COPY 2

U.S. Department of Transportation
**Federal Aviation Administration**

DOT/FAA/CT-88/10

# DIGITAL SYSTEMS VALIDATION HANDBOOK-VOLUME II

**FEBRUARY 1989**

FEDERAL AVIATION ADMINISTRATION
TECHNICAL CENTER

FAA TECHNICAL CENTER LIBRARY
00013515

NOTICE

This document is disseminated under the sponsorship of the U.S. Department of Transportation in the interest of information exchange. The United States Government assumes no liability for the contents or use thereof.

The United States Government does not endorse products or manufacturers. Trade or manufacturers' names appear herein solely because they are considered essential to the objective of this report.

**Technical Report Documentation Page**

| 1. Report No. | 2. Government Accession No. | 3. Recipient's Catalog No. |
|---|---|---|
| DOT/FAA/CT-88/10 | | |

| 4. Title and Subtitle | 5. Report Date |
|---|---|
| DIGITAL SYSTEMS VALIDATION HANDBOOK - VOLUME II | FEBRUARY 1989 |
| | 6. Performing Organization Code: CRMI |
| 7. Author(s): SEE TABLE OF CONTENTS | 8. Performing Organization Report No.: DOT/FAA/CT-88/10 |
| 9. Performing Organization Name and Address: COMPUTER RESOURCE MANAGEMENT INCORPORATED, 950 HERNDON PARKWAY SUITE 360, HERNDON, VA 22070 | 10. Work Unit No. (TRAIS) |
| | 11. Contract or Grant No.: DTFA03-86-C-00042 |
| 12. Sponsoring Agency Name and Address: U.S. DEPARTMENT OF TRANSPORTATION, FEDERAL AVIATION ADMINISTRATION TECHNICAL CENTER, ATLANTIC CITY INTERNATIONAL AIRPORT, NJ 08405 | 13. Type of Report and Period Covered: HANDBOOK |
| | 14. Sponsoring Agency Code: ACD-230 |

15. Supplementary Notes

Point of Contact: Pete Saraceni, ACD-230
FAA Technical Center
Atlantic City International Airport, NJ 08405

16. Abstract

Volume II covers detailed technical topics such as latent faults; data buses; integrated assurance assessment; analytical sensor redundancy; and protection against lightning, electromagnetic interference, and high energy radio frequency fields. These topics are covered in detail to familiarize the certification engineer with the issues involved in implementing the new technologies.

Volume II covers topics that will enable the certification engineer to understand the information presented in type certification and supplemental type certification documentation, to understand variations in the implementation of technologies, and to discuss them with the design engineer.

Volume II also addresses some of the soon-to-be-available technologies in the "Advanced Validation Issues" chapter. The direction of aviation research in the United States is discussed along with challenges and problems that confront the certification engineer in certifying the new technologies.

Since the topics discussed in this Handbook are at the forefront of technological research, some of the concepts presented are subject to discussion by experts in the field. In these areas, the Handbook presents various viewpoints alerting the certification engineer to the various views so that this information will be considered in formulating decisions and developing certification criteria.

| 17. Key Words | 18. Distribution Statement |
|---|---|
| Avionics, Digital, Validation, Certification, Reliability, Redundancy, Latent Faults, Fault Insertion, Data Buses, Lightning, Radio Frequency Fields, Transients, Verification, Composites, Modeling | This document is available to the U.S. public through the National Technical Information Service, Springfield, Virginia 22161 |

| 19. Security Classif. (of this report) | 20. Security Classif. (of this page) | 21. No. of Pages | 22. Price |
|---|---|---|---|
| Unclassified | Unclassified | 900 | |

U.S. Department of Transportation
**Federal Aviation Administration**

Technical Center

Atlantic City Int'l Airport
New Jersey 08405

May 4, 1992

Dear Colleague,

Enclosed is additional material for the Digital Systems Validation Handbook - Vplume II (DOT/FAA/CT-88/10). These additions consist of a copy of Chapter 17 - Software Quality Metrics, and a revised Table of Contents, List of Authors, Glossary, and Acronyms and Abbreviations. Please add these materials to your Handbook.

If you have any questions or comments concerning the material, or if you would like to receive a copy of the Handbook, please feel free to contact me at the Technical Center.

Sincerely,

Pete Saraceni

Pete Saraceni
FAA Technical Center
ACD-230, Building 201
Atlantic City International Airport, NJ 08405

Phone: (FTS) 482-5577
(609) 484-5577

Enclosures: Chapter 17 - Software Quality Metrics
Table of Contents (revised)
List of Authors (revised)
Glossary (revised)
Acronyms and Abbreviations (revised)

DIGITAL SYSTEMS VALIDATION HANDBOOK - VOLUME II

TABLE OF CONTENTS


| Chapter | | Page |
|---|---|---|
| LIST OF AUTHORS | | vii |
| 1. | SUMMARY<br>R. L. McDowall | 1-1 |
| 2. | INTRODUCTION<br>R. L. McDowall | 2-1 |
| 3. | INTEGRATED ASSURANCE ASSESSMENT<br>Hardy P. Curd | 3-1 |
| 4. | QUADRUPLEX DIGITAL FLIGHT CONTROL SYSTEMS<br>Lloyd N. Popish | 4-1 |
| 5. | ADVANCED FAULT INSERTION AND SIMULATION METHODS<br>William W. Cooley | 5-1 |
| 6. | DIGITAL DATA BUSES FOR AVIATION APPLICATIONS<br>Donald Eldredge and Susan Mangold | 6-1 |
| 7. | ANALYTICAL SENSOR REDUNDANCY<br>William W. Cooley and Deborah L. Shortess | 7-1 |
| 8. | ESTIMATION AND MODELING FOR REAL-TIME SOFTWARE RELIABILITY MODELS<br>Donald Eldredge and Susan Mangold | 8-1 |
| 9. | FAULT TOLERANT SOFTWARE<br>Myron J. Hecht | 9-1 |

TABLE OF CONTENTS
(Continued)


| Chapter | | Page |
|---|---|---|
| 10. | LATENT FAULTS | 10–1 |
| | John G. McGough | |
| 11. | AIRCRAFT ELECTROMAGNETIC COMPATIBILITY (Guidelines to Assess EMC Designs) | 11–1 |
| | Clifton A. Clarke and William E. Larsen | |
| 12. | FAST RISE–TIME ELECTRICAL TRANSIENTS IN AIRCRAFT | 12–1 |
| | Roger McConnell | |
| 13. | LIGHTNING STUDIES | 13–1 |
| | William W. Cooley, Barbara G. Melander, and Deborah L. Shortess | |
| 14. | HIGH ENERGY RADIO FREQUENCY FIELDS (Impact on Digital Systems) | 14–1 |
| | John E. Reed and Robert E. Evans | |
| 15. | ELECTROMECHANICAL ACTUATOR SYSTEMS (Electrical Systems Certification Issues) | 15–1 |
| | William W. Cooley | |
| 16. | ADVANCED VALIDATION ISSUES | 16–1 |
| | Hardy P. Curd | |
| 17. | SOFTWARE QUALITY METRICS | 17–1 |
| | Donald A. Elwell and Nancy J. VanSuetendael | |
| 18. | AVIONIC DATA BUS INTEGRATION TECHNOLOGY | 18–1 |
| | Donald A. Elwell, Lee H. Harrison, John H. Hensyl, and Nancy J. VanSuetendael | |
| 19. | PILOT–VEHICLE INTERFACE | 19–1 |
| | Lee H. Harrison, Joan L. Janowitz, and Maurice A. Castronuovo | |

TABLE OF CONTENTS
(Continued)


| Chapter | | Page |
|---|---|---|
| 20. | ARTIFICIAL INTELLIGENCE WITH APPLICATIONS FOR AIRCRAFT | 20-1 |
| | Lee H. Harrison, Pamela J. Saunders, and Joan L. Janowitz | |
| GLOSSARY | | |
| ACRONYMS | | |

LIST OF AUTHORS

Maurice A. Castronuovo
Galaxy Scientific Corp.
2500 English Creek Ave., Bldg. 11
Pleasantville, NJ 08232

Clifton A. Clarke
Boeing Commercial Airplane Company
P. O. Box 3707
Seattle, WA 98124

William W. Cooley
Science & Engineering Associates, Inc.
701 Dexter Avenue
Seattle, WA 98109

Hardy P. Curd
Computer Resource Management, Inc.
950 Herndon Parkway, Suite 360
Herndon, VA 22070

Donald Eldredge
Battelle Columbus Division
505 King Avenue
Columbus, OH 43201

Donald A. Elwell
CTA, Inc.
2500 English Creek Ave., Suite 1000
Pleasantville, NJ 08232

Robert E. Evans
FAA Technical Center
Atlantic City International Airport, NJ 08405

Lee H. Harrison
Galaxy Scientific Corp.
2500 English Creek Ave., Bldg. 11
Pleasantville, NJ 08232

Myron J. Hecht
SoHaR, Inc.
8500 Wilshire Boulevard
Beverly Hills, CA 90211

LIST OF AUTHORS
(Continued)

John H. Hensyl
Diversified International Sciences Corp.
9901-R Business Parkway
Lanham, MD 20706-1840

Joan L. Janowitz
Galaxy Scientific Corp.
2500 English Creek Ave., Bldg. 11
Pleasantville, NJ 08232

William E. Larsen
FAA Field Office
P. O. Box 25
Moffett Field, CA 94035

Susan Mangold
Battelle Columbus Division
505 King Avenue
Columbus, OH 43201

Roger McConnell
CK Consultants
5473 A Clouds Rest
Mariposa, CA 95338

R. L. McDowall
Galaxy Scientific Corp.
2500 English Creek Ave., Bldg. 11
Pleasantville, NJ 08232

John G. McGough, Consultant
150 Walnut Street
Ridgewood, NJ 07450

Barbara G. Melander
Science & Engineering Associates, Inc.
701 Dexter Avenue
Seattle, WA 98109

Lloyd N. Popish, Consultant
525 Davenport Court
Sunnyvale, CA 94087

John E. Reed
AERS/Midwest, Inc.
PO Box 965
106 Washington
Burlington, IA 52601

LIST OF AUTHORS
(Continued)

Pamela J. Saunders
Galaxy Scientific Corp.
2500 English Creek Ave., Bldg. 11
Pleasantville, NJ 08232

Deborah L. Shortess
Science & Engineering Associates, Inc.
701 Dexter Avenue
Seattle, WA 98109

Nancy J. VanSuetendael
Computer Resource Management, Inc.
950 Herndon Parkway, Suite 360
Herndon, VA 22070

# GLOSSARY

α-FAULT. (10) A fault activated by the baseline program (see ß-FAULT).

A PRIORI. (20) Pertaining to deductive reasoning from assumed axioms or supposedly self-evident principles, without reference to experience.

A-SPECIFICATION. (9) The highest level specification typically produced by the contracting organization to define a system (see MIL-STD-1521).

ABSORPTION LOSS. (11) Attenuation or retention of electromagnetic energy passing through a material, a shield. Absorption loss and reflection loss contribute to total shielding effectiveness (SE).

ACCESS. (18) The process of a transmitting bus user obtaining control of a data bus in order to transmit a message.

ACTION INTEGRAL. (13) The action integral is a critical factor in the production of damage. It relates to the energy deposited or absorbed in a system. This energy cannot be defined without knowing the resistance of the system. The instantaneous power dissipated in a resistor is $I^2R$ and is expressed in watts. For the total energy expended, the power must be integrated over time to get the total joules, watt-seconds. By specifying the integral of $i(t)^2$ over the time interval involved, a useful quantity is defined for application to any resistance value. In the case of lightning, this quantity is defined as the action integral and is specified as $i(t)^2dt$ over the time the current flows.

ACTIVE FAULT. (10) A fault that can produce an error (for some input) while executing the current program.

ACTUAL TRANSIENT LEVEL. (13) The actual transient level is the level of transients which actually appear at the system interfaces as a result of the external environment. This level may be less than or equal to the transient control level but should not be greater.

ADAPTIVE FUZZY SYSTEMS. (20) Fuzzy controllers with the ability to learn and explain their reasoning.

ADDRESSING CAPACITY. (6) The number of components addressable by the protocol used on a given data bus.

ADVISORY CIRCULAR. (18) An external FAA publication consisting of nonregulatory material of a policy, guidance, and informational nature.

AIR TRANSPORT AIRCRAFT. (18) Aircraft used in interstate, overseas, or foreign air transportation.

AIRCRAFT LIGHTNING INTERACTION. (13) An encounter with lightning that produces sufficient current within or voltages along an aircraft skin or structure to pose a threat to the aircraft electrical/electronic systems, as a result of a direct lightning attachment.

AIRWORTHINESS STANDARDS. (18) Parts 23, 25, 27, 29, and 33 of the Code of Federal Regulations, Title 14, Chapter 1, Subchapter C.

ALGORITHM. (20) A set of well-defined rules for the solution of a problem in a finite number of steps.

AMBIENT. (16) The substance which absorbs heat from the heat sink.

ANALOG. (20) Pertaining to devices, data, circuits, or systems that operate with variables that are presented by continuously measured voltages or other quantities.

ANALYTICAL REDUNDANCY. (7) The use of software algorithms which use known mathematical relationships between different sensors for sensor failure detection and replace most of additional redundant sensor hardware.

ANALYTICAL ROOT SOLUTION. (4) Information obtained from the roots of the characteristic equations of the airplane model such as short-period or phugoid frequency response.

ANGLE OF ATTACK. (4) Angle between the longitudinal axis of an aircraft and the direction of movement.

ANISOTROPIC. (19) Having different, direction dependent, physical characteristics.

ANODIZE. (11) A preparation by electrolytic process that deposits a protective oxide, insulating film on a metallic surface (aluminum). The oxide defeats electrical bonding. Alodine and iridite finishes on aluminum are conductive.

ANTAGONISTIC QUALITY FACTORS. (17) Quality Factors with conflicting attributes.

APERTURE. (11) An opening, such as a nonconductive panel joint, slot, or crack, allowing electromagnetic energy to pass through a shield.

ARC. (20) The part of a semantic network that connects one object to another.

ARCHITECTURE. (18) The design and interaction of components of a computer system.

ARTIFICIAL INTELLIGENCE. (16) The characteristics of a machine programmed to imitate human intelligence functions.

ARTIFICIAL INTELLIGENCE. (20) The subfield of computer science concerned with emulating human intelligence by use of software and hardware techniques.

ASSISTANT. (20) An Expert System that performs a technically limited subset of an expert's task.

ASSOCIATIVE MEMORY NETWORK. (20) A network where different input patterns, if sufficiently similar, become associated with one another (trigger the same response).

ASSURANCE ASSESSMENT. (4) Procedures whose purpose is to ensure that a proposed system functions according to design specifications.

ASYNCHRONOUS. (20) Operating at a speed determined by the circuit functions rather than by timing signals.

ASYNCHRONOUS MESSAGES. (6) Electronic signals with transmission times that are not known a priori. These may include priority signals requiring immediate access to the bus.

ATTACHMENT POINT. (13) A point of contact of the lightning flash with the aircraft.

ATTENSON. (19) A unique sound used on the flight deck for alerting the flightcrew of warning, caution, and advisory conditions.

ATTRACTOR. (20) The geometrical pattern toward which the trajectory of a dynamical system, represented by a curve in phase space, converges in the course of time.

AUDIO FREQUENCY (AF). (11) The spectrum (20 to 20,000 Hz) of human hearing, often defined as extending from approximately 20 Hz to 50 kHz and sometimes to 150 kHz. Audio noise is nuisance hum, static, or tones from power line 400 Hz, switching regulator and digital clock harmonics, or HF, VHF transmitter frequencies.

AUTOFEATHER. (16) To automatically and swiftly feather the propeller when the engine fails to drive it.

AUXILIARY PROGRAMS. (10) Software executed occasionally.

AVALANCHING LATENT FAULTS. (10) The successive activation of latent faults.

AVIONIC. (18) Electronic equipment used in aircraft.

AXON. (20) The pathway that carries the output of a biological neuron.

ß-FAULT. (10) A fault not activated by the baseline program (see $\alpha$-FAULT).

BABBLING TRANSMITTER. (18) A bus user that transmits outside its allocated time.

BACK PROPAGATION. (20) An algorithm used to adjust weights connecting neurons in successive layers of multi-layer networks. Back propagation learns by example and repetition.

BACKSHELL. (11) Metal shell connecting circuit shields or overbraid to an electrical connector.

BACKWARD CHAINING. (20) A method of problem solving that works backward from a known conclusion in an attempt to find a path of reasoning in order to justify the conclusion.

BACKWARD RECOVERY. (9) Restoration of the system to some previous known correct state and restarting the computation from that point.

BALANCED CIRCUIT. (11) A signal, acting line-to-line, between two conductors having symmetrical voltages identical and equal in relation to other circuits and to ground. "Differential mode" is line-to-line; "common mode" is line to ground.

BALANCED CONFIGURATION. (18) A bus using the HDLC protocol that connects only primary stations.

BANDWIDTH (BW). (11) Frequencies bounded by an upper and lower limit in a given band associated with electronic devices, filters, and receivers.

BASELINE PROGRAM. (10) A set of continuously executed software modules.

BASIN OF ATTRACTION. (20) The collection of all possible initial conditions of a dynamical system for which the trajectories representing that system in phase space will converge to a particular attractor.

BENIGN FAULT. (10) A fault that cannot produce an error while executing the current program, regardless of input, but may produce an error for some other program.

BIDIRECTIONAL DATA BUS. (18) A data bus with more than one user capable of transmitting.

BINARY. (20) Composed of or characterized by two parts or elements.

BINARY SEARCH. (17) A searching algorithm in which the search population is repeatedly divided into two equal or nearly equal sections.

BIREFRINGENCE. (19) The splitting of light into two components, where each component travels at a different velocity.

BIT-ORIENTED PROTOCOL. (18) A communication protocol where message frames can vary in length, with single bit resolution.

BIT TIME. (6) The time it would take to transmit one bit. Usually this is "blank" time when nothing is being transmitted. One nth of the bus speed (i.e., on a 1 kHz bus, the bit time is $10^{-3}$ seconds).

BITS. (17) Binary digits.

BLACK BOX. (20) A component having known input and output, that can be readily inserted into or removed from a specific place in a larger system without knowledge of the component's detailed internal structure.

BLACKBOARD. (20) The controlling agent in a blackboard system that tracks activity between the knowledge sources.

BLACKBOARD SYSTEM. (20) An Expert System that uses multiple knowledge sources that share data.

BLOCK TRANSFER. (6) A data transfer mode allowing the transfer of variable length data blocks.

BOND, ELECTRICAL. (11) Electrical connection at two metallic surfaces securely joined to assure good conductivity often 2.5-mΩ maximum for electrical/electronic units and 1Ω for electrostatic dissipation or safety. A "faying surface" bond maintains contact between relatively large or long surfaces. Inherently bonded parts are permanently assembled and conductivity exists without special preparation: such as with welding, brazing.

BOOLEAN LOGIC. (20) The basic mathematics needed to study the logic design of digital systems. Boolean logic consists of two values and three basic operators.

BRAID, OVERBRAID. (11) Fine metallic conductors woven to form a flexible conduit or cableway and installed around insulated wires to provide protection against electric fields and radio frequencies. Best when peripherally connected to backshells. A grounding strap/jumper may be made of braid.

BREADTH FIRST. (20) The method of searching for solutions by considering all possible subgoals on one level before proceeding to the next level.

BRIDGE. (18) A BIU that is connected to more than one bus for the purpose of transferring bus messages from one bus to another, where all the buses follow the same protocol.

BROADBAND. (12) A frequency spectrum which is wide compared to the bandwidth of the device used to detect it.

BROADCAST. (4) Transmission of messages to all terminals without reference to the identification of the receiving station or terminal.

BROADCAST CAPABILITY. (6) The capacity to transmit messages to all terminals simultaneously.

BROADCAST DATA BUS. (18) A data bus where all messages are transmitted to all bus users.

BUFFER. (18) Memory used to hold segments of the data transferred between asynchronous processes.

BUS. (18) A conductor that serves as a common connection of a signal to multiple users.

BUS CONTROLLER. (18) The electronic unit that is designed to control the bus communication of all users for a centrally controlled bus.

BUS INTERFACE UNIT. (18) The electronics that interface the host CPU of an LRU to a bus medium.

BUS MESSAGE. (18) A complete set of bits that can be transferred between two bus users.

BUS NETWORK. (18) The collection of all BIUs and bus media associated with one bus.

BUS OVERLOAD. (18) The condition that exists when the time it takes to transmit outstanding messages on a bus exceeds the time allotted for those transmissions.

BUS USER. (18) Any LRU attached to a bus.

BYZANTINE RESILIENCE. (5) A fault tolerant process which is tolerant of intermittent faults that can send good information part of the time.

CABLE OR HARNESS. (11) A bundle of separate, insulated, electrical circuits, shielded or unshielded, usually long and flexible and having breakouts, terminations, overbraid, and mounting provisions completely assembled.

CABLEWAY. (11) A solid metallic housing (liner, foil, coating) surrounding and shielding insulated electrical conductors. Also called conduit, tray, or raceway. Crosswise or transverse openings or breaks in the metallic cableway cause noise voltages to be transferred to internal wire circuits.

CANARD. (16) A tail-first aerodyne, usually with auxiliary horizontal surface at the front and a vertical surface at the back.

CAT IIIa LANDING. (6) One of several landing categories defined in FAR 91. CAT IIIa implies the need for an instrument landing approach.

CENTRAL BUS CONTROL. (18) The bus control approach where a single electronic unit attached to a bus controls all the communication of the bus users.

CENTRAL CONTROL. (6) Control from one master, whether stationary or non-stationary.

CERTAINTY FACTOR. (20) A value that represents the level of belief associated with a fact or a rule.

CERTIFICATION. (18,20) The process of obtaining FAA approval for the design, manufacture, and/or sale of aircraft and associated systems, subsystems, and parts.

CHARACTER-ORIENTED PROTOCOL. (18) A communication protocol where messages can vary in length, with single character resolution.

CHARGE TRANSFER. (13) The integral of the current over its entire duration, i(t)dt, in coulombs.

CHECKSUM. (18) An error detection code produced by performing a binary addition, without carry, of all the words in a message.

CHOLESTERIC. (19) A liquid crystal phase with molecules parallel to each other but twisting slightly from layer to layer.

CHORD. (4) The straight line segment intersecting or touching an airfoil profile at two points.

CLOSED-LOOP. (18) A system where the output is a function of the input and the system's previous output.

CLOSED-LOOP. (20) A family of automatic control units linked together with a process to form an endless chain; the effects of control action are constantly measured so that if the controlled quantity departs from the norm, the control units act to bring it back.

CLUSTERING. (20) A technique to group similar objects together under the same label.

CODE. (17) The subset of software which exists for the sole purpose of being loaded into a computer to control it.

CODIFYING. (20) A method of representing knowledge or data so they can be stored in a knowledge base or database for access by a computer.

COLLEAGUE. (20) An Expert System that performs a significant subset of an expert's task.

COMBINER. (19) A screen designed to reflect selected wavelengths of light while remaining transmissive for others.

COMMAND/RESPONSE. (6) "Operation of a data bus system such that remote terminals receive and transmit data only when commanded to do so by the controller." (MIL-STD-1553 Designer's Guide, 1983, p. II-3.)

COMMAND/RESPONSE DATA BUS. (18) A data bus whose protocol initiates each data transfer with a command and terminates the transfer after a proper response is received.

COMMON MODE IMPEDANCE. (11) Impedance or resistance shared by two or more circuits so that noise voltages/currents generated by one are impressed on the others.

COMMON MODE REJECTION. (11) The ability of wiring or an electronic device to reject common mode (line-to-ground) signals and maintain fidelity of differential mode (line-to-line) signals.

COMMON MODE SIGNAL. (11) Identical and equal signals on input conductors or at the terminals of a device relative to ground.

COMPLEMENTARY QUALITY FACTORS. (17) Quality Factors with interrelated attributes.

COMPONENT DAMAGE. (13) Condition arising when the electrical characteristics of a circuit component are permanently altered beyond its specifications.

CONDUCTED EMISSION (CE) OR INTERFERENCE. (11) Voltage/current noise signals entering or leaving a unit on interface conductors. Emission is the general term, interference is undesired noise.

CONFIGURATION MANAGEMENT. (18) The precise control and documentation of the configuration of an entity at any time during its development and deployment.

CONNECTION WEIGHT. (20) A method of representing the strength of the connection between nodes.

CONTENT ADDRESSING. (6) The system of identifying message recipients based on information embedded in the message. This is in contrast to destination terminal addresses.

CONTENTION PROTOCOL. (18) A protocol that allows users to randomly access the bus at any time. When bus contention results, each user tries again to access the bus without contention.

CONTROL LAW. (7) The physical relationship between various sensors and control surfaces.

CONTROL REGISTER. (18) A register in an IC controller that receives commands from a host processor.

CONTROL STRUCTURES. (17) Programming constructs which direct the flow of control.

CONVENTIONAL PROGRAMMING. (20) The use of standard programming languages, as opposed to application development languages, financial planning languages, query languages, and report programs.

CONVERGENCE. (20) The point when the outputs of a Neural Network are no longer changing.

CORONA. (13) A luminous discharge that occurs as a result of an electrical potential difference between the aircraft and the surrounding atmosphere.

COUPLING. (11) The transfer of energy between wires or components of a circuit electrostatically, electromagnetically, or directly.

