). The roll axis of Velocity CWS is especially compatible with the trend vector displayed on the ND. In the pitch axis of this mode, pilot inputs elicit airplane responses that are very much like those of the Attitude CWS Mode. However, when the force input nulls, the current value of flight path angle is then held. Special symbols on the PFD, representing commanded flight path angle and actual flight path angle, have been provided for operation of this more sophisticated flight mode. Essentially, this mode provides control of the velocity vector of the airplane, or the direction in which the airplane is actually going, rather than where it is pointing.

Calibrated Airspeed Mode. The Calibrated Airspeed Mode allows the pilot to control the autothrottle system of the autopilot. There are two submodes, select and engage. In the select mode (color-coded blue in both the AGCU and MFC implementations), the pilot may input an airspeed for use in the engage mode. In the engage mode (color-coded green in both implementations), the autopilot drives the throttles to capture and maintain the selected airspeed.

Auto Mode. Unless this mode is engaged, none of the submodes to be described hereafter can be activated. It functions merely as a simultaneous on/off switch for these various functions. When it is turned off, the autopilot reverts to Velocity CWS.

Track Angle Mode.- The Track Angle Mode allows the pilot to steer the airplane or, prior to engaging the mode, to preselect a track angle for the autopilot to capture and hold when the mode is engaged.

Flight Path Mode.- The Flight Path Mode allows the pilot to control the ascent/descent angle of the airplane or, prior to engaging this mode, to preselect an angle for capture and hold when the mode is engaged.

Altitude Mode.- The Altitude Mode allows the pilot to control the altitude of the airplane. It has three submodes: preselect (color-coded blue), armed (color-coded amber), and engaged (green). The preselect submode allows entry of a desired altitude. The armed submode refers to the state existing when an altitude has been preselected and only certain logic requirements remain to be satisfied before the mode is coupled to the flight-control system. The airplane is not controlled by this mode when in the preselect or armed submodes, but the system automatically will enter the engaged state when the altitude error becomes smaller than a threshold value based on the airplane's vertical speed. In the engaged state, the autopilot will capture and hold the preselected altitude. Small changes in the held value may be made while engaged, but the mode reverts to the armed state for changes larger than 1200 feet. In the armed submode, one of the necessary logic requirements to advance to the engaged submode is a flight path angle that will allow intercept of the desired altitude. Therefore, the Altitude Mode is utilized in cooperation with the Flight Path Mode.

Other Modes.- The Land Mode provides automatic control of the airplane during ILS/MLS capture and tracking, decrab, flare, and rollout. Horizontal Path, Vertical Path, and Time Path provide automatic two dimensional, three dimensional, and four dimensional flight path control, respectively. All of these modes have armed and engaged states, represented by orange and green color codes, respectively. The submode is essentially armed during capture and transitions to the engaged state when certain error conditions are satisfied.

Conventional Implementation (AGCU)

The AGCU panel depicted in Figure 4 uses seven-segment displays to present values for airspeed, altitude, flight path angle and track angle. Back-lighted pushbutton switches are used for engaging/arming/disengaging the associated modes while rotary knobs are used to select/preselect (input) the numeric values to the autopilot. The AGCU is interfaced to the VAX 11/780's 16-bit parallel interface port through a custom-built discrete-multiplexer/formatter (DMF) input/output system. The rotary knobs and associated coarse/fine switches (which control increment sizes) are tied in through analog channels as a matter of convenience. The DMF unit accepts from the host computer data blocks, as a series of consecutive words, which contain the information necessary to display the numeric data on the display area and turn on/off the appropriate switch mode lamps. The unit then unpacks this

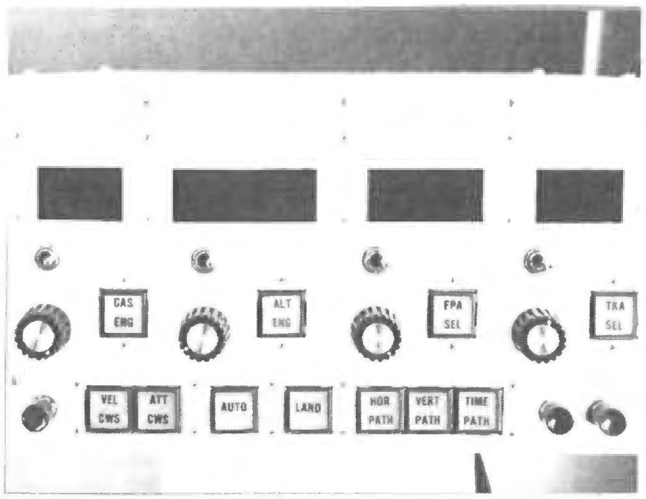


Fig. 4 Engineering model of AGCU panel used in NASA's TCV aircraft.

data, reformats it and routes it to the appropriate display and/or discrete lamp. This I/O system also reads the AGCU panel switch activations, codes it into packed data words and returns a data block to the host computer thus providing two-way communication.

The host computer code that performs the discrete and analog I/O runs as a detached process at a lower priority level than the aircraft model. This scheme allows the the model to run with no interruptions from input/output functions. The data is passed between the two processes in common blocks through global variable sections.

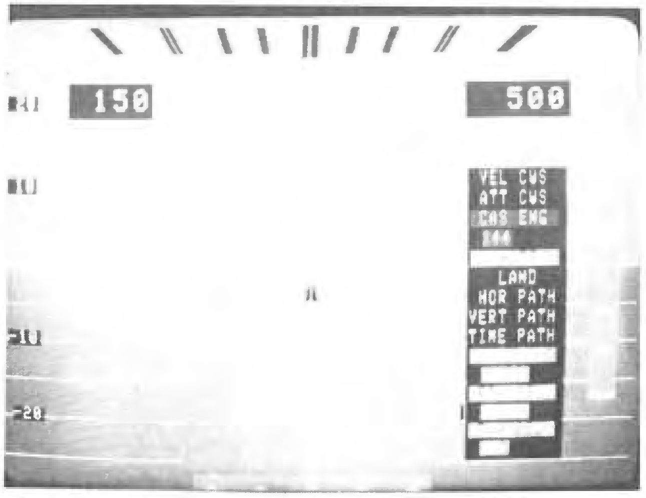


Fig. 5 PFD (EADI) with menu implementation of AGCU functions.

Multifunction Control Implementation

The autopilot is represented in the MFC concept as a cursor-driven menu within the PFD as shown in Figure 5 . All the functions available in the AGCU version are displayed on the PFD within a black rectangle (there is also an option for a transparency mode that does away with the black background and allows one to see through the menu to the parts of the PFD that are "behind" it). The menu functions are accessible via escape and fire buttons, located on the sidearm controller (black and red buttons, respectively), and the throttle switch located within the throttle handle itself (see Figure 6). This switch is a five-position, center-off contact switch which provides a plus or minus, two-level voltage for fine and coarse operations. It was constructed with the contact components from a standard relay thus allowing the switching mechanism to fit within the constraints of the throttle handle. The buttons and throttle switch are interfaced to the host computer via three analog channels. A description of their operation follows :

Throttle Switch. When the cursor is in the mode selection state, activation of this switch will move the cursor, up or down, to the next mode legend. Continual activation (e.g., holding it down) causes continual stepping through the menu. When the cursor is in the numeric input state (cursor resides on a number display), this switch will run the displayed number up or down. The two positions up and two positions down provide for coarse and fine changes.

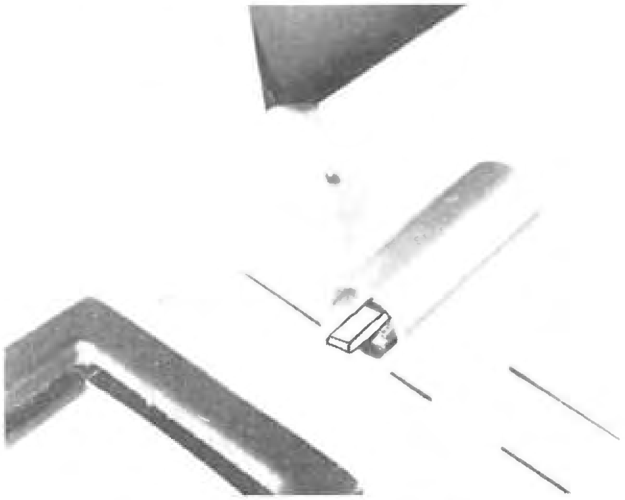


Fig. 6 Throttle switch and handle

Fire Button (Red). When depressed while the cursor is on a mode legend, the mode state is subsequently changed to its next state (select/arm/engage/off). The mode legend background will change color according to the mode status thereby selected. This is the only function of the "fire" button.

Escape Button (Black). This button serves two purposes. The cursor, in the mode selection state, will step through the menu at the command of the throttle switch until a mode legend with an active numeric input state (preselect/select, armed, or engaged) is encountered or selected. At this point, the cursor drops beside the associated numeric display. After "dialing in" a numeric value via the throttle switch, the ESC button will return the cursor to the mode selection function. If the cursor is in the mode selection state, the ESC button will toggle the menu ON or OFF the primary flight display.