COVERAGE. (5) The conditional probability of the system successfully recovering from a component fault and continuing to perform the intended functions correctly, given the presence of the fault. Coverage is the measure of

effectiveness of a system's utilization of redundant hardware. Coverage can be qualified and applied to many different components of a system and phases of recovery process. Examples include, fault detection coverage, fault isolation coverage, latent fault coverage, sensor failure coverage, and memory failure coverage.

COVERAGE. (7) The percent confidence level of a given analytical redundancy fault detection and isolation algorithm for all types of faults.

COVERAGE. (9) The probability that when a fault occurs, it will be detected and recovery from the fault will be successful.

CRISP SET. (20) The membership concept used in traditional system modelling, where objects are either in or out of the set with no intermediate step.

CRITICAL. (13) Functions whose failure would contribute to or cause a failure condition which would prevent the continued safe flight and landing of the aircraft.

CROSS COUPLING (CROSSTALK). (11) Transfer of signals from one channel, circuit, or conductor to another as an undesired or nuisance signal or the resulting noise.

DAMAGE. (11) The irreversible failure of a component.

DATA BUS. (6,18,19) A system for transferring data between discrete pieces of equipment in the same complex.

DATA BUS PROTOCOL. (18) The set of rules that governs the transfer of data between data bus users.

DATA DRIVEN. (20) Describing the execution of a program in a data flow system, in which an instruction is carried out whenever all its input values are present.

DATA LATENCY. (6,18) The delay from the time when a piece of information becomes available at a source terminal to the time it is received at the destination.

DATA LINK ASSURANCE OF RECEIPT. (6) The guarantee of good data through the data link level.

DATA REASONABLENESS CHECK. (18) A check performed to see if a value of data is within reasonable bounds for the given context.

dBμV. (12) Decibels referred to one microvolt. Zero db represents one microvolt.

DECIBEL (dB). (11,12) Decibel expresses the ratio between two amounts of power, P1 and P2, at two separate points in a circuit. By definition, the number of dB = 10 log to the base 10 of (P1/P2). For special cases, when a standard power level P2 = 1 mW or 1 W or 1 kW, then the ratio is defined as "dBm," "dBw," or "dBKW." Because $P = V^2/R$ and also $I^2R$, decibels express voltage and current ratios. Ideally, the voltages and currents are measured at two points having

identical impedances. By definition, dB = 20 log V1/V2 and dB = log I1/I2. For convenience, V2 or I2 are often chosen as 1 $\mu$V or 1 $\mu$A and the ratio is defined as dB above a $\mu$V or dB above a $\mu$A when graphing emission or susceptibility limits.

DECLARATIVE. (20) Generally, the knowledge found in the knowledge base; independent knowledge that does not contain procedures or methods of handling other knowledge.

DECOUPLED MANEUVERS. (4) Changes in an aircraft's direction and attitude in one axis without affecting direction or attitude in other axes.

DEDUCTIVE. (20) Learning by logical inference or inferring from available knowledge.

DEFAULT DATA. (18) An alternative value used for a parameter whenever the normal data is not supplied.

DEFUZZIFICATION. (20) The process of translating fuzzy, or non-crisp values into crisp, clearly defined values.

DEMON. (20) A procedure activated by changing or accessing values in a database.

DENDRITE. (20) The pathway that carries the input of a biological neuron.

DEPTH FIRST. (20) A method of searching for a solution by pursuing each branch of the search tree to the end before considering another branch.

DESIGN ERROR. (4) A functional flaw resulting from a misinterpretation of the specifications of the system.

DESIGN MARGIN. (13) The difference between the equipment transient design levels and the transient control level.

DETERMINISTIC. (6) A system where all parameters are known, as opposed to a statistical system where the outcome is subject to the laws of probability.

DETERMINISTIC PROTOCOL. (18) A protocol where all parameters are known so that its various states are predictable in sequence and time.

DIAGNOSTIC FILTER. (7) An analytical algorithm which processes data from N functionally related sensors. The data are used to estimate some sensor outputs and assess the correct functioning of the sensors.

DICHROIC. (19) Having different light absorption characteristics based on incident polarization direction.

DIELECTRIC. (19) Having an insulating property with respect to an electrical field.

DIELECTRIC STRENGTH. (11) Voltage withstand capability that an insulating material sustains before destructive arcing and current flow, usually expressed

in volts per mil thickness. Dielectric withstand voltage is the voltage level at which insulation breakdown occurs.

DIFFERENTIAL MODE (DM) SIGNAL. (11) The signal in a two-wire circuit measured from line-to-line.

DIGITAL DATA BUS. (18) A data bus that uses digital electronic signals.

DIRECT EFFECTS. (13) Any physical damage to the aircraft or onboard systems due to the direct attachment of the lightning channel. This includes tearing, bending, burning, vaporization, or blasting of aircraft surfaces or structures, and damage to electrical/electronic systems.

DISSIMILAR REDUNDANCY. (18) The redundancy of systems that provide a redundancy of function, but by a different form.

DISSIMILAR SOFTWARE. (18) Redundant computer programs that provide a redundancy of function, but by a different form.

DISTRIBUTED BUS CONTROL. (18) The bus control approach where the total communication control job is distributed across the bus users, each controlling the communications during its period of responsibility.

DISTRIBUTED CONTROL. (6) Concurrent control from multiple points in the data bus system.

DOT PRODUCT. (20) The inner product of vectors $(x_1, \ldots, x_n)$ and $(y_1, \ldots, y_n)$ from n-dimensional euclidean space is the sum of $s_i y_i$, as i ranges from 1 to n.

DOUBLE FAIL-OPERATIONAL SYSTEM. (4) A quadruplex (or higher) redundant flight-control system which is designed to incur failures in two redundant lanes (or channels) before it fails.

DUAL-DUAL ARCHITECTURE. (4) Two parallel dual computers with a voting plane at the output of each dual computing lane.

DUAL FAIL-OPERATIONAL. (7) A reliability requirement placed on a system which requires the system to be operational after two failures have occurred.

DUAL GROUND. (11) Equipment case ground/return through two independent circuit paths to structure implemented in flammable zones and water leakage areas-each path meeting electrical conductivity (resistance) requirements.

ELECTRIC FIELD. (11) High-impedance, radiated voltage field, positive or negative, from a voltage source as contrasted to a low-impedance magnetic field from a current source.

ELECTROMAGNETIC COMPATIBILITY (EMC). (11) Operation within performance specification in the intended electromagnetic interference environment.

ELECTROMAGNETIC INTERFERENCE (EMI). (11) Conducted and radiated voltage/current noise signals, broadband (BB) or narrow band (NB), that degrade the specified performance of equipment.

ELECTROMIGRATION. (5) Drifting of metal atoms toward the cathode of a cathode ray tube.

ELECTROSTATIC CHARGE. (11) Electric potential energy with a surrounding electric field, uniform or nonuniform, moving or at rest, on a material.

EMISSION. (11) Voltage/current noise on a wire or in space. Broadband emission has uniform spectral energy over a wide frequency range and can be identified by the response of a measuring receiver not varying when tuned over several receiver "bandwidths." Or, energy present over a bandwidth greater than the resolution bandwidth where individual spectral components cannot be resolved. Broadband (BB) may be of two types: (1) impulse and coherent varies 20 dB per decade of bandwidth and (2) random or statistical, varies 10 dB per decade. A narrow band (NB) emission or signal, sometimes called continuous wave, occurs at a discrete frequency and does not vary with bandwidth.

EMPIRICAL MODEL. (20) The model of a system based on data and the principles of statistics. Empirical models produce relationships, not insight.

EMULATION. (18) The duplication of the behavior of a system with a different system.

ENVELOPE LIMITING. (4) General or additional limits imposed on the structural, "g" limits, speed, attitude, etc. of the aircraft. In some cases, envelope limiting imposes additional constraints on the envelope that cannot be exceeded regardless of pilot inputs.

EQUIPMENT TRANSIENT DESIGN LEVEL. (13) The level of transients which the equipment is qualified to withstand.

EQUIPMENT TRANSIENT SUSCEPTIBILITY LEVEL. (13) The transient level which will result in damage or upset to the system components. This level will be greater than the equipment transient design level.

EQUIVALENCE STATEMENT. (17) A FORTRAN statement which equates two variable names.

ERROR. (4) A mistake in specification, design, production, maintenance, or operation of a system causing undesirable performance.

ERROR. (8) A state of the system which (in the absence of any corrective action by the system) could lead to a failure that would not be attributed to any event subsequent to the error. (More accurately known as an erroneous state.)

ERROR MASKING. (18) The process of masking the presence of avionic errors, possibly by using an electronic voter to override an erroneous input with the values of substitute inputs.

EVENT. (20) A change to the blackboard of a Blackboard System.

EVENT, EXTREMELY IMPROBABLE. (4) An event with a probability of occurrence on the order of $10^{-9}$ or less.

EVENT, IMPROBABLE. (4) An event with a probability of occurrence on the order of $10^{-5}$ or less.

EVENT, PROBABLE. (4) An event with a probability of occurrence on the order of $10^{-5}$ or greater.

EXPERT. (20) 1. An Expert System that approaches an expert's level of performance within a given domain. 2. A person who has mastered solving specific types of problems.

EXPERT SYSTEM. (20) A computer system designed to simulate the problem solving behavior of a human expert.

EXPERT SYSTEM SHELL. (20) An Expert System development tool and run-time environment with a modifiable knowledge base.

EXTERNAL ENVIRONMENT. (13) Characterization of the natural lightning environment with idealized waveforms for engineering purposes.

FACETTE. (20) A piece of knowledge in a frame.

FAIL-OPERATIONAL. (7) A reliability requirement placed on a system which requires the system to be operational after a single failure has occurred.

FAIL-SAFE. (7) A reliability requirement placed on a system which requires that safe flight not be hindered even after a failure.

FAIL-SAFE. (18) A design philosophy that ensures that any failure in a system does not result in an unsafe condition after the failure.

FAILURE. (4) The inability of a system, subsystem, unit, or part to perform within specified limits.

FAILURE. (5) The deviation of system behavior from specifications (arithmetic failure, storage failure, flight control function failure.)

FAILURE. (8) The situation when the external behavior of a system does not conform to that prescribed by the system specification.

FAILURE, HARD. (5) Repeated use of the same input and initial conditions results in the same incorrect response.

FAILURE, HIDDEN. (4) A failure that is not manifested at the time of its occurrence.

FAILURE MECHANISM. (5) Any situation that could produce an error condition. Examples of failure mechanisms include metal migration, voltage overstress, and lack of air-conditioning.

FAILURE, PERMANENT. (5) Repeated use of the same input and initial conditions results in the same incorrect response.

FAILURE, SOFT. (5) Repeated use of the same input and initial conditions does not result in the same incorrect response.

FAILURE, TEMPORARY. (5) Repeated use of the same input and initial conditions does not result in the same incorrect response.

FAILURE, TRANSIENT. (5) Repeated use of the same input and initial conditions does not result in the same incorrect response.

FALL-TIME. (12) The time required for pulse amplitude to go from a predefined magnitude to a given level.

FALSE ALARM. (7) The declaration of a fault by a fault detection monitor or algorithm when there is no fault.

FAULT. (4) An error in the operation of a system.

FAULT. (5) The phenomenological reason for a failure (open wire, stuck-at fault, design fault, etc.). In general, any condition preventing a digital component from correctly changing state when directed to change by input parameters. For electrical components there is a one-to-one correspondence between faults and failures. The situation is not so simple with digital circuits. For if the circuit is S-A-1, any input causing a one output will be correctly processed; a little like the stopped clock that is correct twice per day. For a processor having a million or so logic gates, it is not possible to test for all the combinations of input and output states.

FAULT. (8) The adjusted cause of error.

FAULT AVOIDANCE. (9) The attempt to prevent any software faults in the final delivered product through disciplined software development practices, testing, and IV&V.

FAULT CONTAINMENT. (6,9) The capacity of a system to prohibit errors and/or failures from propagating from the source throughout the system.

FAULT CURRENT. (11) The maximum current (magnitude and duration) flowing through a fault point. This current is equal to the supply voltage divided by the dc resistance of power line leads, circuit breakers, and the current return in wire or structure.

FAULT DETECTION. (6) The capacity of a system to determine the occurrence of erroneous operation.

FAULT DETECTION. (7) The determination that a sensor is faulted by using a software algorithm.

FAULT, HARD. (4) A defect in the hardware or software of a digital control system that permanently affects some functional performance of the system.

FAULT INSERTION. (4) A testing technique used to obtain information about data latency and built-in test coverage of a digital flight-control system.

FAULT ISOLATION. (6) The capacity of a system to isolate a failure to the required level so it can reconfigure.

FAULT ISOLATION. (7) The determination that a particular sensor is faulted by using a software algorithm.

FAULT, LATENT. (5) A fault which has not yet caused a failure. (For example, a fault in a memory chip that is not being used for the foreground program or in this particular mode of the system is a latent fault.)

FAULT, SOFT. (4) A transient defect in the software of a digital flight-control system that can be overcome by error-correctable code or by recycling of power to the computer system.

FAULT, STUCK-AT. (5) A logic signal which remains at zero (S-A-0) or one (S-A-1).

FAULT TOLERANCE. (6,9) The capability to endure errors and/or failures without causing total system failure.

FAULT TOLERANCE. (7) Accommodation of sensor hardware faults based on some type of comparator scheme.

FAULT TOLERANCE. (18) The ability of a system to continue operation after a fault, possibly in a degraded condition.

FAULT TOLERANT. (4,9) Software which continues to operate satisfactorily in the presence of faults.

FAULT TOLERANT SYSTEM. (5) A system that continues to function although certain components may have faults.

FAULT TREE ANALYSIS. (4) A top-down deductive analysis that identifies the conditions and functional failures necessary to cause a defined failure condition. The fault tree can be used to establish the probability of the ultimate failure condition occurring as a function of the estimated probabilities of contributory events.

FEDERAL AVIATION REGULATIONS. (18,19) Subchapter C of the Code of Federal Regulations, Title 14, Chapter 1.

FEED FORWARD NEURAL NETWORK. (20) A Neural Network where the only connections are from a neuron in one layer to a neuron in the next layer.

FILTER. (11) Device or unit that passes or rejects a frequency band and is designed to block noise from entering or leaving a circuit or unit.

FILTER. (20) A transmission network used in electrical systems for the selective enhancement of a given class of input signals.

FINITE STATE MACHINE. (18) A state machine with a finite number of states.

FLIGHT CODE. (4) The application software of the digital flight-control system.

FLIGHT-CRITICAL. (4,7,18) A description of functions whose failure would contribute to or cause a failure condition preventing the continued safe flight and landing of the aircraft.

FLIGHT-ESSENTIAL. (4,18) A description of functions whose failure would contribute to or cause a failure condition which would significantly affect the safety of the airplane or the ability of its crew to cope with adverse operating conditions.

FLIGHT-NONESSENTIAL FUNCTION. (18) A function whose failure could not significantly degrade aircraft capability or crew ability.

FLIGHT-PHASE CRITICAL. (4) A description of functions which are critical only during certain phases of flight.

FLY-BY-GLASS. (16) Flight control system where fiber optics carry the signal.

FLY-BY-LIGHT. (4,16) Flight control system where fiber optics carry the signal.

FLY-BY-WIRE. (4,16) Flight control system with electric signaling.

FORWARD CHAINING. (20) A method of solving problems by beginning with certain data and moving down the inference chain until a solution is reached.

FORWARD RECOVERY. (9) Restoration of the system to a consistent state by compensating for inconsistencies found in the current state so that the system may continue processing.

FOURIER TRANSFORM. (12) A mathematical method for deriving the frequency spectrum from a time dependent function.

FRAME. (18) A formatted block of data words or bits that is used to construct messages.

FRAME. (20) A way of representing Expert System knowledge that consists of a set of slots that contain data.

FRAME LABEL. (20) A unique frame identification.

FUNCTIONAL PARTITIONING. (18) The partitioning of system functions by placing each group of users, which share a common function, on different data buses.

FUZZIFICATION. (20) The process of converting crisp inputs into fuzzy values.

FUZZY CONTROL VARIABLE. (20) A parameter whose value determines the action taken by fuzzy rules.

FUZZY EXPERT SYSTEM. (20) An Expert System that uses fuzzy logic techniques to solve problems.

FUZZY LOGIC. (20) The theory of representing vague or imprecise concepts to model inherent conditions.

FUZZY SET. (20) An extension of the concept of a set, in which the characteristic function which determines membership of an object in the set can take on any value between 0 and 1.

FUZZY TERMS. (20) Words used for descriptions that may not be precise in determining set memberships.

GATEWAY. (18) A bus user that is connected to more than one bus for the purpose of transferring bus messages from one bus to another, where the buses do not follow the same protocol.

GENERAL AVIATION AIRCRAFT. (18) The non-air transport civil aircraft.

GETTER. (19) A metal alloy used in a vacuum tube to absorb residual gasses.

GENERATOR POLYNOMIAL. (18) The polynomial code that is used to generate the remainder in the division of the CRC check.

GIGABIT. (16) One billion bits.

GLASS COCKPIT. (9) Advanced state-of-the-art electronic displays utilizing flat panel and/or cathode ray tube display technology for cockpit instrumentation.

GLOBAL STATE. (18) A state that represents the condition of the entire network being modeled, including senders, receivers, and the communication link.

GRACEFUL DEGRADATION. (20) A programming technique to prevent catastrophic system failure by allowing the machine to operate, though in a degraded mode, despite failure or malfunction of several integral units or subsystems.

GRAY SCALE. (19) A series of tones, varying from black to white.

GROUND. (11) A generic term having multiple meanings and indicating a circuit return path or a voltage reference: not "zero" voltage reference. Four hundred millivolts of noise voltage is common on "quiet" grounds. There are several types of returns and references.

GROUND EFFECT. (4) Increase in aircraft lift when operating near the ground.

HALF-DUPLEX. (18) Bidirectional communication between two entities on a single channel by each having a turn to control the channel.

HAMMING CODE. (18) An error detection and correction code based on the Hamming distance.

HAMMING DISTANCE. (18) The number of bit positions in which two binary words differ.

HANDSHAKING. (18) The reciprocal responses given by two electronic systems to sequence the steps of a transfer of data between them.

HARD FAILURE. (12) A failure that requires a reset of the equipment.

HARDWARE. (17) The physical components of a computer.

HARDWARE-IN-THE-LOOP SIMULATION. (18) A partial simulation of a system; part of the actual system is used in the simulation.

HAZARD FUNCTION. (8) The conditional probability that a fault is exposed in the interval t to $\Delta t$ given that the fault did not occur prior to time t.

HERMETIC. (19) Having an airtight seal.

HETEROASSOCIATIVE. (20) A Neural Network where output patterns are distinct from input patterns.

HEURISTIC. (20) Any rule of thumb, strategy, or technique used to limit the time required to search for solutions in large problem spaces.

HEURISTIC REASONING. (20) Using an expert's rules of thumb, in the absence of precise control mechanisms, to reduce the space that must be searched for a solution.

HIDDEN LAYER. (20) The middle layer in a Neural Network which takes outputs from the input layer, processes them, and passes them to the output layer.

IMMUNITY. (11) Capability of a circuit or unit to operate within performance specification in a specified electromagnetic interference environment.

INDIRECT EFFECTS. (13) Voltage and/or current transients induced by lightning in aircraft electrical wiring which can produce upset and/or damage to components within electrical/electronic systems.

INDUCED VOLTAGES. (13) A voltage produced around a closed path or circuit by changing magnetic or electric fields or structural IR voltages.

INDUCTIVE. (20) Learning by information repetition.

INFERENCE CHAIN. (20) The path of reasoning that the inference engine follows to find a solution to the problem.

INFERENCE ENGINE. (20) The part of an Artificial Intelligence system that uses knowledge in the knowledge base and acquired knowledge about the problem to form an expert solution.

INFERENCE MECHANISM. (20) Controls the use of the knowledge base and databases when solving a problem.

INITIALIZATION. (6) Setting the beginning parameters and values on system power-up. For redundant systems this includes setting the initial configuration of the system.

INPUT LAYER. (20) The first layer of neurons in a Neural Network. The input layer accepts the input values and passes them to the hidden layer. The input layer does not process data.

INTERNAL ENVIRONMENT. (13) The fields and structural IR potentials produced by the external environment, along with the voltages and currents induced by them.

INTERRUPT VECTOR. (18) The address that points to the beginning of the service routine for an interrupt.

INTERRUPT VECTOR TABLE. (18) The table of interrupt vectors for all interrupts serviced by a system.

ISOLATION. (11) Electrical separation and insulation of circuits from ground and other circuits or arrangement of parts to provide protection and prevention of uncontrolled electrical contact.

ISOTROPIC. (19) Having the same physical properties in all directions.

JOULE. (12) A unit of energy equal to one watt-second.

JUMPER/STRAP. (11) A short wire, strip, strap, or braid conductor installed to make a safety ground connection, to dissipate electrostatic charge, or establish continuity around a break in a circuit.

KILOBYTE. (16) One thousand bytes.

KNOWLEDGE ACQUISITION. (20) The process of gathering the required information from various sources by a knowledge engineer.

KNOWLEDGE BASE. (20) A form of coded knowledge about a specific domain.

KNOWLEDGE-BASED SYSTEM. (20) A computer system whose usefulness derives primarily from a database containing human knowledge in a computerized format.

KNOWLEDGE ENGINEER. (20) The person responsible for extracting, organizing, and encoding the knowledge related to the problem, and importing it into a knowledge base or database.

LABEL. (20) A data item that serves to identify a data record, or a symbolic name used in a program to mark the location of a particular instruction or routine.

LABELED ADDRESSING. (6) The system of identifying message recipients based on labels. This is in contrast to destination terminal addresses.

LATENT FAULT. (10) A fault which has not yet produced a malfunction. (In the context of the single-fault model, benign and latent faults are equivalent.)

LAYER. (20) A row of neurons in a Neural Network; there are usually three layers in a network.

LIGHTNING FLASH. (13) The total lightning event in which charge is transferred from one charge center to another. It may occur within a cloud, between clouds, or between a cloud and the ground. It can consist of one or more strokes, plus intermediate or continuing currents.

LIGHTNING LEADER STROKE. (13) The leader forms an ionized path for charge to be channeled towards the opposite charge center. The stepped leader travels in a series of short, luminous steps prior to the first return stroke. The dart leader reionizes the return stroke path in one luminous step prior to each subsequent return stroke in the lightning strike.

LIGHTNING RETURN STROKE. (13) A lightning current surge that occurs when the lightning leader makes contact with the ground or an opposite charge center.

LIGHTNING STRIKE. (13) Any attachment of the lightning flash to the aircraft.

LIGHTNING STRIKE ZONES. (13) Locations on the aircraft where the lightning flash will attach or where substantial amounts of electrical current may be conducted between attachment points. The location of these zones on any aircraft is dependent on the aircraft's geometry and operational factors and often varies from one aircraft to another.

LIMITING, VOLTAGE/CURRENT. (11) Semiconductor components, diodes, Transorb, or filter designed to clip and shunt to ground an applied transient or steadystate voltage. Used to protect against noise frequencies, faults, lightning, and inductive switching transients.

LINE REPLACEABLE UNIT. (18,19) An electronics unit that is made to be replaced on the flight line, as opposed to one that requires the aircraft be taken to the shop for repair.

LINEAR BUS. (18) A bus where users are connected to the medium; one on each end, with the rest connected in between.

LOGICAL SUM. (20) The final unified result of the rule processes of a fuzzy system.

LOW-PASS FILTER. (12) An electrical circuit which allows the passage of low frequencies and prevents the passage of high frequencies.

MACHINE LEARNING. (20) The process or technique by which a device modifies its own behavior as the result of its past experience and performance.

MAGNETIC FIELD. (11) A radiated, low-impedance field having lines of "flux" or magnetomotive force associated with an electrical current.

MALFUNCTION. (11) Failure or degradation in performance that compromises flight safety.

MANCHESTER II MODULATION. (18) A non-return to zero, bipolar modulation of a voltage that encodes bits based on the zero-crossing direction of the signal.

MEAN AERODYNAMIC CHORD (also mean chord). (4) The chord of an airfoil whose length is equal to the area of the airfoil section divided by the span.

MEAN FAILURE RATE. (10) A measure of survivability defined as the reciprocal of the mean time to system failure.

MESSAGE STRUCTURE. (6) The organization of both protocol and data information in a message.

META-KNOWLEDGE. (20) Knowledge that reveals details about the system knowledge.

METRIC. (17) A measure.

MICRON. (16) One-millionth of a meter.

MISSED ALARM. (7) The failure of a fault detection monitor or algorithm to detect a fault when there is a sensor fault.

MODEL. (20) A mathematical or physical system, obeying certain specified conditions, whose behavior is used to understand a physical, biological, or social system to which it is analogous in some way.

MODELING. (18) Creating a system of mathematical equations that formulate all the significant behavior of a system.

MODULE. (17) A unit of code which implements a function.

MONITORABILITY. (6) The capacity of the protocol to be viewed passively to allow observation of the dynamics of the protocol.

MONOLITHIC. (19) A single substrate used for an integrated circuit.

MONOTONIC FUNCTION. (17) A function in which a certain change in the measure always represents a certain change in the property being measured, where either change is simply an increase or decrease in magnitude.