Pilot's Evaluation

The MFC implementation of the autopilot interface reduced the number of dedicated controls from the four knobs and eleven switches of the conventional AGCU to two buttons and one switch, which could also be used for other flight system functions. In order to validate the concept of the MFC implementation, a NASA research pilot, who has been involved in advanced transport systems work through numerous NASA programs, was asked to subjectively evaluate the autopilot application. This evaluation revealed several areas for improvements. Before discussing these areas, however, it is necessary to understand how a pilot actually uses the conventional autopilot interface.

Conventional Interface Operations

Velocity CWS, Attitude CWS, and Auto Mode are mutually exclusive modes of operation. The CWS modes are straight forward in operation, being either off or engaged. The more highly automated submodes of Auto (Land, Horizontal Path, etc.) are also straight forward in operation, being either off, armed (during the capture phase), or engaged (an automatic transition from the armed state). However, there is a heirarchy in the path submodes. The Vertical Path submode will not arm (and subsequently engage) unless Horizontal Path is in the engaged state, and the Time Path submode will not arm unless Vertical Path has engaged. The other modes and submodes are a bit more complicated.

The Calibrated Airspeed Mode has two submodes, select and engage. When the mode is off, the display tracks the current airspeed of the airplane. In the select submode, the display increments about a reference airspeed (the existing value at entry to the select state) as the appropriate knob is turned. However, the autothrottle system is not coupled to the display and airspeed is not controlled by the autopilot system. In the engaged submode, the autothrottle is coupled to the display, and the system will capture and hold the displayed airspeed. The pilot may change the displayed value of airspeed with knob inputs and still remain in the engaged state. Thus the pilot may "fly" the airspeed simply by turning the knob while in the engaged state. In actual operation, the pilot relys heavily on this mode, not merely to hold a reference airspeed, but also to make changes in airspeed, because the required throttle position for a desired airspeed is not directly known.

The Track Angle Mode and Flight Path Mode are used in a manner similar to the Calibrated Airspeed Mode. When the modes are off, the displays track the current flight values. The preselect submodes are like the select submode of Airspeed, the difference being merely in name to indicate that these submodes are available to the pilot only when the Auto Mode is off or when one of the higher level Auto submodes is armed or engaged. For example, Track Preselect can be used to preselect a track angle when Auto is off, or, when Auto is on, only if Horizontal Path is armed or engaged. Otherwise, if Auto Mode is on, the Track Angle Mode is in the engaged state. The pilot may "fly" the track angle with knob inputs while in the engaged state. Flight Path Angle Preselect may be used in a similar manner, being available when Auto is on, only if either Altitude Mode or Vertical Path Mode is armed or engaged. In actual operation, the pilot is more likely to revert to a CWS mode to change track angle or flight path angle, and very rarely touches the knobs for these modes.

In contrast to the Track and Flight Path Modes, Altitude Mode is heavily used in piloted operation to capture and hold reference altitudes. A typical altitude change would be flown in the following manner. While in Velocity CWS, the pilot would enter Altitude Preselect by turning the altitude knob and entering the desired altitude. He would then establish the desired flight path angle to reach that altitude with a control stick input, engage the Auto Mode and thus the Flight Path Angle Engage (hold) Mode, and then arm the Altitude Mode. The autopilot would then begin to capture that altitude, transitioning to the Altitude Engage state and Flight Path Angle Mode off state in order to hold the reference altitude. In the Altitude Engage state, the pilot may "fly" altitude changes with the knob unless the changes are larger than 1200 feet. For larger changes, the system drops to the Altitude Armed state.

In actual operation, therefore, the pilot uses the calibrated airspeed and altitude knobs heavily, and also the calibrated airspeed , auto, and altitude buttons. The other knobs and buttons are rarely used.

MFC Implementation Problems

Because the menu was programmed with the Fortran Support Subroutines rather than in the more primitive, but much faster in execution speed, Ikonas Display Language (IDL), considerable delays were encountered when using the throttle switch. Both positioning the cursor and changing flight variables were affected by these delays. Update rates for the menu display barely approached two per second while button and switch inputs were sampled at twenty per second, making accurate operation of the switch very difficult. Several human factors problems were identified in both the layout of the menu and in operational use of the switch and Fire/Escape buttons, also. With the conventional interface unit, the pilot makes a change by glancing away from the primary display, locating the desired knob or button, and pushing or twisting to obtain the desired result. With the MFC version, the pilot must shift his lookpoint to the menu and drive the cursor with the throttle switch to the desired location, possibly through a long list of modes/submodes (with activation of the ESC button at each encounter of an active numeric input state). Once in position, he must, in the worst case, push the fire button, drive the variable to the desired value with the throttle switch, and push either the Fire button or the ESC button. Rather than simplifying the pilot's task, more actions are now required and the task may have become more difficult.

Potential Solutions

The update rate of the menu display can be brought up to the speed of the airplane model, 20 Hertz, by reprogramming the display in IDL. Precise control of the cursor location and accurate control of the variable values should thus be obtained. To simplify the mode selection problem, rather than driving the cursor a single step at a time through the entire menu, the coarse up/down inputs will be programmed to "home" on the heavily-used Calibrated Airspeed Mode and the Altitude Mode, respectively. Some intelligent automation will be incorporated, such that the menu will only display those modes/ submodes that are currently available (Figure 7). This feature will reduce the number of cursor locations from fifteen to seven, and combined with the coarse "HOME" locations, will allow access to most modes/submodes with only a coarse then fine input of the switch (an easily produced combination). The ESC button will be changed to a "number I/O" button, and reprogrammed to toggle the throttle switch between a cursor drive and a change number drive (the cursor will drop to an indented position beside the variable display whenever the throttle switch input will affect the number - when the throttle switch acts as a cursor drive, the cursor will be positioned beside the addressed mode/submode title). The Fire button will continue to act as a mode changer.

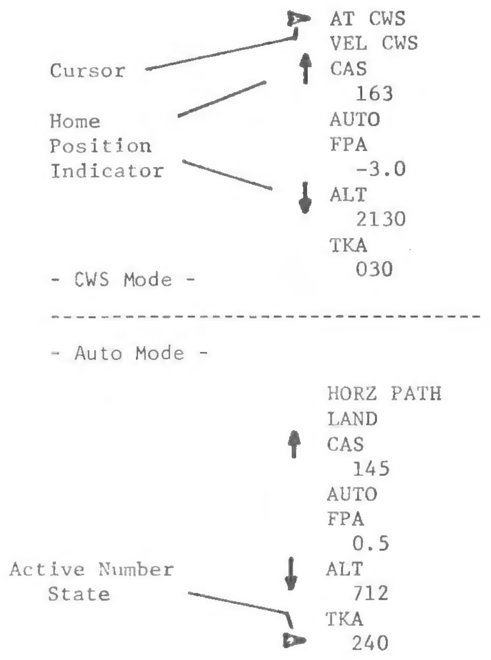


Fig. 7 Automated menu implementations

Several other suggestions were made by the pilot that will also be implemented for evaluation in the revised version of the interface. When the menu is off, the pilot needs a tag on the PFD that would display the current autopilot mode. Also, the perspective runway on the PFD could possibly be occluded by the menu in certain crosswind conditions. Therefore, when the menu display is turned off (which was previously done by the ESC button, but is now to be done with two sucessive coarse up or down cursor commands - commanding moves off of the menu), it may be turned back on in a left-side position by pushing the number I/O toggle (the former ESC button) or in a right-side positon with the Fire button.

The conventional interface device, as well as the current Boeing 757/767 autopilot interface devices, have knob inputs that change the displayed variable values at a rate proportional to the speed at which the pilot twists the knob. A potentiometer-based throttle switch would allow a rate command-type input, changing the number at a rate proportional to switch deflection. The coarse/fine inputs for cursor control would still be provided as discrete steps. Such an implementation will also be explored.

Concluding Remarks

This paper has discussed a research implementation of a multifunction control strategy based on the HOTAS concept. The autopilot interface application revealed several areas for future improvements to this particular application, most of which were related to the method of implementation. Several pilot suggestions to improve ease of use were introduced. Also, more formal evaluations of implementation techniques and other flight system applications are currently being planned, based on the knowledge gained from this evaluation. This study is being done by the Cockpit Systems Branch in accord with its goals, in the area of information management, of performing research on design concepts and systems hardware/software technologies for advanced crew workstations that allow for full management, monitoring and control of subsystems and system-level functions. These concepts and technologies are to be applied to meet both future aircraft and spacecraft workstation requirements.

References

(1) Doetsch, K.H.: "Technical Evaluation Report", AGARD Conf. Proc. No. 312, "The Impact of New Guidance and Control Systems on Military Aircraft Cockpit Design", pp. vii-xv

(2) Morello, Samuel A.; Steinmetz, George G.: "Flight Management of Advanced Systems in the Crew Station", 5th Digital Avionics Systems Conf., November 1983

(3) Moran, John F.: "Integrated Displays and Controls for Military Aircraft", 3rd Digital Avionics Systems Conf., November 1979

(4) Staff of NASA-Langley Research Center: Test Facilities Guide, "Terminal Configured Vehicle Program". NASA SP-435, 1980.