MULTIPLE BURST. (13) A randomly spaced series of bursts of short duration, low amplitude current pulses, with each pulse characterized by rapidly changing currents. These bursts may result from lightning leader progression or branching and may be accompanied by or superimposed on stroke or continuing currents. The multiple bursts appear to be most intense at the time of initial leader attachment to the aircraft.

MULTIPLE STRIKE. (13) Two or more lightning strikes during a single flight.

MULTIPLE STROKE. (13) Two or more return strokes occurring during a single lightning flash.

MULTIPLE TRIP MONITOR. (7) A fault detection algorithm which declares a fault after the sensor output has exceeded a predefined threshold N times.

MULTIVERSION PROGRAMMING. (18) N-version programming.

N-VERSION PROGRAMMING. (18) The independent coding of a number, N, of redundant computer programs that are run concurrently for the purpose of comparing their outputs.

NANOSECOND. (16) One-billionth of a second.

NEGATIVELY STABILIZED. (4) Aircraft design in which the point of effective lift is aft of the center of gravity.

NEMATIC. (19) A liquid crystal phase with molecules having a single optical axis in line with an applied magnetic field.

NETWORK CONTROL STRATEGY. (6) The solution proposed by the designer in addressing his specific problem (design flexibility).

NEURAL NETWORK. (20) A system modelled after the brain, used to solve problems by a mapping of input data to output data.

NEURO-FUZZY. (20) Integrating a Neural Network and fuzzy logic to solve problems.

NEURON. (20) The computational element in a Neural Network. A neuron computes the weighted sum of the inputs from other neurons and produces an output.

NODE. (20) 1. Representation of an object that is related to other objects via connection weights. 2. The part of a Neural Network that calculates and processes information.

NOISE. (11) Conducted or radiated emission causing circuit upset, performance disorder, or undesired sound.

NOISE. (20) Meaningless or erroneous bits that must be ignored or removed from a signal, especially in communication channels.

NON REAL-TIME LEARNING. (20) Learning done in a Neural Network before attempting to solve application problems.

NONLINEARITY. (20) A system in which the outputs do not correspond to the inputs in a direct or inversely proportional relationship.

NUMERICAL APERTURE. (6) The angle of acceptance of light from a light source for a given fiber optic cable.

OBJECT CODE. (17) The translation of source code that is loaded into a computer.

OBJECT-ORIENTED. (20) A representation form for knowledge in which all properties of an object are associated by the outgoing and incoming arcs at its node; the representation is geared towards manipulating objects as independent pieces of knowledge.

OBSERVER. (7) An algorithm which models physical relationships between sensor data and uses the data to provide fault detection for one or more sensors. This is also known as a Luenberger observer or a signal blender.

OPERANDS. (17) The variables or constants on which the operators act.

OPERATORS. (17) Symbols which affect the value or ordering of operands.

OPPORTUNISTIC REASONING. (20) The process of solving problems in a way that is not uniform or predictable. Opportunistic reasoning attempts to solve the problem in the best possible way for each individual situation. Rules fire when they are appropriate, without a priori ordering.

OPTIMIZING COMPILER. (17) A computer program which, while translating source code into object code, removes inefficiencies from the code.

OUTPUT LAYER. (20) The layer of neurons in a Neural Network that takes the outputs from the hidden layer, processes them, and then produces the output of the network.

OVERHEAD. (18) The message timing gaps, control bits, and error detection bits added to some data to satisfy the data bus protocol.

PARALLEL PROCESSING. (20) Performing many different calculations simultaneously to reach one conclusion or solution.

PARAMETERIZATION CAPABILITY. (6) A measure of how well the attributes of the protocol can be described by parameters.

PARITY. (18) An error detection bit added to a data word based on whether the number of "one" bits is even or odd.

PARTITIONED. (18) Colocated hardware or software functions that are designed so that adverse interactions between them cannot occur.

PEAK RATE OF RISE. (13) The maximum instantaneous slope of the waveform as it rises to its maximum value. Mathematically, the peak rate of rise of a function, $i(t)$, may be expressed as the maximum of $d[i(t)]/dt$.

PETRI NET. (18) A state analysis diagram that tracks the status of the state transition conditions of a state machine.

PHONEME. (19) The smallest elements of speech.

PHOTOLITHOGRAPHY. (19) An integrated circuit fabrication technique using photographically produced masks.

PIN LEVEL TEST. (12) An EMC test in which voltage or current is applied directly to a conductor at a connector pin.

PIXEL. (19) The smallest picture element of a display.

POINT-MASS SIMULATION. (4) Same as state variables airplane model (q.v.).

POLLING. (18) A method whereby a CPU monitors the status of a peripheral by periodically reading its status signals.

POLYNOMIAL CODE. (18) A sequence of bits that represents the coefficients of each term in a polynomial.

POSITIVELY STABILIZED AIRCRAFT. (4) Aircraft design in which the effective point of lift is forward of the center of gravity.

PRECIPITATION STATIC (P-static). (11) Electrostatic discharge, corona, arcing, and streamering, steady state or impulsive, causing circuit upset, receiver noise or component damage.

PREDICATE CALCULUS. (20) A formal language of classical logic that uses functions and predicates to describe relations between individual entities.

PREDICATE/TRANSITION NETWORK. (4) A bipartite graph (a type of linear graph) to model concurrency between redundant concurrent events. Basically a modified generalized petri net.

PRIMARY STATION. (18) An intelligent HDLC protocol user, usually used to manage the access of other bus users to the bus.

PROBABILITY. (20) A theory dealing with the uncertainty that results from random behavior. Probability is defined over the numeric range of 0 to 1.

PROCESSING ELEMENT. (20) The part of the Neural Network where the computations are performed.

PRODUCTION RULES. (20) A common technique for representing procedural knowledge in an Artificial Intelligence system.

PROGRAM. (17) A detailed set of instructions for accomplishing some purpose.

PROPAGATION DELAY. (18) The time it takes an electrical signal to travel from its source to its destination.

PROTOCOL. (18) The set of rules by which all bus users must abide to access the bus and ensure its specified operation.

Q. (12) The quality factor of a resonant circuit which is the ratio of the energy stored to the power dissipated per cycle.

QUADRUPLEX ARCHITECTURE. (4) The use of four separate lanes (or channels) of computer redundancy. Each lane can fail separately providing a fail-operational capability for the digital flight-control system.

QUALITY MEASURE. (17) A repeatable, monotonic relationship relating measures of objects (a set of numbers) to subjective qualities.

RADIATED EMISSION (RE). (11) Electromagnetic energy transmitted and propagated in space usually considered as audio frequency or radio frequency noise.

RADIO FREQUENCY (RF). (11) Frequencies in the electromagnetic spectrum used for radio communications extending from kilohertz to gigahertz.

RADIO FREQUENCY INTERFERENCE (RFI). (11) Electromagnetic interference in the radio frequency range.

RASTER. (19) A pattern of parallel electron beam scan lines used for uniform coverage of a CRT screen.

REAL-TIME LEARNING. (20) Learning done in a Neural Network while solving an application problem.

RECONFIGURATION. (6) The capacity of a system to rearrange or reconnect the system elements or functions.

RECONFIGURATION. (18) The process of a system reassigning which hardware performs a particular function.

RECOVERY BLOCK. (18) A block of code executed upon detection of a fault to recover from the erroneous condition that results.

RECOVERY CACHE. (9) The location used to preserve input values until the outputs resulting from them have been accepted.

RECURSION. (20) A technique in which an apparently circular process is used to perform an iterative process.

REDUNDANCY MANAGEMENT. (7) The computer processing which is needed to implement fault detection and isolation algorithms.

REFERENCE. (11) 1. Structure, for electronics, shields, power. 2. A grid of wires, solid sheet, or foil. 3. A wire from circuit to grounding block or case. 4. A wire from circuit to structure. 5. Shield tie. 6. Earth.

REGISTER. (18) A single word of RAM located within an IC controller that is used for transferring data and control information.

RELAXED STATIC STABILITY AIRCRAFT. (4) An aircraft whose center of gravity is behind the wing's point of effective lift.

RELIABILITY ANALYSIS. (4) A means of determining the probability of failure in a system. Military flight-critical systems typically are required to have

reliability levels of $10^{-5}$ to $10^{-7}$, whereas civil flight-critical systems have reliability levels of $10^{-9}$ or less.

REMOTE TERMINAL. (18) The BIU portion of a MIL-STD-1553 bus user.

RESONANCE. (12) Resonance occurs in an electrical circuit when the energy stored in the inductance is equal to the energy stored in the capacitance.

RETURN. (11) 1. Structure, for power, fault, and "discrete" circuits. 2. A grid of wires, solid sheet, or foil. 3. A wire from circuit load back to source or to case. 4. Circuit card "ground plane," also a reference and shield.

RETURN STROKE. (13) See lightning return stroke.

REVERSION MODE. (7) The high level of redundancy in a system having different redundancy requirements for some sensors. Critical sensors may have a high level of redundancy while other sensors have low levels.

RING BUS. (18) A bus where users are connected only to the two adjacent users in a continuous ring; each connected to the next and the last one connected to the first one.

RISE-TIME. (12) The time required for a voltage pulse to reach a predefined magnitude from a given level.

ROBUSTNESS. (9) The ability of the code to perform despite some violation of the assumptions in its specifications usually via substitution of an alternate value and continuation of execution if a software fault is detected.

ROLLBACK. (9) Retrying the calculation in the event that a failure is detected, under the assumption that some external condition may have changed thereby resolving the anomaly.

RULE-BASED. (20) An Artificial Intelligence program where knowledge is represented as rules.

RULES. (20) A method of representing knowledge in the form of IF THEN statements.

SCHEDULER. (20) A part of an inference engine that determines which knowledge source should be activated, and in what order.

SEALANT. (11) An applied substance enclosing and protecting the integrity of a joint, fastener, or electrical bond from moisture, contaminants, oxidation, and acid or alkaline corrosion.

SEARCH TREE. (20) The branches of possible solutions to a problem that start with initial information and spread out following different paths.

SECONDARY STATION. (18) A simple HDLC protocol user.

SELF-SUPERVISED. (20) A method of training a Neural Network without the use of an external monitor. A feedback device that detects errors and adjusts the connection weights accordingly is used.

SEMANTIC NETWORK. (20) A method of representing knowledge using nodes and arcs.

SENSOR. (7) An instrument which measures a particular physical parameter. The data output may be digital or analog and is utilized by the flight computer.

SENSOR. (18,19) Any transducer that converts the measurement of a physical quantity to an electrical signal.

SEQUENTIAL LIKELIHOOD RATIO TEST. (7) A fault detection algorithm which is based on two hypothesized density functions of no fault or sensor fault.

SEQUENTIAL PROBABILITY RATIO TEST. (7) See sequential likelihood ratio test.

SERIAL DATA BUS. (18) A data bus capable of sending only one bit at a time, in series.

SERVICE SPECIFICATION. (18) The specification of the service provided by a protocol layer.

SHIELD. (11) A conductive material, opaque to electromagnetic energy, for confining or repelling electromagnetic fields. A structure, skin panel, case, cover, liner, foil, coating, braid, or cable-way that reduces electric and magnetic fields into or out of circuits or prevents accidental contact with hazardous voltages.

SHIELD EFFECTIVENESS (SE). (11) The ability of a shield to reject electromagnetic fields. A measure of attenuation in field strength at a point in space caused by the insertion of a shield between the source and the point.

SHIELDING. (12) Any metallic structure such as the aircraft fuselage or the woven braid on a cable that provides protection against electromagnetic fields.

SIGNAL RETURN. (11) A wire conductor between a load and the signal or driving source. Structure can be a signal and power return. Commonly, it is the low voltage side of the closed loop energy transfer circuit.

SIMULATION. (18,19) An approximated representation of the behavior of a system with a similar system.

SINGLE-ENDED CIRCUIT. (11) A circuit with source and load ends grounded to case and structure and using structure as return.

SINGLE-POINT FAILURE. (18) A failure of a component that, by itself, causes the failure of the system in which it is contained.

SINUSOID. (12) A wave form that follows the mathematical values of a sine function.

SLOTS. (20) Subdivisions making up the frame that may contain data, procedures, or pointers to other frames.

SMECTIC. (19) A liquid crystal phase with molecules arranged in layers.

SOFT FAILURE. (12) A failure which causes an alteration of data or missing data.

SOFTWARE. (17) Computer programs and the documentation associated with the programs.

SOFTWARE METRIC. (17) A measure of software objects.

SOFTWARE QUALITY FACTOR. (17) Any software attribute that contributes either directly or indirectly, positively or negatively, toward the objectives for the system in which the software resides.

SOFTWARE QUALITY METRIC. (17) (1) A measure that relates measures of the software objects (the symbols) to the software qualities (quality factors). (2) The measure of a software quality factor.

SOURCE CODE. (17) Code that can be read by people.

SPECIAL CONDITION. (18) A regulatory document that adds to, or otherwise alters, the airworthiness standards for particular aircraft.

SPURIOUS STATE. (20) A point in a Neural Network that acts as an attractor, but is not one of the stored patterns.

STABILITY. (20) The property of a system which remains under control and responds in a reasonable manner to an applied input.

STATE-VARIABLE AIRPLANE MODEL (also point-mass model). (4) Fixed aerodynamic variables are used in the solution of the equations of motion of the model instead of using look-up tables in which each derivative varies with airspeed, altitude, etc. The model performance is only accurate at or near the point in the flight envelope for which the variables are chosen.

STATIC MARGIN. (4) The degree of instability in a relaxed statically stable airplane.

STATION. (18) Bus user.

STATIONARY BUS CONTROL. (18) Bus control that is continually performed by a single bus controller, or by one of its backups.

STATUS REGISTER. (18) A register in an IC controller that holds the status of the state of certain controller functions.

STROKE. (19) A nonuniform technique of CRT screen refresh using penlike control of the electron beam.

STROUD NUMBER. (17) The total number of elementary mental discriminations that a person makes per second.

STRUCTURAL IR VOLTAGE. (13) The portion of the induced voltage resulting from the product of the distributed lightning current, I, flowing through the resistance, R, of the aircraft skin or structure.

STRUCTURE. (11) Basic members, supports, spars, stanchions, housing, skin panels, or coverings that may or may not provide conductive return paths and shields for electrical/electronic circuits.

STUB. (18) The short length of cable used to attach a single LRU to a data bus.

SUBROUTINE. (17) A self-contained body of code which can be called by other routines to perform a function.

SUPER-DIAGNOSTIC FILTER. (7) An algorithm which provides all the capabilities of a diagnostic filter. Additionally, it can isolate a specific faulted sensor. At the current time, this is the most complex technique used to implement analytical redundancy.

SUPERVISED LEARNING. (20) A method of training a Neural Network in which an external supervisor monitors the activities, and learning occurs on the basis of direct comparison of the output of the network with known correct answers.

SUSCEPTIBILITY. (11) Upset behavior or characteristic response of an equipment when subjected to specified electromagnetic energy. Identified with the point, threshold, or onset of operation outside of performance limits. Conducted Susceptibility (CS) applies to energy on interface conductors; Radiated Susceptibility (RS) to radiated fields.

SWEPT STROKE. (13) A series of successive attachments due to sweeping of the flash across the surface of the airplane by the motion of the airplane.

SYMBOLIC PROCESSING. (20) A method of processing knowledge where the relationships among the knowledge are stored using symbolic representations; thus, the system can deal freely with objects and not be concerned with their composition.

SYNAPSES. (20) Pathways in a biological neuron that transmit pulses from one neuron to another.

SYNCHRONOUS. (20) In step, or in phase, as applied to two or more circuits, devices, or machines.

SYNCHRONOUS MESSAGES. (6) Messages transmitted at a known a priori sequence and time or time interval.

SYSTEM EXPOSURE TIME. (4) The period during which a system may fail. This period extends from the last verified proper functioning to the completion of the next required performance.

SYSTEM FUNCTIONAL UPSET. (13) Impairment of system operation, whether permanent or momentary (e.g., a change of digital or analog state) which may or may not require manual reset.

SYSTEM INTEGRATOR. (18) The developer who has the responsibility to integrate the various subsystems into a working system.

SYSTEM INTEGRITY. (6) The degree to which a system is dependable.

SYSTEM RELIABILITY. (5) The probability of performing a given function from the some initial time, t=0, to time t.

TESTABILITY. (6) A measure of how well the protocol supports completeness of testing and the protocol's ability to produce repeatable or predictable results.

THRESHOLD, NOISE. (11) The lowest electromagnetic interference signal level that produces onset of susceptibility.

THROUGHPUT. (6) The productivity of a data processing system as expressed in computing work per minute or hour.

THYRISTORS. (16) Solid-state devices that convert alternating current to direct current.

TIME CONSTANT. (4) Time required to double the amplitude of the divergent real root in the pitch axis of the aircraft model.

TOKEN PASSING PROTOCOL. (18) A protocol that limits bus access to the user that has just received the token word.

TRADITIONAL PROGRAMMING. (20) The use of standard programming languages, as opposed to application development languages, financial planning languages, query languages, and report programs.

TRAINING. (20) A change in connection weight values of a Neural Network that results in capturing information that can be recalled later.

TRANSFER FUNCTION. (20) The mathematical relationship between the output of a control system and its input: for a linear system, it is the Laplace transform of the output divided by the Laplace transform of the input under conditions of zero initial-energy storage.

TRANSIENT CONTROL LEVEL. (13) The maximum allowable level of transients appearing at the systems interfaces as a result of the defined external environment.

TRANSPARENCY. (20) A characteristic of knowledge in an Expert System referring to its independence from other knowledge. There is no processing information in the transparent knowledge.

TRANSPARENT RECOVERY. (4) Correcting a soft fault without interrupting the system's intended performance.

TRANSPORT AIRCRAFT. (19) Aircraft used in interstate, overseas, or foreign air transportation.

TRIBOELECTRIC CHARGING. (13) Static electricity produced on a structure from the effects of friction.

UNACCEPTABLE RESPONSE. (11) Upset, degradation of performance, or failure, not designated a malfunction, but is detrimental or compromising to cost, schedule, comfort, or workload.

UNBALANCED CONFIGURATION. (18) A bus using the HDLC protocol that connects one primary and one or more secondary stations.

UNDESIRABLE RESPONSE. (11) Change of performance and output, not designated a malfunction or safety hazard, that is evaluated as acceptable as is because of minimum nuisance effects and excessive cost burdens to correct.

UNIDIRECTIONAL DATA BUS. (18) A data bus with only one user that is capable of transmitting.

UNSUPERVISED LEARNING. (20) A method of training a Neural Network, where no external monitor is involved; the Neural Network organizes itself by grouping and generating its own classification of inputs.

UPSET. (11) Temporary interruption of performance that is self-correcting or reversible by manual or automatic process.

UPSET. (12) A condition in which the state of a digital device is unintentionally altered, but may be restored by automatic means or by operator intervention.

UPSET. (13) See system functional upset.

VALIDATION. (4,11) Demonstration and authentication that a final product operates in all modes and performs consistently and successfully under all actual operational and environmental conditions founded upon conformance to the applicable specifications.

VALIDATION. (18) The process of evaluating whether or not items, processes, services, or documents accomplish their intended purpose in their operating environment.

VERIFICATION. (4,11) Demonstration by similarity, previous in-service experience, analysis, measurement, or operation that the performance, characteristics, or parameters of equipment and parts demonstrate accuracy, show the quality of being repeatable, and meet or are acceptable under applicable specifications.

VERIFICATION. (18) The act of reviewing, inspecting, testing, checking, auditing, or otherwise establishing and documenting whether or not items, processes, services, or documents conform to specified requirements.

VERY LARGE SCALE INTEGRATION. (20) Describing Integrated Circuits with more than 1000 elements.

VOTING PROCEDURE. (8) An algorithm included in fault tolerant software which uses the consensus recovery block method. It compares outputs of the n independent versions and determines which outputs are correct by identifying agreements among two or more versions.

WEIGHT MATRIX. (20) The collection of connection weights for an entire Neural Network.

WELL-BEHAVED FUNCTION. (17) A smooth mathematical relationship.

## ACRONYMS AND ABBREVIATIONS

| | |
|---|---|
| $\lambda$ (17) | Language Level |
| $\mu$C | micro Controller |
| $\mu$m (6,20) | Micrometer |
| $\mu$s (18) | Microsecond |
| $\eta$ (17) | Vocabulary of a Program |
| $\eta_1$ (17) | Number of Unique Operators |
| $\eta_1^*$ (17) | Minimum Number of Unique Operators |
| $\eta_2$ (17) | Number of Unique Operands |
| $\eta_2^*$ (17) | Number of Different Input and Output Parameters |
| $\phi$M (6) | Phase Modulation |
| $1/E_0$ (17) | Average number of discriminations a person is likely to make for each bug introduced into the code. |
| 3-D (16) | Three-Dimensional |
| A/C (11) | Aircraft |
| A/L (3) | Approach/Land |
| $A^3I$ (19) | Army-NASA Aircrew/Aircraft Integration |
| AAES (15) | Advanced Aircraft Electrical System |
| ABET (19) | Aerospace Behavioral Engineering Technology |
| ac (3,6,12,15,19) | Alternating Current |
| AC (3,5,14,17,18,19,20) | Advisory Circular |
| ACAP (13) | Advanced Composite Airframe Program |
| ACARS (11,12) | ARC Communications Addressing and Reporting System |
| ACES (13) | Applied Computational Electromagnetics Society |
| ACK (18) | Acknowledge |
| ACO (18,20) | Aircraft Certification Office |
| ACS (16) | Automatic Control System |
| ACT (11,12) | Active Controls Technology |
| ACT (17) | Analysis of Complexity Tool |
| AD (19) | Alerting Display |
| ADC (11,12) | Air Data Computer |
| ADF (11,12,19) | Automatic Direction Finder |
| ADI (3) | Automatic Direction Indicator |
| AE (6) | Avionics Equipment |
| AE4L (5,13) | SAE Subcommittee (Lightning) |
| AEEC (18) | Airlines Electronic Engineering Committee |
| AEHP (13) | Atmospheric Electricity Hazards Protection |
| AERA (16) | Automated En Route Air Traffic Control System |
| AES-S (6) | Aerospace and Electronic Systems Society |
| AF (11,12) | Audio Frequency |
| AFBW (4) | Augmented Fly-By-Wire |
| AFCS (11,12) | Automatic Flight Control System |
| AFFDL (7,8,13) | Air Force Flight Dynamics Laboratory |

| | |
|---|---|
| AFM (16) | Advanced Fuel Management |
| AFSC (18) | Air Force Systems Command |
| AFWAL (6,13) | Air Force Wright Aeronautical Laboratory |
| AGARD (8,20) | Advisory Group for Aerospace Research and Development |
| AHRS (6) | Attitude Heading Reference System |
| AI (16,19,20) | Artificial Intelligence |
| AIAA (5,6,9,15,18,20) | American Institute of Aeronautics and Astronautics |
| AIM (18) | Advanced Integrated MUX |
| AIM (20) | Airmen's Information Manual |
| AIR (18,19) | Aerospace Information Report |
| AIRLAB (5,18) | Avionics Integration Research Laboratory |
| AK (7) | Altitude Kinematics |
| ALCM (13) | Air Launched Cruise Missile |
| ALPA (19) | Air Line Pilot's Association |
| ALU (3,5,10) | Arithmetic Logic Unit |
| AM (19) | Active-Matrix |
| AM (6,14) | Amplitude Modulation |
| AM (5) | Amplitude Modulated |
| AMS (19) | Aerospace Material Specification |
| AMSC (5) | Document number prefix used by the Department of Defense |
| ANSI (11,12,20) | American National Standards Institute |
| AOA (4) | Angle of Attack |
| AP (18) | Application Processor |
| APU (11,12,15) | Auxiliary Power Unit |
| AR (7) | Analytical Redundancy |
| ARC (11,12) | Aeronautical Radio, Incorporated |
| ARIES (3) | Automated Reliability Interactive Estimation System |
| ARINC (3,6,18,20) | Aeronautical Radio, Incorporated |
| ARP (17,18,19,20) | Aerospace Recommended Practice |
| ARPA (20) | Advanced Research Projects Agency (formerly DARPA) |
| ARTERI (15) | Analytical Redundancy Technology for Engine Reliability Improvement |
| AS (19) | Aerospace Standard |
| ASCB (6,18) | Avionics Standard Communications Bus |
| ASDS (11,12) | Airport Surface Detection System |
| ASEE (5,15) | American Society of Electrical Engineers |
| ASIC (20) | Application Specific Integrated Circuit |
| ASME (5) | American Society of Mechanical Engineers |
| ASRS (19) | Aviation Safety Reporting System |
| ATC (19,20) | Air Traffic Control |
| ATCRBS (11,12,14) | Air Traffic Control Radar Beacon System |
| ATE (20) | Automatic Test Equipment |
| ATF (16) | Advanced Tactical Fighter |
| ATI (16) | Access Time Interval |
| ATTR (5) | Attribute |
| AWACS (14) | Airborne Warning and Control System |

| | |
|---|---|
| $\hat{B}$ (17) | Number of Bugs (Estimated) |
| B-dot (13) | Derivative of the magnetic field with respect to time |
| B-GLOSS (5) | Gate Logic Software Simulator developed by Bendix |
| BABBAGE (20) | Boeing Advanced BlackBoard Ada Generation Environment |
| BAC (18) | Balanced Asynchronous Configuration |
| BAT (17) | Battlemap Analysis Tool |
| BB (11,12) | Broadband |
| BC (18) | Bus Controller |
| BCAC (18) | Boeing Commercial Airplane Company |
| BCD (18) | Binary Coded Decimal |
| BCI (12) | Bulk Cable Injection |
| BFCS (18) | Beacon Frame Check Sequence |
| BGU (10) | Bus Guardian Unit |
| BIR (6) | Benchmark Information Rate |
| BIT (6,15,18,20) | Built-In Test |
| BITE (6,11,12,19) | Built-In Test Equipment |
| BIU (6,18) | Bus Interface Unit |
| BNR (18) | Binary |
| BOCP (18) | Bit-Oriented Communications Protocol |
| BP (18) | Basic Protocol |
| bps (6) | bits per second |
| BUSY (18) | Destination Busy |
| BW (11,12) | Bandwidth |
| C/I (5) | Communicator Interstage |
| C (7) | Comparator |
| CA (18) | Criticality Analysis |
| CAA (14,19) | Civil Aviation Authority |
| CAD (5,19) | Computer Aided Design |
| CALSEL (19) | Call Select |
| CAP (5) | Collins Application Processor |
| CAPS (3) | Computer Aided Production Simulator |
| CARE (3,5) | Computer Aided Reliability Evaluator |
| CARSRA (3,7) | Computer-Aided Redundant System Reliability Analysis |
| CAS (11,12) | Criticality Advisory System |
| CASSY (20) | Cockpit Assistant System |
| CAST (3) | Complementary Analytic Simulative Technique |
| CBD (5) | Commerce Business Daily |
| CBI (19) | Computer Based Instruction |
| CCITT (6) | Consultative Committee for International Telephone and Telegraph |
| CD (6) | Collision Detection |
| cdf (8) | Cumulative Density Function |
| CDU (11,12,19,20) | Control Display Unit |
| CE (11,12) | Conducted Emission |
| CE (17,18,19,20) | Certification Engineer |
| CF (20) | Certainty Factor |
| CFR (17,19) | Code of Federal Regulations |

| | |
|---|---|
| cm (19) | Centimeter |
| CM (20) | Configuration Management |
| CM (11,12) | Common Mode |
| CMC (18) | Current Mode Coupler |
| CMOS (5,12,20) | Complimentary Metal-Oxide Semiconductor |
| CMU (19) | Computer Mock-Up |
| CNI (20) | Communication, Navigation, and Identification |
| COMBIMAN (19) | Computerized Biomechanical Man-model |
| CONUS (14) | Contiguous United States |
| CP (18) | Combined Protocol |
| CPA (5) | Central Processor - A |
| CPU (5,10,18,19,20) | Central Processing Unit |
| CR (5) | Contractor Report |
| CR (6) | Command Response |
| CR/LF (17) | Carriage Return/Line Feed |
| CRC (6,18) | Cyclic Redundancy Check |
| CRM (19) | Cockpit Resource Management (also Crew Resource Management) |
| CRMI (2) | Computer Resource Management, Incorporated |
| CRT (11,12,16,19) | Cathode Ray Tube |
| CS (11,12) | Conducted Susceptibility |
| CSC (9) | Computer Software Component |
| CSCI (9,17) | Computer Software Configuration Item |
| CSDB (18) | Commercial Standard Data Bus |
| CSDL (5,10,15) | Charles Stark Draper Laboratories |
| CSERIAC (19) | Crew System Ergonomics Information Analysis Center |
| CSMA (6,16,18) | Carrier Sensed Multiple Access |
| CSMA/CD (6) | Carrier Sense Multiple Access/Collision Detection |
| CT (2,6) | Technical Center (designation used in FAA report numbering scheme) |
| CTA (3) | CAPS Test Adapter |
| CTA (5) | Collins Test Adaptor |
| CTS (18) | Clear To Send |
| CW (13) | Continuous Wave |
| D (17) | Program Difficulty |
| DADC (6) | Digital Air Data Computer |
| DARPA (16) | Defense Advanced Research Projects Agency (Now known as Advanced Research Projects Agency) |
| DATAC (6,18) | Digital Autonomous Terminal Access Communication |
| dB (6,12) | Decibel |
| dBi (14) | Decibels with respect to one milliampere |
| dBm (6) | Decibels with respect to one milliwatt |
| dc (6,12,15,19) | Direct Current |
| DC (18) | Display Computer |
| DE (5) | Diagnostic Emulation |
| DEFN (5) | Definition |
| DEP (19) | Design Eye Position |
| DET (18) | Driver Enable Timer |
| DEV (5) | Development |

| | |
|---|---|
| DF (7) | Diagnostic Filter |
| DFC (7) | Digital Flight Control |
| DFCS (3,4,7,16) | Digital Flight Control System |
| DFDAU (11,12) | Digital Flight Data Acquisition Unit |
| DFDR (11,12) | Digital Flight Data Recorder |
| DGAC (14) | Directorate Generale Aviation Civile |
| DISAC (15) | Digital Integrated Servo Actuator Controller |
| DITS (6,11,12,18) | Digital Information Transfer System |
| DM (11) | Differential Mode |
| DM (6) | Delay Modulation |
| DMA (5,6) | Direct Memory Access |
| DMA (18) | Direct Memory Addressing |
| DME (11,12,18,19) | Distance Measuring Equipment |
| DNA (13) | Defense Nuclear Agency |
| DoD (5,8,12,14,16,19,20) | Department of Defense |
| DOE (13) | Department of Energy |
| DOS (20) | Disk Operating System |
| DOT (2,3,6,7,8,20) | Department of Transportation |
| DR (17) | Direct Ratio (Average) |
| DRB (9) | Distributed Recovery Block |
| DS (17) | Direct Score |
| DSP (3) | Discrete Switch Panel |
| DSP (20) | Digital Signal Processor |
| DTSA (18) | Dynamic Time Slot Allocation |
| | |
| E/E (11,12) | Electrical/Electronic |
| E-FIELD (11,12) | Electric Field |
| E (17) | Programming Effort |
| E-dot (13) | Derivative of the electric field with respect to time |
| $E^3$ (11,12) | Electromagnetic Environmental Effects |
| EADI (11,12,19) | Electronic Attitude Director Indicator |
| ECAC (11,12,14) | Electromagnetic Compatibility Analysis Center |
| ECM (14) | Electronic Counter Measures |
| ECS (11,12) | Environmental Control System |
| EEC (5,11,12,18) | Electronic Engine Control |
| EED (11,12) | Electro-Explosive Device |
| EEPROM (20) | Electrically Erasable Programmable Read Only Memory |
| EES (18) | Electromagnetic Emission and Susceptibility |
| EFID (18) | Electronic Flight Instrument Display |
| EFIS (11,12,18,19) | Electronic Flight Instrument System |
| EFMA (3) | Executive Failure My A |
| EFMB (3) | Executive Failure My B |
| EFOA (3) | Executive Failure Other A |
| EFOB (3) | Executive Failure Other B |
| EFW (3) | Executive Failure Word |
| $E_g$ (19) | Energy gap |
| EGT (11,12) | Exhaust Gas Temperature |
| EHSI (11,12) | Electronic Horizontal Situation Indicator |
| EIA (18) | Electronic Industries Association |
| EICAS (11,12) | Engine Indication and Crew Alerting System |

| | |
|---|---|
| EIU (16) | Electronic Interface Unit |
| EL (19) | Electroluminescent |
| ELS (19) | Electronic Library System |
| EM (5,11,12,13) | Electromagnetic |
| EMA (15) | Electromechanical Actuator |
| EMAS (2,15) | Electromechanical Actuator System |
| EMC (6,11,12,13,14) | Electromagnetic Compatibility |
| EMCad$^{tm}$ (12) | Electromagnetic Computer aided design |
| EME (14) | Electromagnetic Environment |
| EME (11,12) | Electromagnetic Effects |
| EMI (5,6,11,12,13,15,16) | Electromagnetic Interference |
| EMIC (11,12) | Electromagnetic Interference/Compatibility |
| EMP (11,12,13) | Electromagnetic Pulse |
| EMR (5,14) | Electromagnetic Radiation |
| EMUX (6) | Electrical Multiplex |
| ENRZ (6) | Enhanced Non-Return to Zero |
| EOF (17) | End of File |
| EPES (20) | Emergency Procedures Expert System |
| EPR (11,12,15) | Engine Pressure Ratio |
| EPROM (16) | Erasable Programmable Read-Only Memory |
| ES (20) | Expert System |
| ESD (11,12) | Electrostatic Discharge |
| ESE (11,12) | Electric (field) Shield Effectiveness |
| ESS (5,9) | Electronic Switching System |
| ETANN (20) | Electrically Trainable Analog Neural Network |
| ETDL (13) | Equipment Transient Design Level |
| EUROCAE (14) | European Organization for Civil Aviation Electronics |
| EVS (19) | Enhanced Vision System |
| EXCHNG (5) | Exchange |
| EXP (5) | Experiment |
| F/FA (18,20) | Fault and Failure Analysis |
| FAA (ALL) | Federal Aviation Administration |
| FADEC (6,15,16) | Full Authority Digital Engine Controller |
| FAFTEEC (16) | Full Authority Fault Tolerant Electronic Engine Control |
| FAR (17,18,19,20) | Federal Aviation Regulation |
| FAR (3,4,6,16) | Federal Acquisition Regulation |
| FAS (19) | Flight Deck Alerting System |
| FBL (15,16) | Fly-By-Light |
| FBW (4,7,16) | Fly-By-Wire |
| FCA (20) | Fuzzy Computational Accelerator |
| FCC (3,4,5,6,10,11, 12,15,18,19) | Flight Control Computer |
| FCR (5) | Fault Containment Regions |
| FCS (18) | Frame Check Sequence |
| FCS (4,5,10,16) | Flight Control System |
| FD (7) | Fault Detection |
| FDEP (11,12) | Flight Data Entry Panel |
| FDFM (15) | Fault Detection and Failure Management |
| FDI (19) | Flight Director Indicator |

| | |
|---|---|
| FET (5,16,19) | Field Effect Transistor |
| FI (7) | Fault Isolation |
| FI (5) | Fault Insertion Circuitry |
| FIAT (5) | Fault Injection Automated Testing |
| FICA (15) | Failure Indication and Corrective Action |
| FIDE (20) | Fuzzy Inference Development Environment |
| FIIS (5,10) | Fault Insertion and Instrumentation System |
| FIL (20) | Fuzzy Inference Language |
| FIM (5) | Fault Injection Manager |
| FINDER (20) | Flight-Plan Interactive Negotiation and Decision-Aiding System for Enroute Rerouting |
| FIRE (5) | Fault Injection Receptor |
| FLEX (20) | FORTRAN Library for EXpert systems |
| FM (5) | Frequency Modulated |
| FM (6,14) | Frequency Modulation |
| FMC (11,12,19) | Flight Management Computer |
| FMEA (3,9,18) | Failure Mode and Effect Analysis |
| FMECA (5,15) | Failure Modes and Effects Criticality Analysis |
| FMECA (18) | Failure Mode, Effects, and Criticality Analysis |
| FMS (19) | Flight Management System |
| FOV (19) | Field of View |
| FP (17) | Function Point |
| FSM (18) | Finite State Machine |
| ft (14) | feet |
| FT (5) | Fault Tolerant |
| FTA (18) | Fault Tree Analysis |
| FTC (5) | Fault Tree Compiler |
| FTMP (5,10,18) | Fault-Tolerant Multiprocessor |
| FTP (5) | Fault Tolerant Processor |
| G/E (13) | Graphite Epoxy |
| g1 (19) | Grid 1 |
| g2 (19) | Grid 2 |
| g3 (19) | Grid 3 |
| g4 (19) | Grid 4 |
| GA (18) | General Aviation |
| GaAs (6,16,19,20) | Gallium Arsenide |
| GaAsP (19) | Gallium Arsenide Phosphide |
| GAMA (6,18) | General Aviation Manufacturers Association |
| GaP (19) | Gallium Phosphide |
| GCR (6) | Group Code Recording |
| GE (5) | General Electric |
| GEMACS (13) | General Electromagnetic Model for the Analysis of Complex Systems |
| GEN (5) | Generation |
| GGLOSS (5) | Generalized Gate-Level Logic System Simulator |
| GLOSS (5) | Gate Logic Software Simulator |
| GNC (16) | Guidance, Navigation, and Control |
| GPC (5) | General Purpose Computer |
| GPS (11,12) | Global-Positioning-System |
| GPWS (11,12) | Ground Proximity Warning System |
| Gr/Ep (11,12) | Graphite/Epoxy |

| | |
|---|---|
| GS (10) | Glideslope |
| GUI (20) | Graphical User Interface |
| H-FIELD (11,12) | Magnetic Field |
| H1 (11,12) | Fan Speed |
| HA (18) | Hazard Analysis |
| HARP (5,18) | Hybrid Automated Reliability Predictor |
| HDBK (5) | Handbook |
| HDD (19) | Head-Down Display |
| HDLC (6,18) | High-Level Data Link Control |
| HERF (5,14,16,18) | High-Energy Radio Frequency |
| HF (11,12,13,14,19) | High-Frequency |
| Hg (19) | Mercury |
| HIRF (15,18,19) | High-Intensity Radiated Fields |
| HLD (19) | Head-Level Display |
| HMD (19) | Helmet Mounted Displays |
| HMS (19) | Helmet Mounted Sight |
| HOL (17,20) | High Order Language |
| HSI (3,19) | Horizontal Situation Indicator |
| HSRB (6,18) | High-Speed Ring Bus |
| HUD (19) | Head-Up Display |
| HVDC (15) | High Voltage dc |
| HW (18) | Hardware |
| Hz (3,15,18,19) | Hertz |
| I (13) | Current |
| I (17) | Intelligence Content |
| I-dot (13) | Derivative of the current with respect to time |
| I/O (5,10,15,18,20) | Input/Output |
| IAA (3) | Integrated Assurance Assessment |
| IAAC (11,12) | Integrated Application of Active Controls Technology (to an Advanced Subsonic Transport Project) |
| IACS (18) | Integrated Avionic Computer System |
| IBM (5) | International Business Machines |
| IC (3,18,19,20) | Integrated Circuit |
| ICAO (14) | International Civil Aviation Organization |
| ICIS (5) | Intercomputer Interface Sequencer |
| ICS (5) | Intercomputer Sequencer |
| ID (5) | Identification |
| ID (18) | Identifier |
| IDG (11,12) | Integrated Drive Generator |
| IEEE (5,6,9,17,18,20) | Institute of Electrical and Electronics Engineers, Incorporated |
| $I_F$ (19) | Forward Current |
| IETM (20) | Interactive Electronic Technical Manual |
| IFC (17) | Information Flow Complexity |
| IFCS (18) | Information Frame Check Sequence |
| IFF (14,19) | Identification - Friend or Foe |
| IGGLOSS (5) | Gate Logic Software Simulator (improved version developed at NASA Langley) |

| | |
|---|---|
| IJCNN (20) | International Joint Conference on Neural Networks |
| ILS (3,11,12,19) | Instrument Landing System |
| IMR (18) | Interrupt Mask Register |
| INS (6,11,12,19) | Inertial Navigation System |
| IOPA (5) | Input/Output Processor - Channel A |
| IOS (5) | Input/Output Subsystem |
| IR (19) | Infrared |
| IRS (11,12,19) | Inertial Reference System |
| ISO (17,18) | International Standards Organization |
| ITO (19) | Indium Tin Oxide |
| ITS (19) | Intelligent Tutoring System |
| ITT (16) | (Consultative Committee for) International Telegraphy and Telephony |
| IV&V (9,20) | Independent Verification and Validation |
| IVT (18) | Interrupt Vector Table |
| JPL (5) | Jet Propulsion Laboratories |
| k (6,20) | Thousand |
| kA (13,16) | Kiloampere |
| KBAP (20) | Knowledge-Based AutoPilot |
| kHz (5,6,12,14,19) | Kilohertz |
| km (6) | kilometer |
| $\hat{L}$ (17) | Estimated Program Level |
| L (17) | Program Level |
| LAN (5) | Local Area Network |
| LaRC (5) | Langley Research Center |
| LC (19) | Liquid Crystal |
| LCC (11,12) | Life Cycle Cost |
| LCD (16) | Liquid Crystal Display |
| LCS (19) | Liquid Crystal Shutter |
| LED (6,19,20) | Light Emitting Diode |
| LF (13) | Low Frequency |
| LISP (20) | LISt Processor |
| LOC (11,12) | Localizer |
| LOFT (19) | Line Oriented Flight Training |
| LPN (13) | Lumped Parameter Network |
| LRC (6) | Longitudinal Redundancy Check |
| LRRA (11,12,19) | Low Range Radio Altimeter |
| LRU (5,6,11,12,13,18,19) | Line Replaceable Unit |
| LSB (6,18) | Least Significant Bit |
| LSI (5,18,19) | Large Scale Integration |
| LTPB (6,18) | Linear Token Passing Bus |
| LTRI (13) | Lightning and Transients Research Institute |
| LVDT (15) | Linear Variable Differential Transformers |
| LVDT (3) | Linear Voltage Differential Transducer |
| m (18) | Original Address of Last Transmission |
| m (6) | meter |
| M (6) | Million |

| | |
|---|---|
| M (5) | Mutual (when used with RLC) |
| mA (3) | Milliampere |
| MAADS (6) | Multibus Avionic Architecture Design Study |
| MAC (4) | Mean Aerodynamic Chord |
| MAFT (16) | Multicomputer Architecture for Fault Tolerance |
| Mbps (6,16) | Million bytes per second |
| MC (18) | Mode Code |
| MCDP (11,12) | Maintenance Control and Display Panel |
| MCDU (19) | Multi-Purpose Control Display Unit |
| MCFCS (18) | Message Control Frame Check Sequence |
| MCP (11,12) | Mode Control Panel |
| MDICU (4,5) | Modular Digital Interface Conversion Unit |
| MDICU (3) | Modular Digital Interface Control Unit |
| MDT (6) | Mean Down Time |
| MEP (20) | Mission Equipment Package |
| MFCS (18) | Message Frame Check Sequence |
| Mflops (16) | Million floating-point operations per second |
| MFM (6) | Modified-Frequency Modulation |
| MFR (10) | Mean Failure Rate |
| MHz (5,6,12,14,18,20) | Megahertz |
| mil (11,12) | One thousandths of an inch (0.001) |
| MIL (5) | Military |
| MIL-HDBK (18) | Military Handbook |
| MIL-STD (6,18) | Military Standard |
| MKR (19) | Markers |
| ML (18) | Message Length |
| MLE (8) | Maximum Likelihood Estimates |
| MLS (11,12,19) | Microwave Landing System |
| mm (19,20) | Millimeter |
| Mn (19) | Manganese |
| MOS (5) | Metal-Oxide Semiconductor |
| MPP (16) | Massively Parallel Processor |
| MPSC (18) | Multi-Protocol Serial Controller |
| MPX (5) | Multiplex |
| ms (6,10,18) | Millisecond |
| MSB (6,18) | Most Significant Bit |
| MSE (11,12) | Magnetic (Field) Shielding Effectiveness |
| MSI (5) | Medium Scale Integration |
| MT (18) | Message Time |
| MTBCF (6) | Mean Time Between Critical Failures |
| MTBF (9,18) | Mean Time Between Failures |
| MTTF (8,10) | Mean Time to Failure |
| MTTR (6,18) | Mean Time To Repair |
| Mux (5) | Multiplexed |
| MUX (4,18) | Multiplexer |
| | |
| $\hat{N}$ (17) | Estimated Length |
| n (18) | Address of User Performing Computation |
| N (18) | Maximum Number of Users |
| N (17) | Implementation Length of a Program |
| $N_1$ (15) | Low Rotor Speed |
| $N_1$ (17) | Total Number of Operator Occurrences |

| | |
|---|---|
| $N_1^*$ (17) | Minimum Number of Operators |
| $N_2$ (15) | High Rotor Speed |
| N2 (11,12) | Core Engine Speed |
| $N_2$ (17) | Total Number of Operand Occurrences |
| $N_2^*$ (17) | Minimum Number of Operands |
| NA (6) | Numerical Acceptance |
| NA (3) | Normal Accelerometers |
| NADC (6,7) | Naval Air Development Center |
| Naecon (5) | National Avionics and Electronics Conference |
| NAECON (6) | National Aerospace and Electronics Conference |
| NAND (5) | Not AND |
| NAS (19) | National Airspace System |
| NASA (2,3,5,7,13,15,18,19,20) | National Aeronautics and Space Administration |
| NASC (13) | Naval Air Systems Command |
| NATO (14) | North Atlantic Treaty Organization |
| NB (11,12) | Narrow Band Signal |
| NCTS (18) | Not Clear To Send |
| ND (19) | Navigation Display |
| NEC (13) | Numerical Electromagnetics Code |
| NEMP (5,12) | Nuclear Electromagnetic Pulse |
| NHPP (8) | Non-Homogeneous Poisson Process |
| nmi (14) | nautical mile |
| NN (20) | Neural Network |
| NPRM (14) | Notice of Proposed Rulemaking |
| NRZ (6) | Non-return to Zero |
| NRZ-I (6) | Non-return to Zero Inverted |
| NRZ-L (6) | Non-return to Zero Dual Level |
| nsec (6) | Nanosecond |
| NSWC (13) | Naval Surface Weapons Center |
| NTSB (19) | National Transportation Safety Board |
| NVG (19) | Night Vision Goggles |
| NVS (8) | N-version Software |
| | |
| OMEGA (11,12) | Very Low Frequency Navigation |
| OMV (16) | Orbital Maneuvering Vehicle |
| ONR (20) | Office of Naval Research |
| OS (5) | Operating System |
| OSI (18) | Open Systems Interconnection |
| OTV (16) | Orbital Transfer Vehicle |
| | |
| P-Static (11,12) | Precipitation Static |
| P (10) | Processor |
| PA (19) | Public Address |
| PA (20) | Pilot's Associate |
| PAI (19) | Primary Attitude Instrument |
| PAL (5) | Programmable Array Logic |
| PAS (6) | Pilot Assist System |
| PAWS (5) | Padé Approximation With Scaling |
| PBW (15) | Power-By-Wire |
| PC (17) | Processing Complexity |
| PC (5,20) | Personal Computer |
| PCA (17) | Processing Complexity Adjustment |

| | |
|---|---|
| PCS (16) | Primary Control System |
| PCU (11,12) | Power Control Unit |
| pdf (8) | Probability Density Function |
| PE (6) | Phase Encoding |
| PE (20) | Processing Element |
| pf (12) | picofarad |
| PFD (19) | Primary Flight Display |
| PGA (20) | pin grid array |
| PLA (15) | Power Level Actuator |
| PLA (16) | Power Level Angle |
| PMA (18) | Parts Manufacturer Approval |
| PMA (20) | Portable Maintenance Aid |
| PMS (5) | Physical Message Switch |
| PNF (19) | Pilot Not Flying |
| PNI (19) | Primary Navigation Instrument |
| PRF (11,12) | Pulse Repetition Frequency |
| PROC (5) | Processor |
| PROLOG (20) | PROgramming in LOGic |
| PROM (3,10,17,18) | Programmable Read-Only Memory |
| psi (15) | pounds per square inch |
| PVI (16,19) | Pilot-Vehicle Interface |
| PWM (11,12) | Pulse Width Modulation |
| PZ (15) | Piezoelectric |
| | |
| QUAD (5) | Quadruple |
| | |
| R (13) | Resistance |
| R-C (12) | Resistor-Capacitor |
| RADC (8,17) | Rome Air Development Center |
| RAE (13) | Royal Aircraft Establishment |
| RAE (20) | Royal Aerospace Establishment |
| RAM (3,5,10,16,17,18) | Random Access Memory |
| RAT (6,18) | Ring Admittance Timer |
| RB (8) | Recovery Block |
| RBDCP (3) | Reliability Block Diagram Computer Program |
| RCA (16) | Radio Corporation of America |
| RCP (19) | Reconfigurable Cockpit |
| RCVR (5) | Receiver |
| RDFCS (3) | Redundant Digital Flight Control System |
| RDFCS (4) | Reconfigurable Digital Flight Control System (facility) |
| RDFCS (5) | Reconfigurable Digital Flight Control System |
| RDMI (11,12) | Radio Distance Magnetic Indicator |
| RE (11,12) | Radiated Emission |
| REG (5) | Register |
| REL (3) | Reliability |
| REL COMP (3) | Reliability Computers |
| RF (5,11,12,13,14,16,18) | Radio Frequency |
| RFI (11,12) | Radio Frequency Interference |
| RIM (18) | Ring Interface Module |
| RISC (20) | Reduced Instruction Set Computer |
| RIU (18) | Ring Interface Unit |