(5) Steinmetz, George G.; Morello, Samuel A.; Knox, Charles E.; and Person, Lee H., Jr.: "A Piloted-Simulation Evaluation of Two Electronic Display Formats for Approach and Landing". NASA TN D-8183, 1976.

(6) Dieudonne, James E.; Grove, Randall D.; and Steinmetz, George G.: "A Simulation Study of Curved, Descending, Decelerating, Landing Approaches for Transport Aircraft". NASA TN D-8190, 1976.

(7) Steinmetz, George G.: "A Simulation Investigation of Cockpit Display of Aircraft Traffic During Curved, Descending, Decelerating Approaches". NASA TM-80098, 1979.

(8) Houck, J. A.: "A Simulation Study of Crew Performance in Operating an Advanced Transport Aircraft in an Automated Terminal Area Environment". NASA TM 84610, pp. 42-47: May 1983.

MICRO-BASED CONTROL AND DISPLAY RESEARCH

1Lt Darleen A. Sobota

1Lt Gretchen D. Lizza

Dr. John M. Reising

Flight Dynamics Laboratory

Wright Patterson AFB, Ohio

Abstract

This paper discusses a distributed, microcomputer-based research facility which provides the ability to conduct cockpit display research in a very cost effective manner. Two examples of studies are discussed, each using different capability levels of the micro-computer system. The final conclusion is that the design of a microcomputer-based network has provided an inexpensive and flexible solution to the rising costs of high fidelity simulations, when the objective of the simulation is the prescreening of new technologies. In the future, for many types of simulation, micros may totally replace the mainframe computers of today.

Introduction

In the design of advanced controls and displays for cockpits, simulators have provided an avenue for researchers to investigate technologies early in the developmental process. Simulation reduces the cost and risks associated with in-flight evaluation of new concepts. The value of simulator-based research has been borne out by the successful transition of various devices, such as electro-optical displays, into several current aircraft. Electro-optical displays are but one example of the many new technologies entering the cockpit (1); each of these technologies, however, has to be evaluated so that the crewmembers can take fullest advantage of the system's capabilities.

This need for proper evaluation dictates that many levels of simulation take place in order to foster the flow of technology into the cockpit. A large range of simulation facilities exists today, allowing researchers the opportunity to really "ring-out" proposed concepts and hardware for future applications. If we limit the scope of this discussion to that simulation capability which concentrates on the dynamic nature of controls and displays during a flying task, fixed-based simulators are representative of the low end of the spectrum. These types of simulators have been used for research and training since WW II. The Link trainer, introduced in the same era, allowed pilots to train in a more realistic environment than the static simulators because of its three degree of freedom motionbase. In the seventies, six degrees of freedom became possible. Even though this new capability added additional realism to simulation, problems and controversy began to surface with respect to the coordination of the simulator's movements with the visual cuing system and proprioceptive thresholds (2). The gantry/model board systems which present an out-the-window-view made giant leaps towards resolving these problems. Today, with motion-based, multi-channel Computer Generated Imagery (CGI), and increased computer power, simulators provide a very realistic environment in which to conduct research.

However, because of rapid technology advances, it became necessary to replace early digital computers with second, and then third generation mainframes in order to enhance power, fields of view, and resolution. Even though the advancements in this area have been exciting from the researchers point of view, the economics of simulation are disturbing. In short, sophistication has meant spiraling costs. In light of these costs, it is important to seriously consider the level of simulation needed for a particular research area. For basic concept evaluation, the researcher would be well advised to consider the feasibility of a fixed-based, dynamic cockpit.

Dynamic Cockpits

There are two basic levels to the dynamic cockpit. The less sophisticated version utilizes slide projectors and ground glass screens to emulate electro-optical displays such as cathode ray tubes (CRT's) (3). In the past, these wooden shell cockpits were not able to provide the dynamic display capability needed to evaluate concepts during a flying task. The advent of the microprocessor has given a whole new meaning to the "dynamic" in these dynamic cockpits, creating a second type of dynamic cockpit with a higher level of fidelity (4). This second type uses actual CRT's but is still a step lower in fidelity than the mainframe, fixed-based simulator previously described. The shell of the cockpit is wood, but the control and display panel are no longer ground glass surfaces. With a micro-network, graphic generators and CRT's, the dynamic cockpit can provide real time displays, e.g., a head up display, as well as various situation awareness and status displays. Now, this is not to imply that all levels of research can be conducted in dynamic