| | |
|---|---|
| RL (13,15) | Resistance/Inductance (Out of order in c.15) |
| $R_L$ (19) | Load Resistor |
| RLC (5,13) | Resistance/Inductance/Capacitance |
| RLCM (13) | Resistance/Inductance/Capacitance/Mutual |
| RM (7) | Redundancy Management |
| RNRZ (6) | Randomized Non-return to Zero |
| ROM (5,10,19) | Read Only Memory |
| RPA (20) | Rotorcraft Pilot's Associate |
| RPV (16) | Remotely Piloted Vehicle |
| RR (18) | Read Register |
| RRT (18) | Ring Rotation Time |
| RS (18) | Recommended Standard |
| RS (11,12) | Radiated Susceptibility |
| RSS (4,7) | Relaxed Static Stability |
| RT (5,18) | Remote Terminal |
| RTCA (2,3,11,12,14, 16,17,18,20) | Radio Technical Commission for Aeronautics (now known as Requirements and Technical Concepts for Aviation) |
| RTE (18) | Real-Time Executive |
| RTI (5) | Remote Terminal Interface |
| RTS (18) | Request To Send |
| RZ (6) | Return to Zero |
| S (17) | Stroud Number |
| S/A (11,12) | Spectrum Analyzer |
| S-a-0 (5,10) | Stuck at Zero |
| S-a-1 (5,10) | Stuck at One |
| S-GLOSS (5) | Gate Logic Software Simulator developed by Stevens |
| SAE (2,5,6,13,14,16, 17,18,19,20) | Society of Automotive Engineers (now known as Engineering Society for Advanced Mobility Land Sea Air and Space) |
| SAI (18) | Systems Architecture and Interfaces |
| SAS (4,7) | Stability Augmentation System |
| SATCOM (19) | Satellite Communications |
| SATNAV (19) | Satellite Navigation |
| SC (18) | Special Condition |
| SC (20) | Special Committee |
| SCC (18) | System Configuration Controller |
| SCM (18) | Software Configuration Management |
| SCP (18) | Self-Checking Pair |
| SD (19) | System Display |
| SDF (7) | Super-Diagnostic Filter |
| SE (11,12) | Shielding Effectiveness |
| SELCAL (19) | Select Call |
| SEU (5) | Single Event Upset |
| SG (18) | Synchronization Gap |
| SHF (11,12) | Super High-Frequency |
| SHRD (5) | Shared |
| SIAM (15) | Society for Industrial and Applied Mathematics |

| | |
|---|---|
| SIF (14) | Selective Identification Facility |
| SIM (18) | Serial Interface Module |
| SIR (18) | Shared Interface RAM |
| SLRT (7) | Sequential Likelihood Ratio Test |
| SMF (18) | Self Monitor Function |
| SMOTEC (14) | Special Missions Operation Test and Evaluation Center |
| SO (17) | Second Order (Average) |
| SOP (19) | Standard Operating Procedure |
| SPRT (7) | Sequential Probability Ratio Test |
| SQA (18) | Software Quality Assurance |
| SQF (17) | Software Quality Factor |
| SQL (5) | Software Query Language |
| SQM (17) | Software Quality Metrics |
| SQPP (17) | Software Quality Program Plan |
| SRS (17) | Software Requirement Specification |
| SSA (18) | System Safety Assessment |
| SSI (5) | Small Scale Integration |
| SSP (3) | Servo Simulation Panel |
| SSPC (15) | Solid-State Power Controller |
| SSS (17) | System/Segment Specification |
| STANAG (14) | Standardization Agreement (NATO) |
| STC (2) | Supplemental Type Certification |
| STC (18,20) | Supplemental Type Certificate |
| STEM (5) | Scaled Taylor Expansion Matrix |
| STN (19) | Supertwisted Nematic |
| STOL (16) | Short Takeoff and Landing |
| str (6) | string |
| SURE (5) | Semi-Markov Unreliability Range Evaluator |
| SW (18) | Software |
| SYN (5) | Synch |
| $\hat{T}$ (17) | Estimated Programming Time |
| T/R (6) | Transmitter/Receiver |
| TACAN (14,19) | Tactical Air Navigation |
| TASRA (3) | Tree Aided System Reliability Analysis |
| $T_C$ (18) | Count Duration |
| TC (2) | Type Certification |
| TC (18,20) | Type Certificate |
| TCAP (13) | Threshold Circuit Analysis Program |
| TCAS (11,12,16,18) | Traffic Alert and Collision Avoidance System |
| TCL (13) | Transient Control Level |
| TDM (6) | Time Division Multiplex |
| TDMA (18) | Time Division Multiple Access |
| $T_F$ (18) | Frame Time |
| TEAMS (20) | Technical Expert Aircraft Maintenance System |
| TFCS (18) | Token Frame Check Sequence |
| TFEDF (18) | Token Frame Ending Delimiter Field |
| TFEL (19) | Thin-Film Electroluminescence |
| TFT (19) | Thin-Film Transistor |
| TG (18) | Terminal Gap |
| THT (6,18) | Token-Holding Timer |

| | |
|---|---|
| TI (18) | Transmit Interval |
| TLA (11,12) | Thrust Lever Angle |
| $T_m$ (18) | Wait Time for User |
| TMC (11,12) | Thrust Management Computer |
| TMR (10) | Triple Modular Redundant |
| TN (19) | Twisted Nematic |
| TRT (18) | Token Rotation Timer |
| TRU (15) | Transformer Rectifier Unit |
| TSDF (18) | Token Starting Delimiter Field |
| TSO (18,19,20) | Technical Standard Order |
| TTL (5,11,12,13,16,19,20) | Transistor-Transistor Logic |
| TV (14) | Television |
| TWTD (13) | Thin Wire Time Domain |
| TX (5) | Transmit |
| | |
| U.K. (13,14) | United Kingdom |
| U.S. (14) | United States |
| UAC (18) | Unbalanced Asynchronous Configuration |
| UART (15) | Universal Asynchronous Receiver Transmitter |
| UHF (11,12,13,14,19) | Ultra High-Frequency |
| UNC (18) | Unbalanced Normal Configuration |
| UNIBUS (5) | Universal Bus |
| UPS (15) | Uninterruptible Power Supplies |
| USAF (16,20) | United States Air Force |
| USB (16) | Upper Surface Blowing |
| USEG (5) | Unsegmented |
| UUT (20) | Unit Under Test |
| UV (19) | Ultra-Violet |
| | |
| V (17) | Volume |
| V (19) | Volt |
| V/m (14) | Volt/meter |
| $V^*$ (17) | Potential Volume |
| V&V (18,20) | Verification and Validation |
| $V_B$ (19) | Breakdown voltage |
| $V_{CC}$ (19) | Collector Circuit Voltage |
| $V_F$ (19) | Forward Voltage |
| VFD (19) | Vacuum Florescent Display |
| VHF (11,12,13,14,19) | Very High-Frequency |
| VHSIC (6,16) | Very-High-Speed Integrated Circuits |
| VLF (11,12,19) | Very Low-Frequency |
| VLSI (5,6,14,18,20) | Very Large Scale Integration |
| VLSIC (6,16) | Very Large Scale Integrated Circuits |
| VOR (11,12,14,18,19) | VHF Omnidirectional Range |
| VORTA/VHF (11,12) | Omnirange/Tactical Air Navigation |
| VRC (6) | Vertical Redundancy Check |
| $V_S$ (19) | Sustain-Voltage |
| VSI (11,12) | Vertical Speed Indicator |
| VSV (15) | Variable Stator Vane |
| VT (18) | Validation Testing |
| VTOL (16) | Vertical Takeoff and Landing |

| | |
|---|---|
| W/P (15) | Fuel Flow to Burner Pressure |
| WAI (3) | Warning Annunciation Indicator |
| WFM (15) | Main fuel metering valve actuator sensor |
| WR (18) | Write Register |
| WRU (11,12) | Weapons Replaceable Unit |
| WSO (17) | Weighted Second Order (Average) |
| | |
| XAB (5) | Transmit Compare A B |
| XMT (5) | Transmit |
| XMTR (3) | Transmitter |
| XOR (5) | Exclusive OR |
| | |
| ZM (6) | Zero Modulation |
| ZnS (19) | Zinc Sulfide |
| ZnSe (19) | Zinc Selenide |

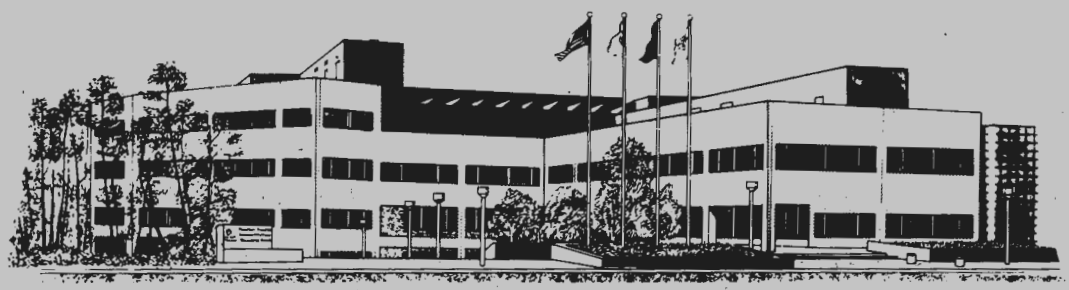
*U.S. GOVERNMENT PRINTING OFFICE: 1994-504-078-00146

U.S. Department of Transportation
**Federal Aviation Administration**

DOT/FAA/CT-88/10

# HANDBOOK-VOLUME II
# DIGITAL SYSTEMS VALIDATION

## CHAPTER 1
## SUMMARY



**PREPARED BY:**

**COMPUTER RESOURCE MANAGEMENT, INC.**
**950 HERNDON PARKWAY, SUITE 360**
**HERNDON, VIRGINIA 22070**

**PREPARED FOR:**

**FEDERAL AVIATION ADMINISTRATION**
**TECHNICAL CENTER**
**ATLANTIC CITY INTERNATIONAL AIRPORT, NEW JERSEY 08405**

NOTICE

This document is disseminated under the sponsorship of the U.S. Department of Transportation in the interest of information exchange. The United States Government assumes no liability for the contents or use thereof.

The United States Government does not endorse products or manufacturers. Trade or manufacturers' names appear herein solely because they are considered essential to the objective of this report.

# EXECUTIVE SUMMARY

The Digital Systems Validation Handbook - Volume II (DOT/FAA/CT-88/10) is a followup and companion to the Handbook - Volume I Validation of Digital Systems in Avionics and Flight Control Applications (DOT/FAA/CT-82/115).

Volume I was issued in July 1983 and revised in September 1986. It deals with system overview topics such as the System Life Cycle, Mission Factors, System Architectures, Crew Workload Evaluation, Current Validation Procedures, Recommended Validation Procedures, and Recommended Configuration Management Procedures.

The emphasis in Volume I is on the validation of digital avionic systems. Validation of a system is the determination that the system is well grounded on principles or evidence. The topics presented in Volume I assist the certification engineer in understanding how to determine if the validation of a system has been correctly performed.

The emphasis in Volume II is on the certification of advanced technology digital avionic systems, as well as their validation. Volume II covers detailed technical topics such as latent faults; data busses; integrated assurance assessment; analytical sensor redundancy; and protection against lightning, electromagnetic interference, and high energy radio frequency fields. These topics are covered in detail to familiarize the certification engineer with the issues involved in implementing the new technologies.

Volume II covers topics that will enable the certification engineer to understand the information presented in type certification and supplemental type certification documentation, to understand variations in the implementation of technologies, and to discuss them with the design engineer.

Volume II also addresses some of the soon-to-be-available technologies in the "Advanced Validation Issues" chapter. The direction of aviation research in the United States is discussed along with challenges and problems that confront the certification engineer in certifying the new technologies.

Since the topics discussed in this Handbook are at the forefront of technological research, some of the concepts presented are subject to discussion by experts in the field. In these areas, the Handbook presents various viewpoints alerting the certification engineer to the various views so that this information will be considered in formulating decisions and developing certification criteria.

It is intended that this Handbook will be a living document. As technology advances, addenda to existing chapters will be issued. Where major new technologies are addressed, new chapters will be added to the Handbook. The Federal Aviation Administration (FAA), therefore, solicits your comments and

suggestions for the improvement of this Handbook. Please address any comments to:

Flight Safety Research Branch, ACD-230
Digital Systems Validation Handbook
FAA Technical Center, Bldg. 201
Atlantic City International Airport, New Jersey 08405

U.S. Department of Transportation
Federal Aviation Administration

DOT/FAA/CT-88/10

# HANDBOOK-VOLUME II
# DIGITAL SYSTEMS VALIDATION

## CHAPTER 2
## INTRODUCTION

**PREPARED BY:**

**COMPUTER RESOURCE MANAGEMENT, INC.**
**950 HERNDON PARKWAY, SUITE 360**
**HERNDON, VIRGINIA 22070**

**PREPARED FOR:**

**FEDERAL AVIATION ADMINISTRATION**
**TECHNICAL CENTER**
**ATLANTIC CITY INTERNATIONAL AIRPORT, NEW JERSEY 08405**

## NOTICE

This document is disseminated under the sponsorship of the U.S. Department of Transportation in the interest of information exchange. The United States Government assumes no liability for the contents or use thereof.

The United States Government does not endorse products or manufacturers. Trade or manufacturers' names appear herein solely because they are considered essential to the objective of this report.

TABLE OF CONTENTS


| Section | | Page |
|---|---|---|
| 1. | INTRODUCTION | 2-1 |
| 1.1. | Background | 2-1 |
| 1.2. | Purpose | 2-2 |
| 1.3. | Scope | 2-2 |
| 2. | HANDBOOK ORGANIZATION | 2-3 |
| 2.1. | Handbook Format | 2-3 |
| 2.2. | Chapter Formats | 2-3 |
| 2.3. | Contents of Technical Chapters | 2-3 |
| 2.3.1. | Chapter 3 - Integrated Assurance Assessment Activities | 2-3 |
| 2.3.2. | Chapter 4 - Quadruplex Digital Flight Control System Assurance Assessment | 2-3 |
| 2.3.3. | Chapter 5 - Advanced Fault Insertion & Simulation Methods | 2-3 |
| 2.3.4. | Chapter 6 - Data Bus Integrity Issues | 2-4 |
| 2.3.5. | Chapter 7 - Analytical Sensor Redundancy | 2-4 |
| 2.3.6. | Chapter 8 - Software Reliability Assessment Methods | 2-4 |
| 2.3.7. | Chapter 9 - Software Fault Tolerance | 2-4 |
| 2.3.8. | Chapter 10 - Latent Faults | 2-4 |
| 2.3.9. | Chapter 11 - Guidelines To Assess EMC Designs | 2-5 |
| 2.3.10. | Chapter 12 - Fast Rise Time EMC Transient Testing | 2-5 |
| 2.3.11. | Chapter 13 - Lightning Studies | 2-5 |
| 2.3.12. | Chapter 14 - High Energy Radio Frequency Fields | 2-5 |
| 2.3.13. | Chapter 15 - Electromechanical Actuator Systems (EMAS) | 2-5 |
| 2.3.14. | Chapter 16 - Advanced Validation Issues | 2-5 |
| ACRONYMS AND ABBREVIATIONS | | 2-7 |

# 1. INTRODUCTION

## 1.1. Background

The rapid developments in computers and digital technology over the past 20 years have made digital systems increasingly useful in aircraft. As computers become more sophisticated, they are able to perform an increasing number of tasks which were previously performed by the pilot or by analog systems. As a result, certification engineers are seeing increasing numbers of digital systems proposed for inclusion in new generations of aircraft and for incorporation into existing aircraft designs.

The new systems pose a number of problems for the certification engineer. Error limits must be set for flight-critical, flight-essential, and non-essential systems. In setting error limits, safety is paramount, but other factors must also be taken into account. These factors include cost for incremental error reduction, weight penalties, and system interactions.

Integrating a new digital "black box" into an existing design changes many of the electromagnetic characteristics of the aircraft. Care must be taken to ensure not only that the black box performs its functions within the required error limits but also to ensure that its addition does not degrade any other systems which have already been certificated.

The design of a completely new aircraft poses its own problems in digital technology. Each individual component must meet error limits, weight limits, and size limits. Systems (composed of several components more or less collocated) must meet these requirements, plus the requirements of being able to collect systemic information and pass it on to other systems while not interfering with the operation of any of the other systems.

In the summer of 1975, the Federal Aviation Administration (FAA) and the National Aeronautics and Space Administration (NASA) began to consider and plan a joint program to assess and upgrade the technologies used to evaluate the reliability of digital flight control and avionics systems. In 1976, the FAA and NASA cosponsored a workshop on digital flight controls and avionics.

A number of reports emerged from this program. These reports were collected into the Digital Systems Validation Handbook - Volume I (DOT/FAA/CT-82/115) "Validation of Digital Systems in Avionics and Flight Control Applications." Since the 1976 workshop and the 1982 publication of Volume I, however, a number of technological advances have been made. It is time to review the issues of digital avionics and flight control systems in light of the new, emerging technologies.

There are approximately 20 areas of technological investigation that have been sponsored by the FAA and/or NASA. The investigations, the final technical

reports, and the technical results are the basis for the tutorial materials in this Handbook.

## 1.2. Purpose

The purpose of this Handbook is to give the certification engineer background information on the emerging technologies which will figure prominently in designs now being submitted for certification and anticipated by the end of the century. Certification engineers will then be more knowledgeable in these new areas of technology so that they may more effectively evaluate the materials submitted in support of a request for a Type Certificate (TC) or Supplemental Type Certificate (STC).

This Handbook will serve as a guide to the certification engineer. Each chapter addresses a specific area of technology. Alternate viewpoints are presented as appropriate. Worked examples guide the engineer through the application of the technology. Bibliographies in each chapter give the engineer sources of additional information about the chapter topic. Each chapter can be considered a stand-alone volume about the subject area in question and can be separated into its own volume for ease of reference.

## 1.3. Scope

This Handbook provides the certification engineer with information on emerging technologies. Since much of the technology in this Handbook is new, some of it is still at issue and is subject to various interpretations. For completeness, alternate viewpoints have been included wherever appropriate. This will provide the certification engineer with the widest variety of information available from which to make certification decisions.

The user should also note that although the Handbook contains the latest information available at the time each tutorial was written, some of this information may be out of date. The certification engineer should therefore use this Handbook as a guideline rather than a hardline.

# 2. HANDBOOK ORGANIZATION

## 2.1. Handbook Format

This Handbook is a dynamic document. As such, it is being issued in a loose-leaf format so that it can be expanded easily as knowledge of newer technology applications becomes available. Each chapter has the same type of tutorial/ worked example format and will be updated as a function of technological progress in the areas of civil fixed or rotary wing aircraft, electronics, and system integration. As new information is received, it will be issued either in the form of chapters to be added or revision pages for existing chapters.

The Glossary and the Acronym list for the entire book contain information from all chapters.

## 2.2. Chapter Formats

Each chapter contains its own Table of Contents, List of Acronyms and Abbreviations, Glossary, and References. The individual acronym lists are particularly important when dealing with such diverse topics since different subject areas may use common acronyms with different interpretations.

## 2.3. Contents of Technical Chapters

### 2.3.1. Chapter 3 - Integrated Assurance Assessment Activities

This chapter covers Integrated Assurance Assessment methods including probability analysis, fault-tree analysis, and failure modes and effects analysis. It shows how these methods may be meshed to establish the air-worthiness of a flight-critical system. It also demonstrates how these methods reinforce each other in this determination.

### 2.3.2. Chapter 4 - Quadruplex Digital Flight Control System Assurance Assessment

Quadruplex digital flight control systems are an evolution from the dual-dual architecture, using four parallel systems rather than two redundant systems. This chapter covers the history of the technology and identifies the functional relationships between the hardware and the software.

### 2.3.3. Chapter 5 - Advanced Fault Insertion & Simulation Methods

One method of testing flight-critical systems is to insert faults into the system to see if they are found by the error detection and correction mechanisms embedded in the system. There are three categories of faults: (1) faults which are inserted and found, (2) faults which are inserted and not found, and (3) no-fault situations which are identified as faults. This chapter covers the

hardware systems used in the insertion of faults and the interpretation of the results.

2.3.4. Chapter 6 - Data Bus Integrity Issues

Data buses are currently used in flight control and avionics applications to transfer data. Errors in data transmission may result in erroneous or incomprehensible data being transmitted to processors or to flight control mechanisms. This chapter covers the integrity of data buses. It contains an overview of alternative data bus architectures, including, an evaluation of their reliability. It also addresses areas of uncertainty in reliability and minimum currently accepted redundancy standards.

2.3.5. Chapter 7 - Analytical Sensor Redundancy

Analytical sensor redundancy is the use of software algorithms to substitute for hardware in providing redundant signals. If an accurate model of the airframe can be created, the software can extrapolate from prior observations and make short-term predictions about the status of the aircraft. This allows the system to compensate for hardware failure. This chapter covers the details of analytical sensor redundancy, its uses, and its limitations.

2.3.6. Chapter 8 - Software Reliability Assessment Methods

As flight functions are increasingly carried out by digital systems, the issue of the reliability of the software controlling these systems becomes more and more important to flight safety. Higher order languages, testing, debugging, and software engineering can all contribute to increased reliability. Models have been developed to assess the reliability of these systems. This chapter covers existing software reliability assessment models. It compares the models and gives an overview of their reliability as predictive tools.

2.3.7. Chapter 9 - Software Fault Tolerance

This chapter covers software methods and techniques used for detection and correction of faults. These may include checks on the reasonableness of data and comparison of information from redundant systems. This chapter also covers fault tolerant design using complexity measures and other techniques.

2.3.8. Chapter 10 - Latent Faults

Latent faults are faults which have occurred, but have not been detected because the path on which they have occurred has not been traveled. They can be both hardware and software faults. This chapter covers experiments conducted to inject faults and discover them. It discusses the prediction, triggering, identification, and correction of latent faults.

2.3.9. Chapter 11 - Guidelines To Assess EMC Designs

This chapter covers guidelines that may be used to assure electromagnetic compatibility during the design process. It also includes information on how

the certification engineer can assess how well a design or system adheres to the guidelines.

2.3.10. Chapter 12 - Fast Rise Time EMC Transient Testing

This chapter discusses EMC transient testing experiments. Presently, the chapter contains the theory for the testing only, since the testing itself has not been conducted and may not be completed in time for this handbook.

2.3.11. Chapter 13 - Lightning Studies

As more and more flight-critical systems are digitized and as composite materials become more widely used in aircraft construction, the threat from lightning increases. This chapter covers currently accepted information about lightning including estimates of currents, electric fields, and magnetic fields which aircraft might expect to encounter. It reviews methods for protecting electrical and electronic equipment from this hazard and it covers methods for isolating equipment so that potential damage may be limited.

2.3.12. Chapter 14 - High Energy Radio Frequency Fields

With changes in aircraft construction from aluminum to composites and increased criticality of software-based digital flight control and avionic systems, total aircraft RF susceptibility hardening and protection is vital. This chapter assesses the present criteria, specifications, and procedures for industry and certification authorities in the light of increasing levels of RF over a wide frequency range.

2.3.13. Chapter 15 - Electromechanical Actuator Systems (EMAS)

This chapter will cover the technical description, testing, and redundancy requirements for smart servos and actuators, primary and secondary power sources, and digital engine controllers.

2.3.14. Chapter 16 - Advanced Validation Issues

This chapter gives a brief overview of areas where advances are likely in the near future. As new information becomes available, each of these sections may be expanded to form a separate chapter.

The topics to be covered are:

- Artificial Intelligence
- Remote Processing
- Robotics (Testing Control)
- Advanced Architecture (Distributed Fault Tolerant Processing)
- Advanced Controls and Displays

- Voice Command Processing
- System Integration Technology

## ACRONYMS AND ABBREVIATIONS

| | |
|---|---|
| CRMI | Computer Resource Management, Incorporated |
| CT | Technical Center Identification code |
| DOT | Department of Transportation |
| EMAS | Electromechanical Actuator System |
| FAA | Federal Aviation Administration |
| NASA | National Aeronautics and Space Administration |
| RTCA | Radio Technical Commission for Aeronautics |
| SAE | Society of Automotive Engineers |
| STC | Supplemental Type Certification |
| TC | Type Certification |

U.S. Department of Transportation
Federal Aviation Administration

DOT/FAA/CT-88/10

# HANDBOOK-VOLUME II
# DIGITAL SYSTEMS VALIDATION

## CHAPTER 3
## INTEGRATED ASSURANCE ASSESSMENT

PREPARED BY:

COMPUTER RESOURCE MANAGEMENT, INC.
950 HERNDON PARKWAY, SUITE 360
HERNDON, VIRGINIA 22070

PREPARED FOR:

FEDERAL AVIATION ADMINISTRATION
TECHNICAL CENTER
ATLANTIC CITY INTERNATIONAL AIRPORT, NEW JERSEY 08405

## NOTICE

This document is disseminated under the sponsorship of the U.S. Department of Transportation in the interest of information exchange. The United States Government assumes no liability for the contents or use thereof.

The United States Government does not endorse products or manufacturers. Trade or manufacturers' names appear herein solely because they are considered essential to the objective of this report.

TABLE OF CONTENTS


| Section | | Page |
|---|---|---|
| 1. | INTRODUCTION TO INTEGRATED ASSURANCE ASSESSMENT | 3-1 |
| | 1.1. Justification for Integrated Assurance Assessment | 3-1 |
| | 1.2. Primary Source for this Tutorial | 3-2 |
| | 1.3. How to Use this Document | 3-2 |
| 2. | IAA OF A RECONFIGURABLE DIGITAL FLIGHT CONTROL SYSTEM | 3-5 |
| | 2.1. The Redundant Digital Flight Control System | 3-5 |
| | 2.2. The IAA Study | 3-11 |
| | 2.3. Conclusions and Observations | 3-11 |
| 3. | CONSIDERATIONS IN THE DEVELOPMENT OF AN IAA APPROACH | 3-15 |
| | 3.1. General Comments | 3-15 |
| | 3.2. An IAA Approach | 3-15 |
| | 3.3. Fault Tree Role in Integrated Assurance | 3-16 |
| | 3.4. Failure Mode and Effect Analysis Role in Integrated Assurance | 3-17 |
| | 3.5. Reliability Analysis Role in Integrated Assurance | 3-18 |
| | 3.6. Fault Insertion Role in Integrated Approach | 3-19 |
| 4. | FAULT TREE ANALYSIS | 3-21 |
| | 4.1. General Comments | 3-21 |
| | 4.2. Fault Tree Development | 3-21 |
| | 4.3. Quantitative Fault Tree Analysis | 3-37 |
| 5. | FAILURE MODE AND EFFECT ANALYSIS | 3-43 |
| | 5.1. General Comments | 3-43 |
| | 5.2. Application of FMEA to RDFCS | 3-44 |
| 6. | FAULT INSERTION | 3-49 |
| | 6.1. General Comments | 3-49 |
| | 6.2. Application of Fault Insertion to RDFCS | 3-49 |
| | 6.3. Fault Insertion Results | 3-56 |

TABLE OF CONTENTS (Continued)


| Section | | Page |
|---|---|---|
| 7. | FAILURE RATE DEVELOPMENT | 3-57 |
| | 7.1. General Comments | 3-57 |
| | 7.2. Rationale for Developing Failure Rates for RDFCS | 3-57 |
| 8. | RELIABILITY PREDICTION USING CARSRA | 3-63 |
| | 8.1. General Comments | 3-63 |
| | 8.2. RDFCS Reliability - Fault Tree Analysis | 3-63 |
| | 8.3. RDFCS Reliability - CARSRA | 3-64 |
| | 8.4. CARSRA Application | 3-64 |
| | 8.4.1. Configuration Description | 3-64 |
| | 8.4.2. Markov Models | 3-64 |
| | 8.4.3. Dependencies | 3-67 |
| | 8.4.4. Availability | 3-69 |
| 9. | SUMMARY | 3-79 |
| | 9.1. Background | 3-79 |
| | 9.2. Concluding Remarks | 3-79 |
| | 9.3. Analysis Enhancements | 3-80 |
| APPENDICES | | |
| | A - SUMMARY DESCRIPTION FOR REDUNDANT DIGITAL FLIGHT | 3-83 |
| | B - DESCRIPTION OF RDFCS SIMULATOR | 3-85 |
| | C - FAILURE MODES AND EFFECTS ANALYSIS RESULTS | 3-89 |
| | D - PROCESSOR SCHEMATIC DIAGRAMS | 3-101 |
| | E - RESULTS OF FAULT INSERTION | 3-107 |
| BIBLIOGRAPHY | | 3-123 |
| ACRONYMS AND ABBREVIATIONS | | 3-125 |

LIST OF ILLUSTRATIONS


| Figure | | Page |
|---|---|---|
| 2.1-1 | RDFCS DUAL-DUAL CONFIGURATION | 3-6 |
| 2.1-2 | RDFCS SIMULATOR | 3-7 |
| 2.1-3 | CTA AND COMPUTER BREAKOUT PANEL | 3-8 |
| 2.1-4 | SERVO SIMULATOR PANEL | 3-9 |
| 2.1-5 | DISCRETE SWITCH PANEL | 3-10 |
| 4.2-1 | FAULT TREE TOP LEVEL | 3-24 |
| 4.2-2 | SENSING FUNCTION | 3-25 |
| 4.2-3 | NORMAL ACCELERATION SENSING | 3-26 |
| 4.2-4 | NO DUAL ANNUNCIATION | 3-29 |
| 4.2-5 | COMPUTING FUNCTION | 3-31 |
| 4.2-6 | CHANNEL 1A RUDDER COMMAND | 3-32 |
| 4.2-7 | FCC PROCESSOR CONTROL CARD | 3-33 |
| 4.2-8 | FCC PROCESSOR DATA PATH CARD | 3-34 |
| 4.2-9 | YAW AUTOPILOT SERVO COMMAND WARNING | 3-35 |
| 4.2-10 | SERVO FUNCTIONS | 3-39 |
| 4.2-11 | NO. 1 YAW AUTOPILOT SERVO | 3-40 |
| 4.2-12 | MULTIPLE FAILURES DURING CRUCIAL PHASE | 3-41 |
| 5.1-1 | ONE OUTPUT, TWO INPUT CONDITIONS | 3-46 |
| 5.1-2 | TWO OUTPUT, ONE INPUT CONDITION | 3-47 |
| 5.2-1 | PROCESSOR BLOCK DIAGRAM | 3-48 |
| 6.2-1 | CTA AND COMPUTER BREAKOUT PANEL | 3-50 |
| 6.2-2 | SERVO SIMULATOR PANEL | 3-51 |

## LIST OF ILLUSTRATIONS (Continued)


| Figure | | Page |
|---|---|---|
| 6.2-3 | DISCRETE SWITCH PANEL | 3-52 |
| 6.2-4 | FCC WITH FLIGHT CARD EXTENDED | 3-54 |
| 6.2-5 | FCC PROCESSOR DATA PATH CARD | 3-55 |
| 8.4-1 | MARKOV MODEL OF DUAL STAGE | 3-66 |
| 8.4-2 | MARKOV MODEL CODING FOR SENSOR STAGES | 3-71 |
| 8.4-3 | CARSRA INPUT (1 of 6) | 3-72 |


## LIST OF TABLES


| Table | | Page |
|---|---|---|
| 1.1-1 | INTEGRATED ASSURANCE ASSESSMENT MODEL | 3-2 |
| 2.2-1 | ASSURANCE METHOD FUNCTIONS | 3-12 |
| 4.3-1 | QUANTITATIVE RESULTS | 3-38 |
| 7.2-1 | FCC CONTROL CARD FAILURE RATE | 3-59 |
| 7.2-2 | PREDICTED FCC CARD FAILURE RATES | 3-60 |
| 7.2-3 | FAILURE RATES FOR MAJOR RDFCS COMPONENTS | 3-61 |

# 1. INTRODUCTION TO INTEGRATED ASSURANCE ASSESSMENT

## 1.1. Justification for Integrated Assurance Assessment

The validation of flight-essential and flight-critical digital systems is of crucial importance in the determination of each system's airworthiness. This tutorial is a reference for certification engineers involved in the functional assessment of aircraft which use digital systems for avionics and flight control functions. Part 25 of the Federal Acquisition Regulation (FAR) (specifically FAR 25.1309) contains the requirements for flight-essential and flight-critical avionic systems. Federal Aviation Administration (FAA) Advisory Circular (AC) 25.1309-1 provides guidance material for demonstrating compliance with FAR 25.1309. This AC outlines the use of quantitative safety analysis techniques that may include: (1) Probability Analysis, (2) Fault Tree Analysis, (3) Failure Modes and Effects Analysis (FMEA), and (4) other comparable techniques for determining compliance with the requirements of FAR 25.1309 (b).

The use of any of these analysis techniques in isolation for an entire avionic system introduces some difficulties. To eliminate or minimize these difficulties, an Integrated Assurance Assessment (IAA) is desirable. A model for an integrated assurance approach may include a number of techniques, each contributing to the value of the analysis. For instance, fault tree analysis provides assurances that most system components have no single failure mode that results in system failure. When components have several failure modes that may have different effects on system readiness, fault tree analysis may prove cumbersome and inefficient. In this latter case, FMEA may be used to extend the examination below the component level to the failure modes of the individual piece-parts that comprise the component. Integrating fault tree analysis and FMEA permits the diagnosis of system failure from system-level functions through Integrated Circuit (IC) pins.

The above simple integration may be used to develop a probability for system failure. If the analysis contains no errors, this probability is true. If an error exists, the probability may be erroneously low and go undetected. Computation of the system failure probability by an independent analytical technique would either confirm the results of the above analysis or detect that an error was made. A computer-aided reliability analysis would provide that independent confirmation. Extending the integration to incorporate this technique means that the scope of the analysis begins at the system level and extends through the piece-part level, a system probability error may be computed, and a confirmation of that system probability error obtained.

In performing the fault tree analysis or FMEA, the effects of certain faults may not be known or may be uncertain. If this is the case, a technique for determining or verifying the effects of certain faults would improve the IAA. Fault Insertion is a technique for determining or confirming the fault effect.

Incorporating the fault insertion technique in an integrated assurance model improves the analysis that may be performed with that model.

The IAA model defined above utilizes four analysis techniques. Each of the four was incorporated in the model for specific reasons. Table 1.1-1 shows the techniques and the simplified capabilities each adds to the integrated model. Section 2 will discuss this model in greater detail.

TABLE 1.1-1. INTEGRATED ASSURANCE ASSESSMENT MODEL

| Technique | System Level Capability |
|---|---|
| Fault Tree | Determine System Failure Effect of Component and Piece-Parts; Compute System Probability of Failure |
| Failure Mode and Effect | Determine System Failure Effect of Piece-Parts |
| Reliability Analysis | Confirms System Probability of Failure |
| Fault Insertion | Confirms System Failure Effect of Component and Piece-Parts |

The evolution of an integrated assurance model is dependent on the system to be diagnosed. Simply, the capabilities of analysis, detection, annunciation, and confirmation must be provided. The specific analytic techniques that provide these capabilities may vary from model to model.

## 1.2. Primary Source for this Tutorial

The primary source for this tutorial is report Number DOT/FAA/CT-82/154, developed by the Lockheed-Georgia Company, entitled "IAA of a Reconfigurable Digital Flight Control System." The report documents an assurance assessment of a representative contemporary Digital Flight Control System (DFCS). The complementary use of various analytical techniques for safety analysis was stressed. Their primary objective was to explore and demonstrate the integrated application of reliability, failure effects, and system simulator methods in establishing the airworthiness of a flight-critical DFCS. The emphasis was on the mutual reinforcement of the methods.

## 1.3. How to Use this Document

This tutorial is divided into nine major sections:

- Section 1, "Introduction to IAA," provides an overview of IAA. It covers the Regulations and ACs to which IAA is a response. It shows four typical safety analysis techniques meshed to form an IAA model. The justification for incorporating each of the techniques is given.

- Section 2, "IAA of a Reconfigurable Digital Flight Control System," provides a discussion of the research effort introduced in Section 1. The digital system is defined and the dual-dual attribute of this system is discussed. Both the production configuration and the simulator for the subject digital system are discussed.

- Section 3, "Considerations in the Development of an IAA Model," discusses the benefits that accrue to the IAA effort for each of the safety analysis techniques incorporated. The IAA model demonstrated here was the model actually used in the subject study.

- Section 4, "Fault Tree Analysis," demonstrates the use of fault tree analysis as a top-down, deductive analysis tool to be used in the identification of conditions and functional failures that contribute to higher level component or system failure. This section contains a detailed application of fault tree analysis in the subject study.

- Section 5, "Failure Mode and Effect Analysis," demonstrates the use of FMEA as a bottom-up, inductive analysis tool to be used to determine what happens to the system or higher level components when individual parts fail. This section contains a detailed analysis of three modules of a microprocessor.

- Section 6, "Fault Insertion," demonstrates the use of fault insertion in determining and confirming fault effects by fault simulation. A detailed example is given.

- Section 7, "Failure Rate Development," demonstrates how failure rates required by the fault tree analysis and the reliability prediction tool were obtained. It does not illustrate a safety analysis technique applied in the IAA.

- Section 8, "Reliability Prediction using (Computer-Aided Redundant System Reliability Analysis) CARSRA," demonstrates the use of an analytical reliability prediction program in obtaining the probability of system failure. The details of the application of this tool to the subject study are given.

- Section 9, "Summary Statement," summarizes the tutorial. Summary commentary is given on the experience gained from performing the Redundant Digital Flight Control Analysis - relating benefits, limitations, and suggested improvements.

If the reader is interested in general knowledge, sections 1, 2, and 9 will suffice. This will provide an overview of what IAA is, when its use should be considered, and what advantages or disadvantages are likely to result from its use. Section 3 contains a detailed discussion on why the selected techniques were incorporated in the IAA model. Sections 4 through 8 contain a detailed discussion on each of the selected safety analysis techniques and their use in the subject study.

# 2. IAA OF A RECONFIGURABLE DIGITAL FLIGHT CONTROL SYSTEM

## 2.1. The Redundant Digital Flight Control System

Under report number DOT/FAA/CT-82/154, entitled "IAA of a Reconfigurable Digital Flight Control System," referenced in section 1.2, an IAA was performed on the Redundant Digital Flight Control System (RDFCS). This digital system was procured jointly by the FAA and the NASA-Ames Research Center and was a central part of the DFCS Verification Laboratory at NASA-Ames. Appendix A contains a summary description of the system architecture. While the sensors and servos are described in the production configuration, they were not actually present in the RDFCS configuration but were simulated.

In most operational modes, the system is fail-passive with a dual channel configuration. For automatic landings under Category IIIa conditions (see AC 120-28C), the system can be brought into a dual-dual fail-operational, fail-passive configuration. The classification dual-dual relates primarily to the four computer channels in the system. Each of the two Flight Control Computers (FCCs) has two channels which run frame-synchronously; each channel drives one coil of a dual-coil servo in each axis. Any indication of disagreement between the two channels in an FCC causes the servo connected to that FCC to disengage. Figure 2.1-1 summarizes the dual-dual configuration.

The RDFCS simulator is comprised primarily of a PDP 11/60 computer and the RDFCS pallet. The pallet is shown in figure 2.1-2. The PDP 11/60 computer hosts a discrete-state model of the airplane in which the RDFCS is installed. The RDFCS pallet contains the FCCs, the core memory for the FCCs, and the following devices:

- Modular Digital Interface Control Unit (MDICU) controls interface to the PDP 11/60.
- Computer Breakout Panel permits optional interruption of signals passing from MDICU to the FCC. The lower portion of figure 2.1-3 shows part of the Computer Breakout Panel.
- CAPS (Computer Aided Production Simulator) Test Adapters (CTA) permits address monitoring on the FCC processor bus. The upper portion of figure 2.1-3 shows the CTA.
- Servo Simulation Panel (SSP) simulates the dynamics of servos. The SSP is shown in figure 2.1-4.
- Discrete Switch Panel (DSP) contains switches affecting the Instrument Landing Systems (ILS). The DSP is shown in figure 2.1-5.

Appendix B contains a detailed description of the RDFCS simulator.

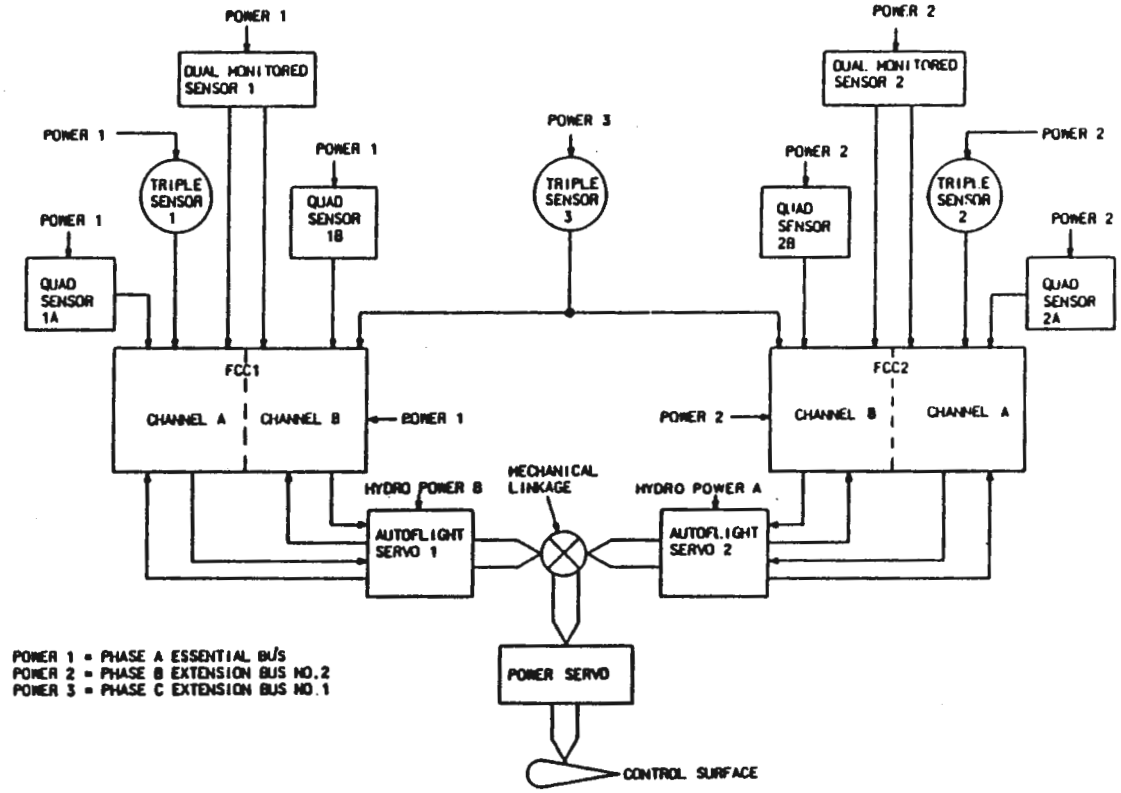



FIGURE 2.1-1. RDFCS DUAL-DUAL CONFIGURATION

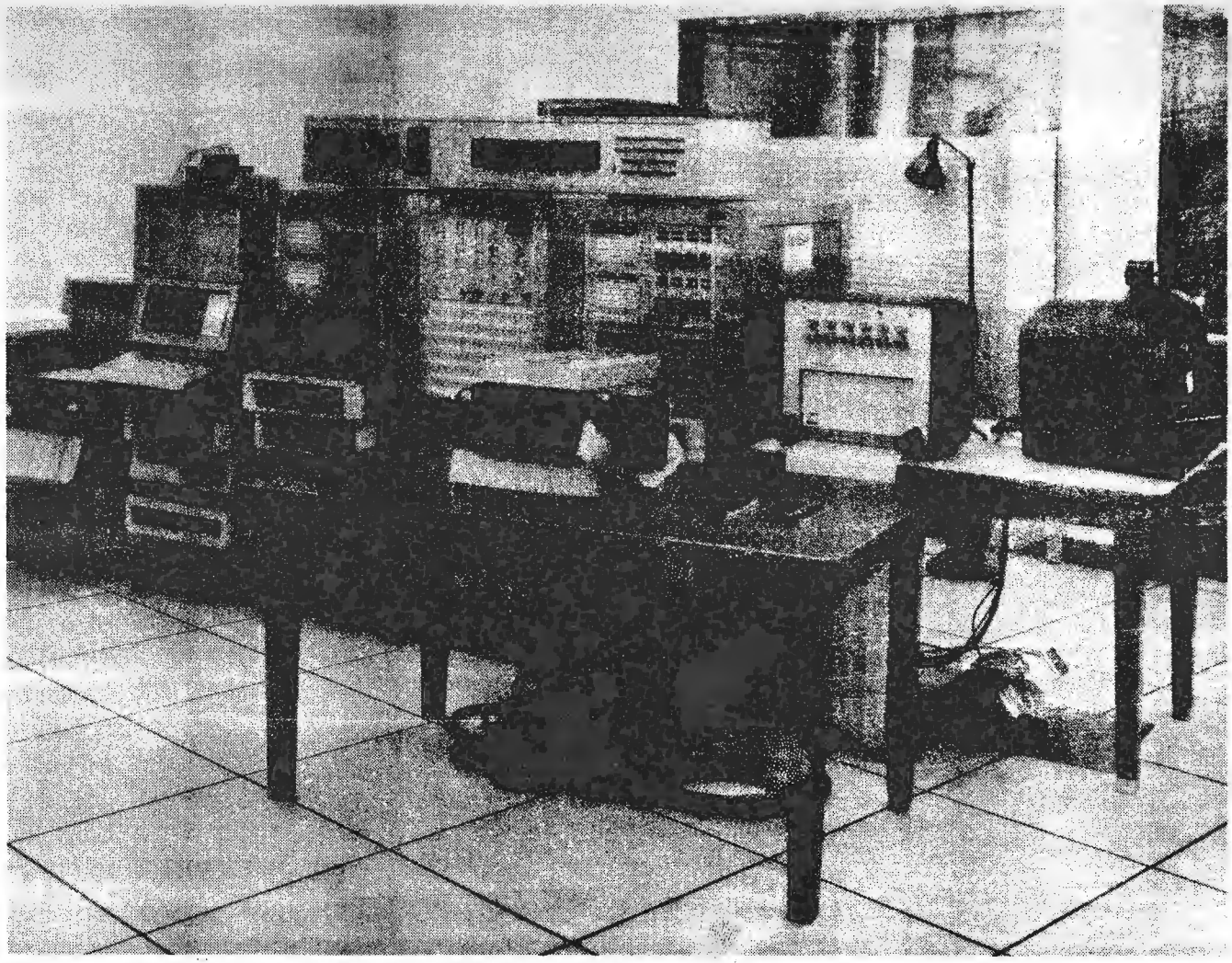


FIGURE 2.1-2. RDFCS SIMULATOR

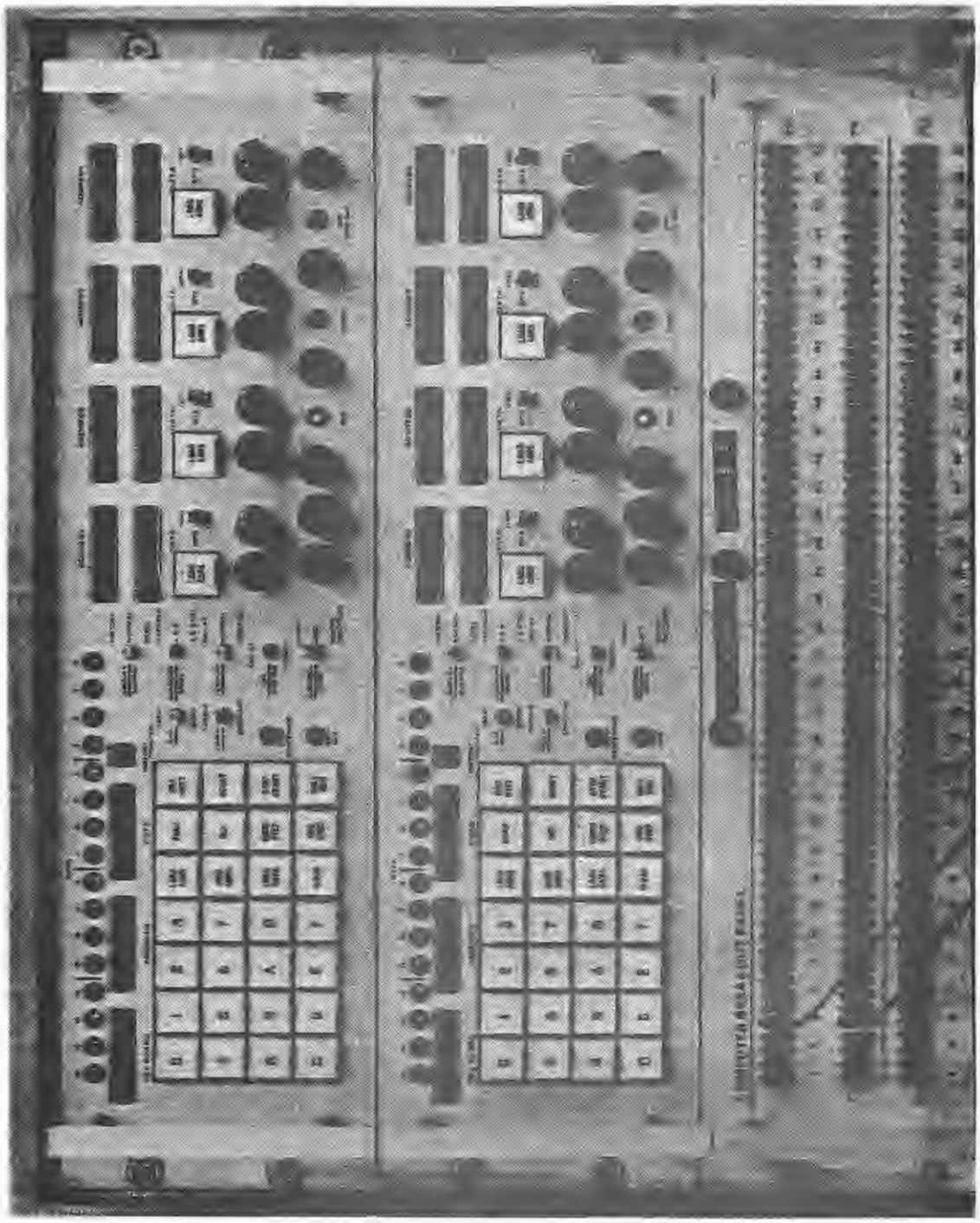


FIGURE 2.1-3. CTA AND COMPUTER BREAKOUT PANEL

FIGURE 2.1-4. SERVO SIMULATOR PANEL

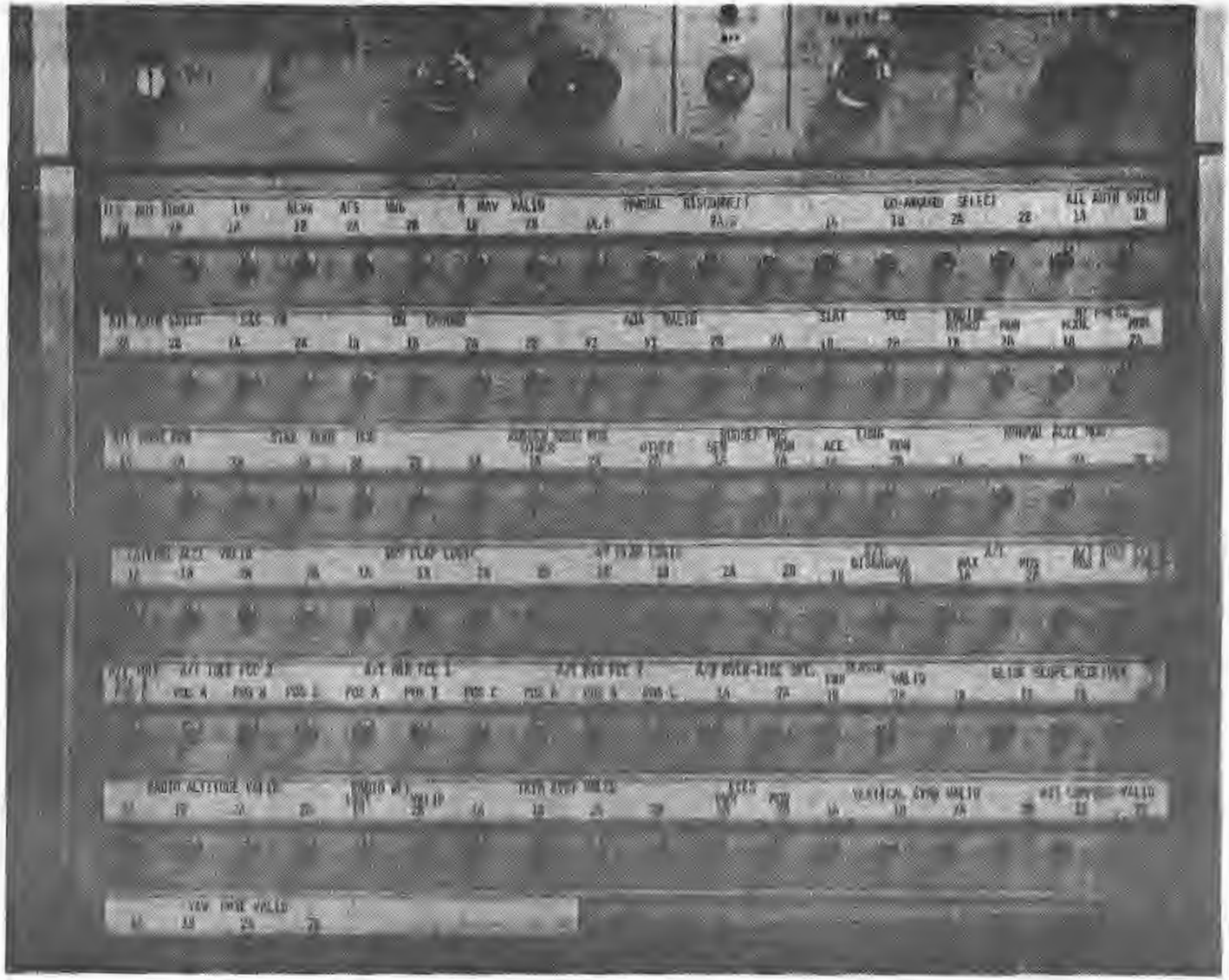


FIGURE 2.1-5. DISCRETE SWITCH PANEL

## 2.2. The IAA Study

During this study, the assurance methods of fault tree analysis, automated reliability prediction, FMEA, and fault insertion were applied to create a workable approach to confirming the airworthiness of the RDFCS. The scope of the effort was primarily limited to assessment of the automatic landing maneuver under Category IIIa conditions as defined in AC 120-28C. The analysis is based on the event that the airplane has an unacceptable deviation from the desired flight profile during the last 150 feet of descent while executing an automatic landing. This portion of the flight is the only flight phase during which the RDFCS performs a critical function. Category IIIa conditions imply that the human pilot cannot complete the landing using visual cues should the RDFCS fail. The application of safety analysis techniques was on a selective basis and addressed those system aspects identified in table 2.2-1.

The assessment consisted of the following major tasks:

- Application of fault tree analysis starting at the highest system functional level and proceeding to the hardware circuit card level and the module level for the processors.

- Development of a representative set of failure rates for the relevant hardware items.

- Application of an automated program, CARSRA, to the system failure modes affecting airworthiness.

- Application of FMEA to IC pin faults of three processor modules.

- Definition of faults to be inserted in the RDFCS. Faults to be inserted include faults used to determine the effect of the fault when analysis was not feasible, and faults used to confirm the manual analysis. These faults were subsequently inserted and the effects recorded.

## 2.3. Conclusions and Observations

The conclusions and observations resulting from this study are as follows:

- The integrated approach used in the study is capable, with diligent application, of establishing the airworthiness of a DFCS within the context of AC 25.1309-1. Specifically, this approach addresses those system aspects shown in table 2.2-1, including freedom from single-point failure modes and system failure probability.

- The IAA used in this study should be considered for use in validating other digital systems, including DFCS, in compliance with AC 25.1309-1.

TABLE 2.2-1. ASSURANCE METHOD FUNCTIONS

| SYSTEM ASPECT | ASSURANCE METHOD | |
|---|---|---|
| | PRIMARY | CONFIRMATION |
| FAILURE EFFECTS | | |
| - COMPONENT | FAULT TREE ANALYSIS | FAULT INSERTION |
| - DIGITAL MODULE | FAULT TREE ANALYSIS, FAILURE MODE AND EFFECT ANALYSIS | FAULT INSERTION |
| - DIGITAL INTEGRATED CIRCUIT | FAILURE MODE AND EFFECT ANALYSIS | FAULT INSERTION |
| - UNTRACTABLE CASES | FAULT INSERTION | |
| FAULT DETECTION/ ANNUNCIATION | FAULT TREE ANALYSIS | FAULT INSERTION |
| SOFTWARE FUNCTION IMPLEMENTATION | SOFTWARE TEST PROGRAM | FAULT TREE ANALYSIS |
| NO SINGLE-POINT FAILURE MODES | ABOVE, AS RELEVANT | ABOVE, AS RELEVANT |
| SYSTEM FAILURE PROBABILITY | RELIABILITY PREDICTION PROGRAM | FAULT TREE ANALYSIS QUANTITATIVE EVALUATION |

- The quantitative assessment of system failure probability by two methods (fault tree analysis and analytical reliability prediction) offers increased assurance that the system meets the quantitative requirements of AC 25.1309-1. For a flight-critical system, this requirement is that the system failure probability not exceed $10^{-9}$ per hour of flight for each critical function the system performs.

- Fault insertion confirms that the fault detection capability and the fault tolerance capability described in the system documentation are actually implemented in the system. Since the fault tree analysis is based largely on the system response to faults as described in the system documentation, the fault insertion confirms that the fault tree analysis correctly reflects the behavior of the actual system in the presence of faults.

- The fault tree analysis generates software test requirements in terms of functions which the software must perform. These, in turn, provide a check of function criticality and of test requirements generated in accordance with Radio Technical Commission for Aeronautics (RTCA) Document DO-178.

- Fault tree analysis proved unwieldy below the circuit card level, because at lower levels many more functions are being performed than there are hardware failure modes. FMEA was accomplished successfully at the IC pin level.

- As a training facility and a Reconfigurable Test Bed, the RDFCS facility has significant and valuable capabilities for investigating assurance issues of currently definable DFCS architectures. It also has potential enhanced capability in certain areas, such as automated insertion of pin-level faults, for confirmation of analytically determined failure effects.

- The comparison of the time or cost required for the integrated approach reported here with that required for other possible assurance approaches was not specifically addressed in this study. However, the time required for the integrated approach is expected to compare favorably with that for other approaches, assuming the same depth of analysis. The cost should also compare favorably, provided a facility suitable for fault insertion is available.

# 3. CONSIDERATIONS IN THE DEVELOPMENT OF AN IAA APPROACH

## 3.1. General Comments

As a first step, a preliminary hazard analysis should be performed on the aircraft model. This analysis should identify those system functions that are critical and the aircraft systems that must operate correctly in order for the function to be performed correctly. According to AC 25.1309-1, a critical function is one "whose failure would contribute to or cause a failure condition which would prevent the continued safe flight and landing" of the aircraft. Essential system functions should also be identified. Essential functions are those "whose failure would contribute to or cause a failure condition which would significantly impact the safety of the airplane or the ability of the flight crew to cope with adverse operating conditions." A complete quantitative system analysis is normally required for critical systems but is more discretionary for essential systems. The following list illustrates the types of functions that may be critical:

- The primary flight control system.
- Hydraulic power for airplanes with powered flight control systems and no manual reversion.
- Secondary flight control systems, if failure of these systems can result in uncontrolled flight.
- Engine control system elements that affect all engines simultaneously.

Since the RDFCS performs critical system functions, a rigorous assurance assessment is justified. AC 25.1309-1 suggests that failure conditions for critical system functions be improbable. This requires a failure probability of less than $1.0 \times 10^{-9}$. An IAA incorporating both quantitative and qualitative analyses offers advantages over the application of a single methodology. The analytical techniques used here to perform the integrated assessment are in harmony with AC 25.1309-1.

## 3.2. An IAA Approach

The steps required in performing a typical integrated assessment of the RDFCS are as follows:

- System Description: Define a baseline configuration of the RDFCS and prepare a corresponding analytical description. This description may include those additional components, beyond the RDFCS, whose exclusion would make the analysis unrealistic.

- Fault Tree Analysis: Perform a fault tree analysis beginning at the system level. To be meaningful, this analysis should extend beyond the component level to the piece-part level. This will be the case when digital modules are present with a large number of different failure modes. To perform this total analysis, both the fault tree and the FMEA tools should be used.

- Failure Rate Generation: Develop a set of representative failure rates for the components and parts of the RDFCS that will permit the evaluation of the fault tree for failure probability.

- Fault Simulation Review: Define the simulated fault conditions that are to be inserted into the RDFCS simulator. These faults shall be inserted for two purposes: to confirm the assumptions underlying the fault tree analysis, and to resolve uncertainty of the effect of the fault when analysis is not easily controlled.

- Flight Case Review: Review those flight cases available on the RDFCS simulator. If they are inadequate, augment them. Assure that all needed transitions between cases are possible.

- Reliability Analysis: Determine the tool to be used for analytic computation of system reliability. The CARSRA reliability program was used in the subject study, and is discussed in this tutorial.

## 3.3. Fault Tree Role in Integrated Assurance

The IAA of the RDFCS begins with a fault tree analysis of the system function. As shown in table 2.2-1, the fault tree analysis has several functions. The first function is to assure that no system component has any failure mode which can result in system failure. Most of the components, such as the sensors and servos, have only a few failure modes which can be observed at the interfaces with the rest of the system. For these components, the fault tree analysis provides assurance that no failure modes can cause system failure. The assurance is obtained by reviewing the completed tree and determining that system failure can only occur as a result of multiple failures.

In general, digital modules (and therefore digital components) can have a substantial number of different failure modes. In such cases, it becomes quite laborious to continue the fault tree development to a level of detail sufficient to confirm that none of those failure modes can cause system failure. The second function of fault tree analysis is to identify which digital modules are involved in performing critical functions. The task of assuring that no single module level failure can cause system failure is performed with FMEA.

A major benefit of fault tree analysis is that it focuses on the functions performed by the system elements, including those system elements involved in detecting faults and providing appropriate annunciation to the flight crew. Consequently, the third function of fault tree analysis is to confirm the adequacy of monitoring (i.e., fault detection and annunciation) in the system.

Fault tree analysis is also used to identify specific software functions required for system operation, including fault monitoring implemented in

software. The software test requirements for these functions are then specifically reviewed to confirm that these requirements are adequate. This fourth function of fault trees is illustrated and discussed in greater detail as the tree for the RDFCS is developed.

The fifth function of fault tree analysis is to provide an alternate means of computing the probability of system failure. This provides a check of the probability obtained from the reliability analysis. In the subject study, the fault tree analysis verified the probability obtained from the CARSRA program to ensure that the CARSRA input does not have errors which would produce a false low probability of system failure.

### 3.4. Failure Mode and Effect Analysis Role in Integrated Assurance

Fault tree analysis provides assurance that most system components, such as analog sensors and servos, have no single failure mode which produces system failure. This is because such components have only a few possible failure modes, and it frequently is not necessary to distinguish in the fault tree among these modes. When it is necessary to distinguish among modes, it is usually fairly simple to identify the modes which are relevant in the branch of the tree being developed. The analysis can often be extended below the component level to the failure modes of the individual piece-parts which comprise the component. Analysis to this very detailed level is sometimes necessary to ascertain that a component has no failure modes which could remain undetected until a second failure occurs elsewhere in the system.

However, fault tree analysis is cumbersome and inefficient if extended from system level to the IC pin level in the processor of a digital system. Essentially, this is a result of two basic characteristics of digital systems:

- Functions which are described very simply at a higher level (e.g., sensor monitoring) require a myriad of sequential operations at the IC level. These operations are required to obtain the proper data, route it to the proper registers within the Arithmetic Logic Unit (ALU) where arithmetic and logic operations are actually performed, and route the results to the proper storage register or output port. Many different ICs are involved in each of these operations.

- Many interfaces between ICs involve several pins, and it is the combination of pin states (electrically high or low) which is significant. That is, each combination of pin states represents a different data value or instruction, and the effect of a single pin being in the wrong (faulted) state depends on the state of the other (non-faulted) pins.

The net result of these characteristics of digital hardware is that there are many more integrated-circuit-level operations performed in executing the flight software than there are pin-level failure modes. In extending a fault tree analysis from failure of system-level functions to failure of IC pins, all of these detailed operations must be included and accounted for-an extremely inefficient process. Once the fault tree had been fully developed, another extremely laborious task would remain: reviewing the tree to make certain (1) that all of the failure modes of the ICs had been accounted for, and that no

failure mode could remain undetected until a second failure occurred, with the combined effect of both faults producing a hazardous condition; and (2) that no failure mode could by itself produce a hazardous condition.

FMEA provides a means of systematically examining all of the potential failure modes of the ICs to confirm that none of them could cause a hazard directly or remain latent and subsequently cause a hazard in conjunction with a second failure.

### 3.5. Reliability Analysis Role in Integrated Assurance

CARSRA is an analytical reliability prediction program used in this IAA to obtain the probability of system failure (Bjurman, et al., 1976). The use of CARSRA, along with the quantitative assessment produced by evaluating the fault tree analysis, provides two independent computations of system failure probability. This reduces the risk of a false, low probability of failure being produced by a single method and the error remaining undetected.

Although CARSRA is identified specifically in the IAA used in this analysis, some other method (except fault tree analysis) could be used. If an alternate method is used, it should have sufficient configuration adaptability to produce the predicted probability of system failure without requiring simplifying assumptions which would produce a false, low prediction. According to Hitt and Eldredge, the particular fault-tolerant system attributes that reliability models should be capable of handling are the following:

- Computer design (e.g., failure modes, loss of subsets of functions)
- Dependencies between subsystems
- Number of redundant components
- Redundancy management (e.g., active or standby spares, voting algorithms)
- Permanent, latent, and transient faults
- Fault recovery (i.e., coverage)
- Data transmission schemes and bus architectures
- Partitioning of processing functions

Some of the earlier reliability analysis models were adequate for analysis of certain types of systems. Examples of these are Tree Aided System Reliability Analysis (TASRA), Reliability Block Diagram Computer Program (RBDCP), Reliability (REL 70), and Reliability Computers (REL COMP) models that were primarily developed in the early 1970s. Later models such as, Automated Reliability Interactive Estimation System (ARIES) (Ng and Avizienis, 1978), Computer Aided Reliability Evaluator (CARE II) (NASA, NASI-12668, March 1974), CARSRA (Bjurman, et al., 1976), and Complimentary Analytic Simulative Technique (CAST) (Conn et al., 1977) offered expanded capabilities that were more adaptable to systems

having more components and greater complexity. CARE III and an improved ARIES are examples of the continuing evolution.

The certification engineer must choose the appropriate reliability analysis model based on the ability of that model to represent the system to be analyzed. In the event that a suitable match cannot be made, manual analysis may be the only alternative.

## 3.6. Fault Insertion Role in Integrated Approach

Fault insertion is used in this IAA for three purposes as shown in table 2.2-1.

- Faults are inserted, on a sampling basis, to confirm the fault effects reflected in the fault tree analysis and fault effects determined during FMEA. This includes faults of components (sensors and servos in this study) and faults of ICs (pin-level faults in the digital processor).

- Faults are inserted, also on a sampling basis, to confirm fault detection and annunciation functions implemented in the system. Many of these are also inserted to confirm fault effects.

- Faults are inserted to determine the effect when the analysis is intractable or when there is some uncertainty in the analysis result.

# 4. FAULT TREE ANALYSIS

## 4.1. General Comments

The role of fault tree analysis in the IAA was specified in section 3.3. Appendix 1 of AC 25.1309-1 stipulates that for the purpose of conducting failure analyses, "the format of quantitative analyses, which use NUREG-0492 as a guide, will be acceptable to the FAA." NUREG-0492 is a textbook for fault tree analysis published for those unfamiliar with the concepts of systems analysis. It is titled "Fault Tree Handbook," and contains an excellent bibliography on the entire subject of reliability.

## 4.2. Fault Tree Development

AC 25.1309-1 defines fault tree analysis as "a top down deductive analysis identifying the conditions and functional failures necessary to cause a defined failure condition." The fault tree, when defined for the entire system failure state, may be used to mathematically determine a system failure probability by proper consideration of the failure probabilities of all contributing events.

Fault tree analysis is a structured method of "exploding" a top-level fault into all external events, conditions, or lower-level faults that contribute to it. The top-level fault may be a system failure or a component failure. In the former case, the lower-level faults are component faults; in the latter, piece-part faults. If they are not elementary, component faults may then be recursively exploded into their respective causative elements. The topology of the resulting graph (fault tree) permits the evaluation of the combinations of external events, conditions, or faults that cause the top-level fault.

In complex systems, the graphs become so cumbersome that user aids are necessary. One such aid is the "cut set." The "cut set" is the combination of lower-level events that result in a certain higher-level failure. The procedure for obtaining cut sets may either be manual or automated. The length of a cut set portrays the levels of redundancy in the system design. Minimal cut sets are important in that they show the minimum levels of redundancy. Minimal cut sets simplify the computation of the probability of failure of the respective top-level event.

The fault tree analysis contained in this section is based on the RDFCS being in the failure condition stipulated in section 2.2: simply, during a crucial flight landing phase, when the human pilot may not complete the landing, the RDFCS fails. In the following analysis this time period will be referred to as the "crucial phase."

The analysis begins with the RDFCS in the dual-dual configuration. The system is placed in this configuration by depressing the ILS push-button. The use of the ILS push-button results in the selection of the Approach/Land (A/L) mode.

After this switch has been momentarily depressed, the A/L mode is transmitted to the FCCs and latched in. The switch is no longer needed, and therefore does not enter into the analysis.

Figures 4.2-1 through 4.2-12 contain a partial fault tree analysis of the RDFCS during the crucial phase. The numbers in engineering notation appearing at nodal points are failure probabilities and will be used in sections 4.3 and 7.2.

The top event of figure 4.2-1, Unacceptable Deviation from Path/Altitude/ Speed Profile during Crucial Flight Phase, may be considered as equivalent to RDFCS failure. This event can be caused by any of three conditions or subevents. For convenience, these can be referred to as Level-2 events with the top event considered to be at Level-1. The Level-2 events are shown as the middle row in figure 4.2-1. The first of these is that the system design is in some manner deficient for the environmental conditions encountered. This includes the possibility that the conditions encountered are outside of the system design requirements. It also includes the possibility that the control laws are deficient for some conditions which may be expected. This possibility is outside the scope of this tutorial and is not pursued here. Mulcare et al., (1979) and DOT/FAA/CT-82/140 (1982) address this subject. In particular, section 3.3.1.3. of Mulcare et al., (1979), discusses establishing an upper bound on the probability of a deficient control law by statistical methods.

The second of the Level-2 events occurs if the airplane enters the crucial phase with the RDFCS not fail-operational, and then a component failure occurs which prevents the system from completing the landing.

The third of the Level-2 events is that the crucial phase is entered with a fail-operational RDFCS, but multiple component failures occur before the end of the phase, and these failures result in RDFCS system failure.

The second of the Level-2 events, that the crucial phase is initialized without fail-operational capability, is expanded into three relevant functional areas or Level-3 events: sensing aircraft altitude and position, computing required outputs, and servo response to computed commands. The first of these, the sensing function, is expanded in figure 4.2-2 into the various parameters needed by the FCCs in the automatic landing control laws. At this and higher levels, the fault tree is functionally oriented: failures are in terms of loss of function rather than loss of hardware.

The fault tree stub of figure 4.2-3 extends the sensing function for normal acceleration to the individual hardware elements used to measure the acceleration and transmit it to the computers. The failure of that normal acceleration signal No. 1 to be present in all computer channels can be caused by loss of the sensor itself, associated wiring, or one of the circuit cards involved in receiving the signal and transmitting it to all channels. The A24 Autoland Sensor Input and A27 Discrete Input Cards are both involved. The A24 card handles the analog acceleration signal and the A27 card handles the validity discrete signal. The processor itself is not involved in the data acquisition process and so is not shown. At this level, the transition has been made from required functions to the hardware which perform these functions.

Failure of the system to provide a NO DUAL annunciation is shown in figure 4.2-4. This figure is of particular interest because of the explicit software function identified. A failure rate of zero is assigned to failure of this function, because it can be explicitly and exhaustively tested. Once it has been so tested, the probability of both NO DUAL annunciations failing because of a generic software error is taken to be zero. A generic software error is a discrepancy in the software that will cause all computer channels which use that software to produce the same, but wrong, result. Multiple computer channels do not provide redundancy with respect to generic software errors as long as the same software is used in all channels (as it is in most contemporary systems) including the RDFCS. DOT/FAA/CT-82/140 may be consulted for a discussion of software errors, and RTCA Document DO-178 should be consulted for a discussion of software test requirements.

Fault tree stubs similar to that shown in figure 4.2-3 were developed for the other sensors of figure 4.2-2. These are very much like the stub shown in figure 4.2-3 and so are not included in this tutorial.

The second of the Level-3 events of figure 4.2-1 is initiation of the crucial flight phase without fail-operational computing capability, and that an additional component failure causes system failure before the phase is complete. This is shown in figure 4.2-5 as four Level-4 events. The first of these, that channel A of FCC No. 1 fails above alert height, can be caused by either channel of the FCC failing to produce a required output, as shown by the eight events at the lowest level (Level-5) in figure 4.2-5.

Figure 4.2-6 continues the development of the fault tree for one of the Level-5 events of figure 4.2-5. This event, failure of the A channel of FCC No. 1 to produce a rudder command, can be caused by failure of any one of several cards within the channel. In this study, the two cards which make up the processor were considered in more depth than the others. These two, the A13 Control Card and the A14 Data Path Card, are shown in figures 4.2-7 and 4.2-8, respectively. Also shown in each of figures 4.2-7 and 4.2-8 is a subevent for failure of a miscellaneous part, such as the circuit board, the edge connector, or other part which is not included in one of the modules named in the other blocks.

Theoretically, the fault tree analysis of the failure of the processor to compute the rudder command can be continued below the module level to the individual IC pins or discrete piece-parts. The desirability of doing this is questionable, however, because of the nature of the processor. The processor is not designed to perform a single specific function, such as computing rudder commands. It is designed to efficiently perform a number of simple functions, such as addition, multiplication, and logic operations. A suitable sequence of such operations (i.e., the flight software) is used to make the processor generate the rudder command, the aileron command, and so forth. It is much easier to relate the modules and ICs to the simple functions (addition, multiplication) than to relate them to the more complicated functions of computing the command for a particular servo.

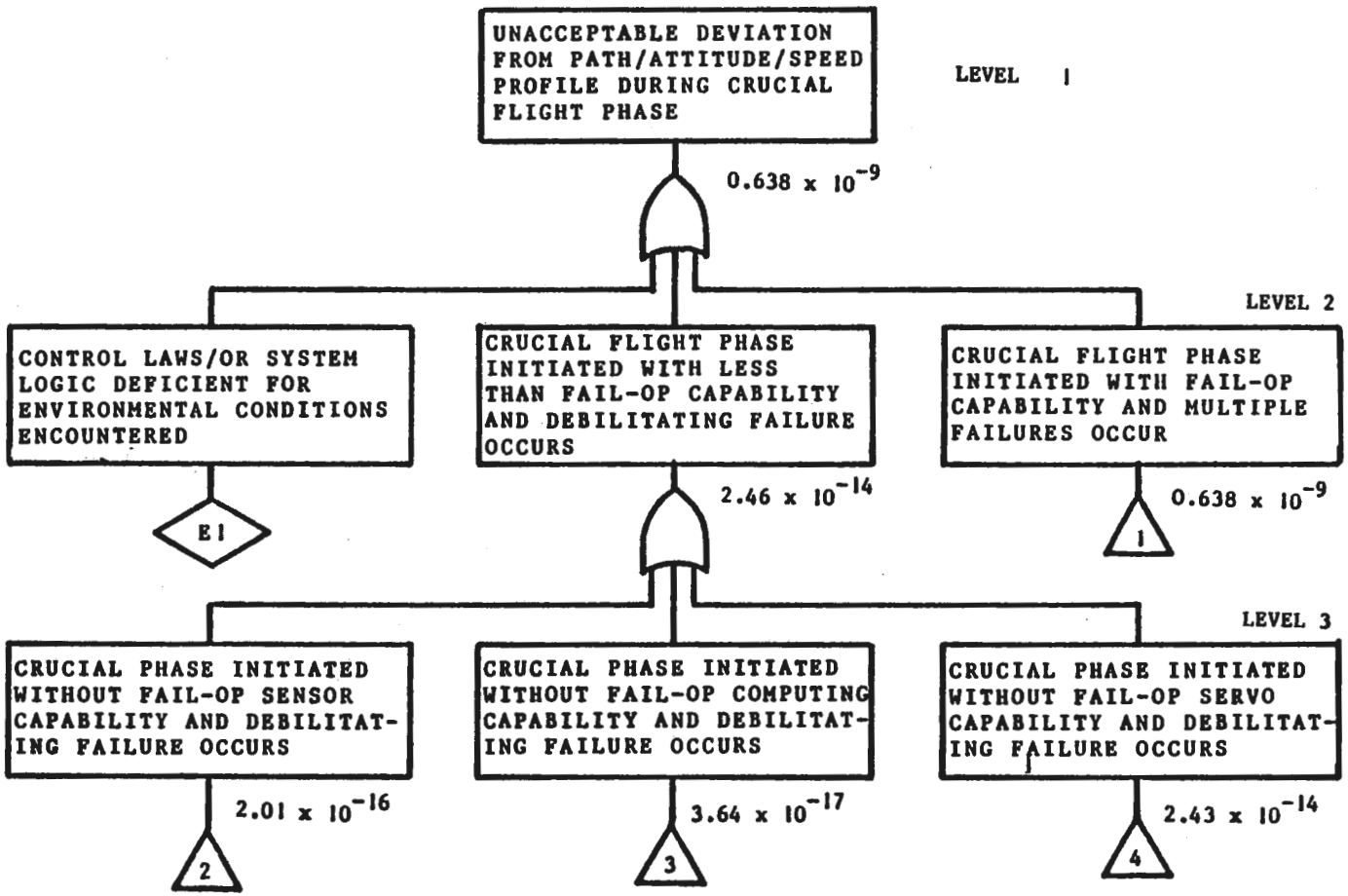



FIGURE 4.2-1. FAULT TREE TOP LEVEL

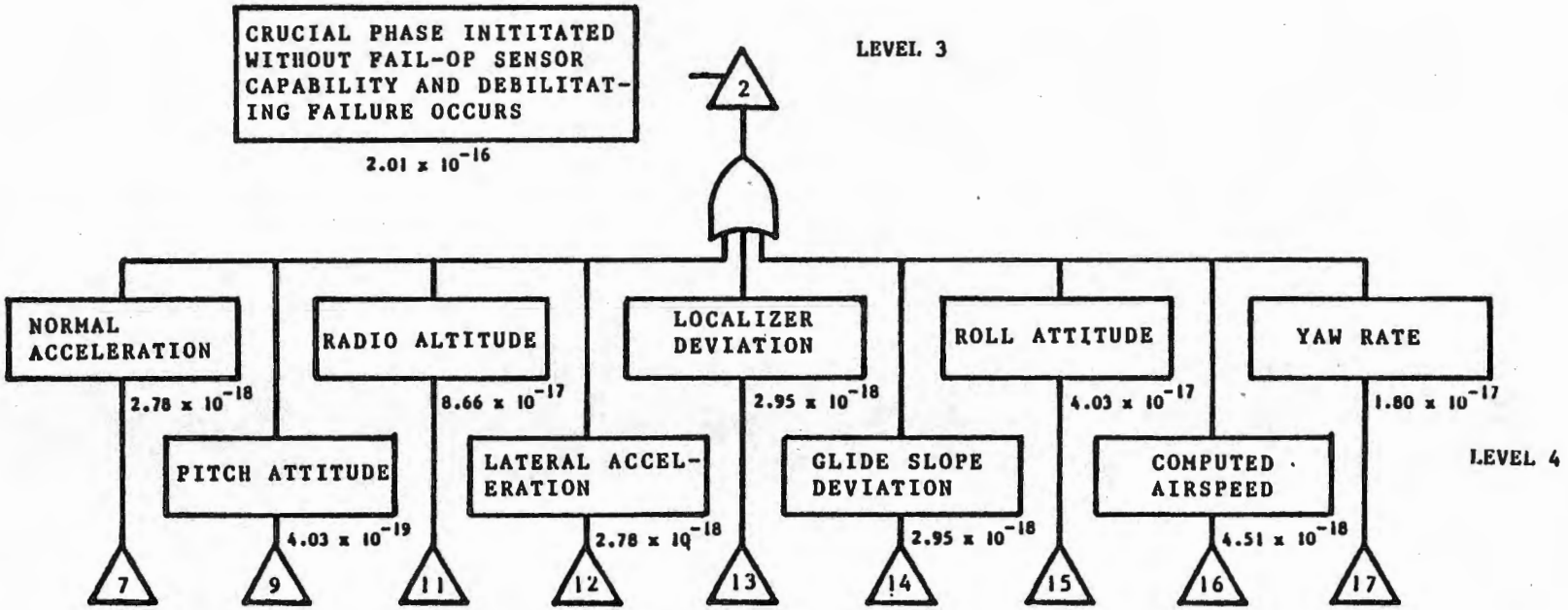



FIGURE 4.2-2. SENSING FUNCTION

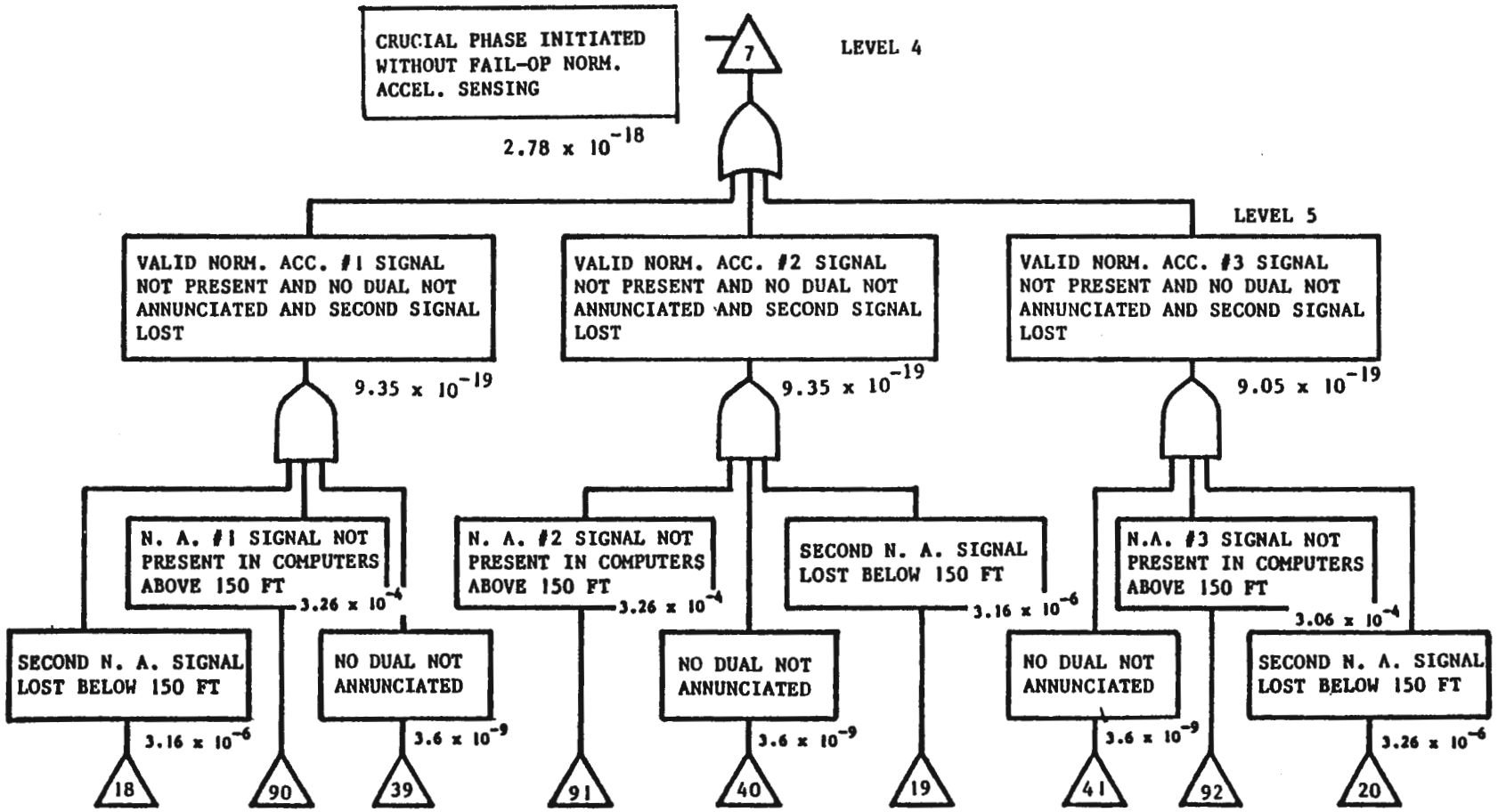



FIGURE 4.2-3. NORMAL ACCELERATION SENSING (1 of 3)

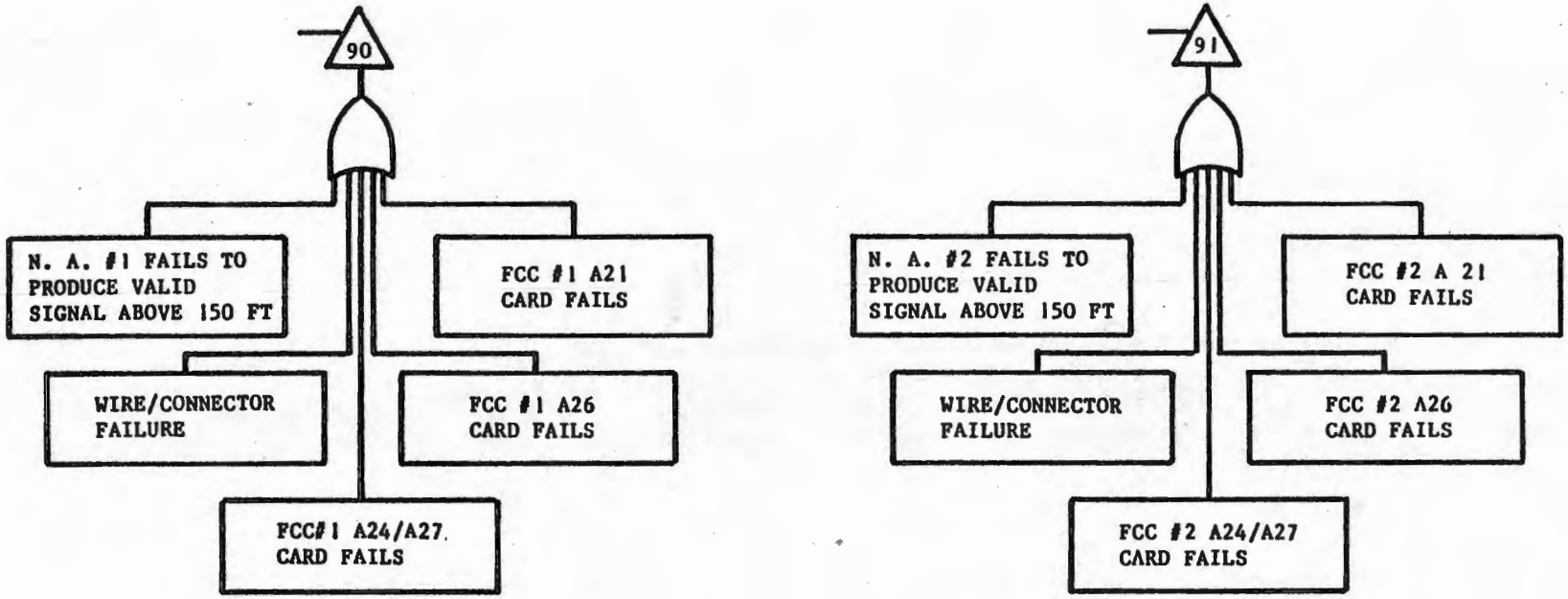



FIGURE 4.2-3. NORMAL ACCELERATION SENSING (2 of 3)

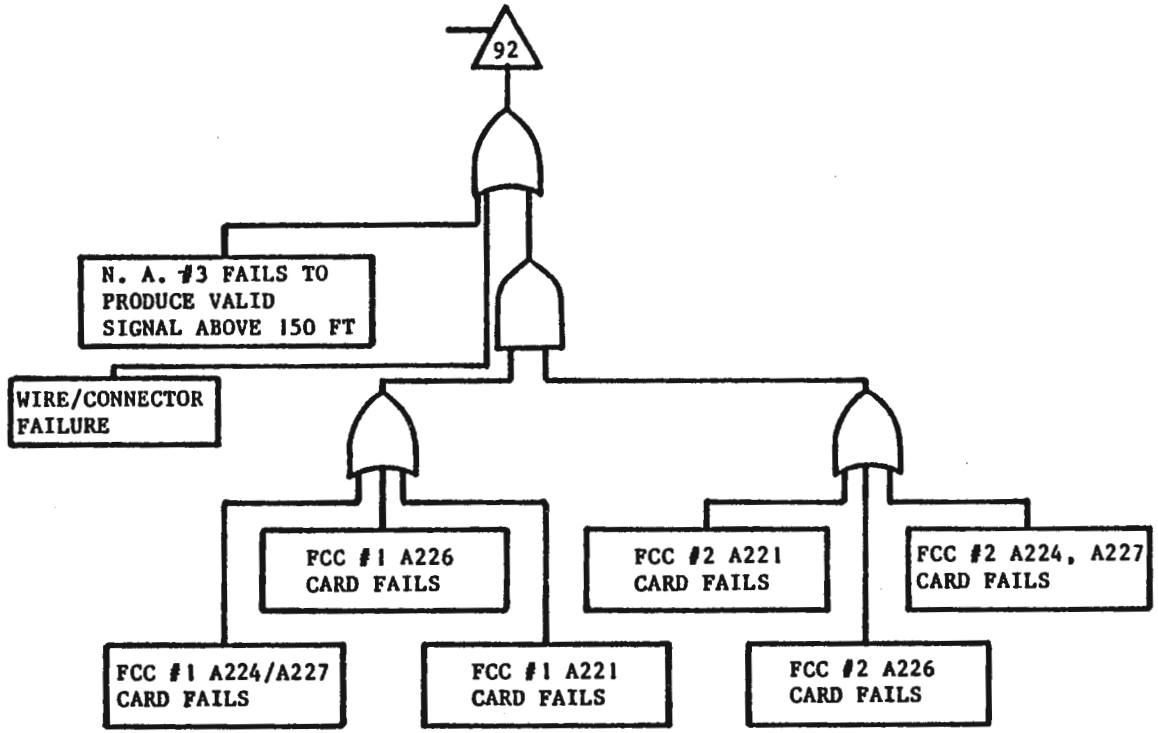



FIGURE 4.2-3. NORMAL ACCELERATION SENSING (3 of 3)

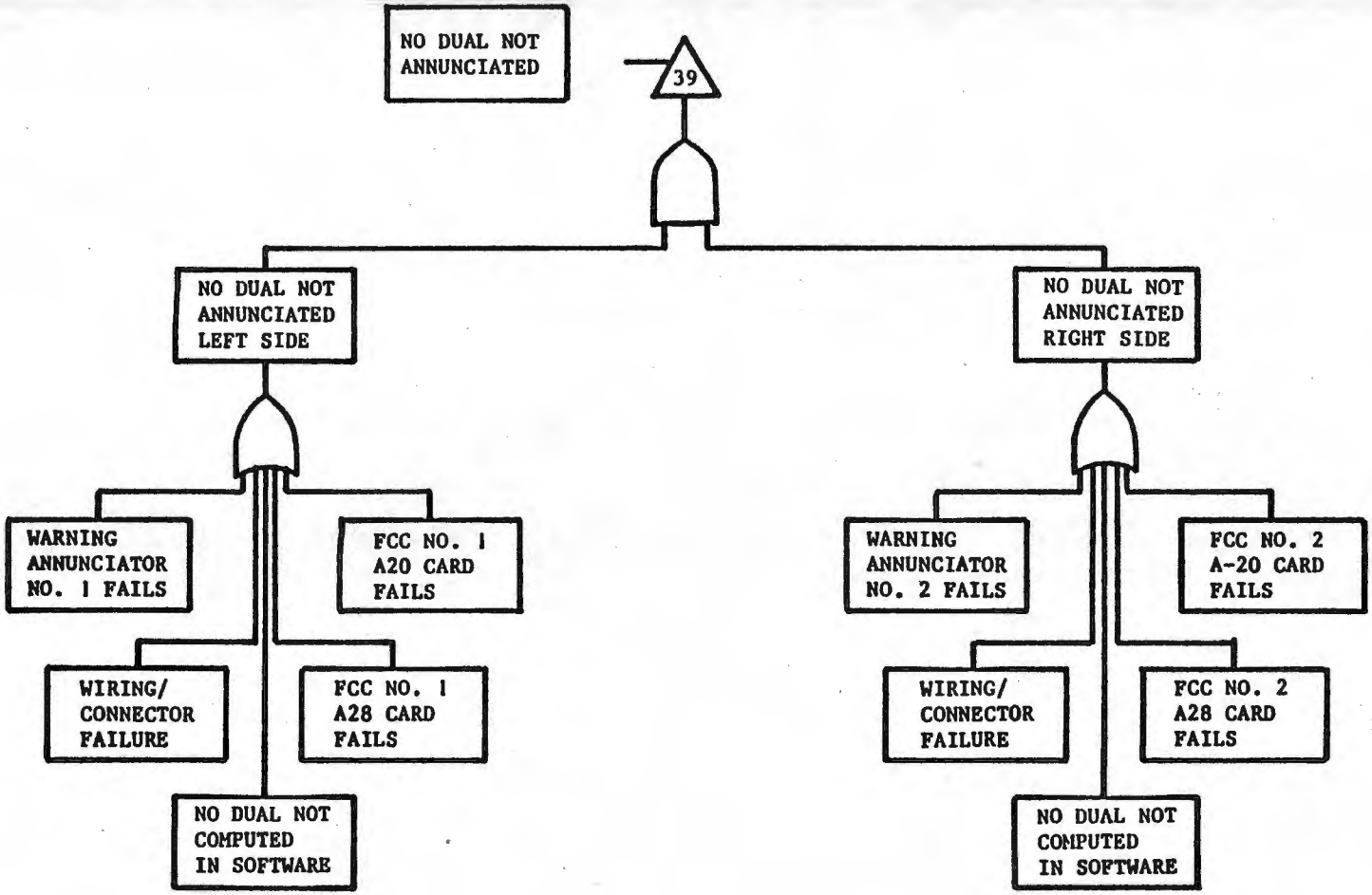



FIGURE 4.2-4. NO DUAL ANNUNCIATION

It is also easier, in general, to relate a specific failure mode of an IC within the processor to its effect on the processor operation than to start with the effect and then work in the other direction to the IC failure modes which would produce the effect. In other words, it is easier to do an FMEA than a fault tree analysis at this level. FMEA will be discussed in section 5.

Another reason for preferring FMEA to fault trees at this level is that in the course of performing the fault tree analysis, the analyst must account for all of the ways the processor can fail; that is, all of the ways in which the processor output can be wrong.

These ways are the failure modes of the processor. Each of these modes must then be traced to all possible combinations of IC pin failures which could produce the processor failure mode. Because processors have many different possible outputs, there are a high number of ways that the output could be wrong. There is no practical way of assuring that all of these possibilities have actually been covered in the fault tree. The FMEA requires that all pin-level IC failure modes be considered. These modes are much better understood, and there are fewer of them, so that it is much easier to be certain that they have all been covered. This is not meant to imply that a complete pin-level FMEA is easy or inexpensive; it is neither.

Due to the above considerations, the fault tree analysis of the processor was not continued below the level developed in figures 4.2-7 and 4.2-8. Instead, the FMEA approach was used as described in section 5.

To continue with the development of other branches of the fault tree, figure 4.2-9 develops the event of figure 4.2-6 in which the pilot is not warned that FCC No. 1 A channel is not generating a correct rudder command. This portion of the fault tree includes several software functions. In a production program, the test requirements of each of these functions should be reviewed to confirm that they satisfy the criteria of RTCA Document DO-178. In this tutorial, conducted for illustrative purposes, this review was not made.

Tree stubs similar to that developed in figures 4.2-6 through 4.2-9 were developed for the other required outputs from Channel A of FCC No. 1 and the other three channels (figure 4.2-5). They are not included here because they repeat the analysis shown.

The last of the Level-3 events of figure 4.2-1 is that the crucial phase is initiated without fail-operational servo capability, and a debilitating failure occurs. This is expanded in figure 4.2-10 into the three aircraft control axes: roll, pitch, and yaw. Figure 4.2-11 shows the fault tree for failure of the No. 1 yaw autopilot servo, with the servo failure not annunciated to the crew.

Fault tree stubs for the other five servos of figure 4.2-10 were developed to complete the analysis of the Level-3 events of figure 4.2-1. These are similar to the stub shown for the rudder servo and are not included in this discussion. This completes the discussion of the second of the Level-2 events of figure 4.2-1.

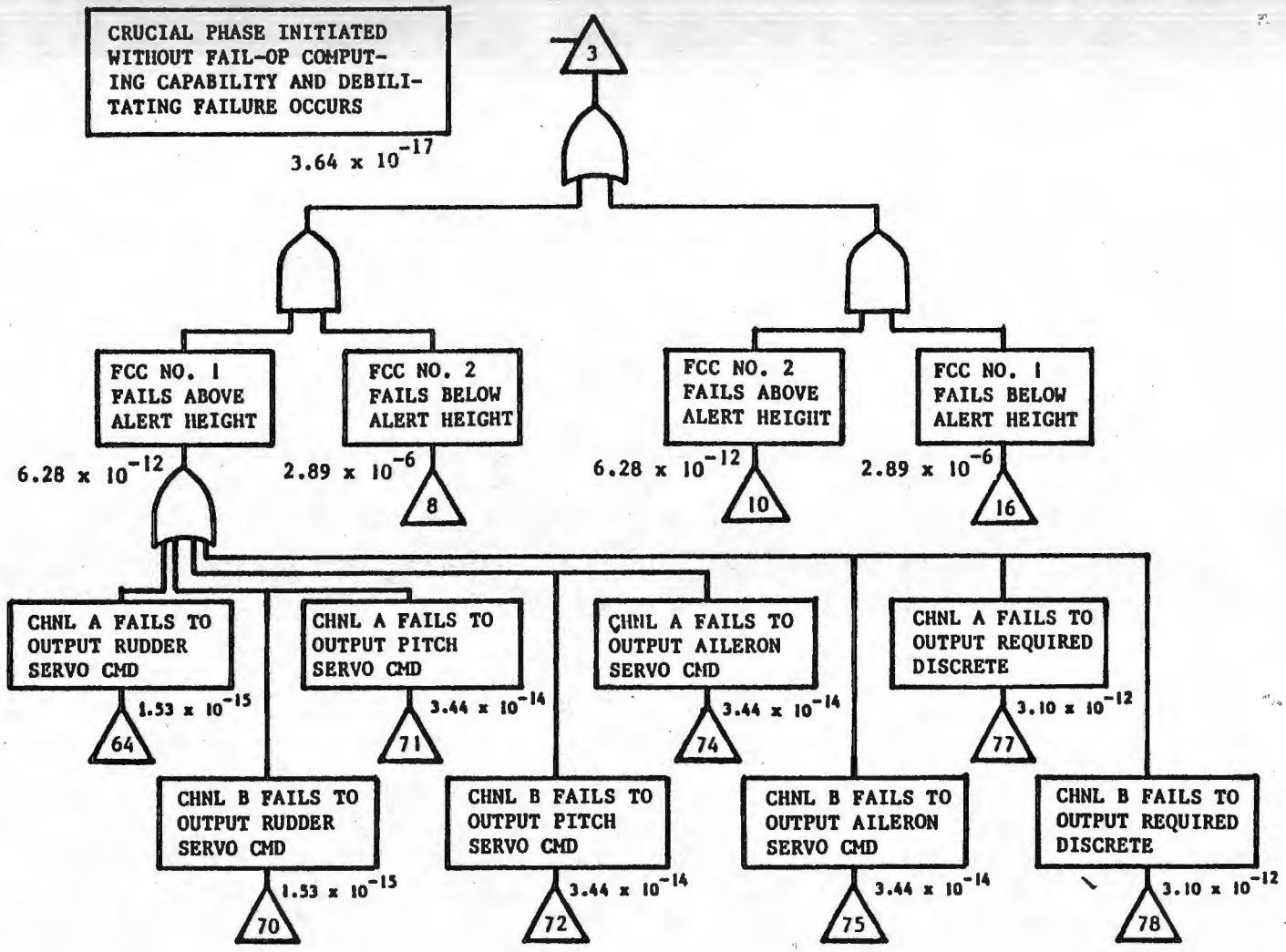



FIGURE 4.2-5. COMPUTING FUNCTION

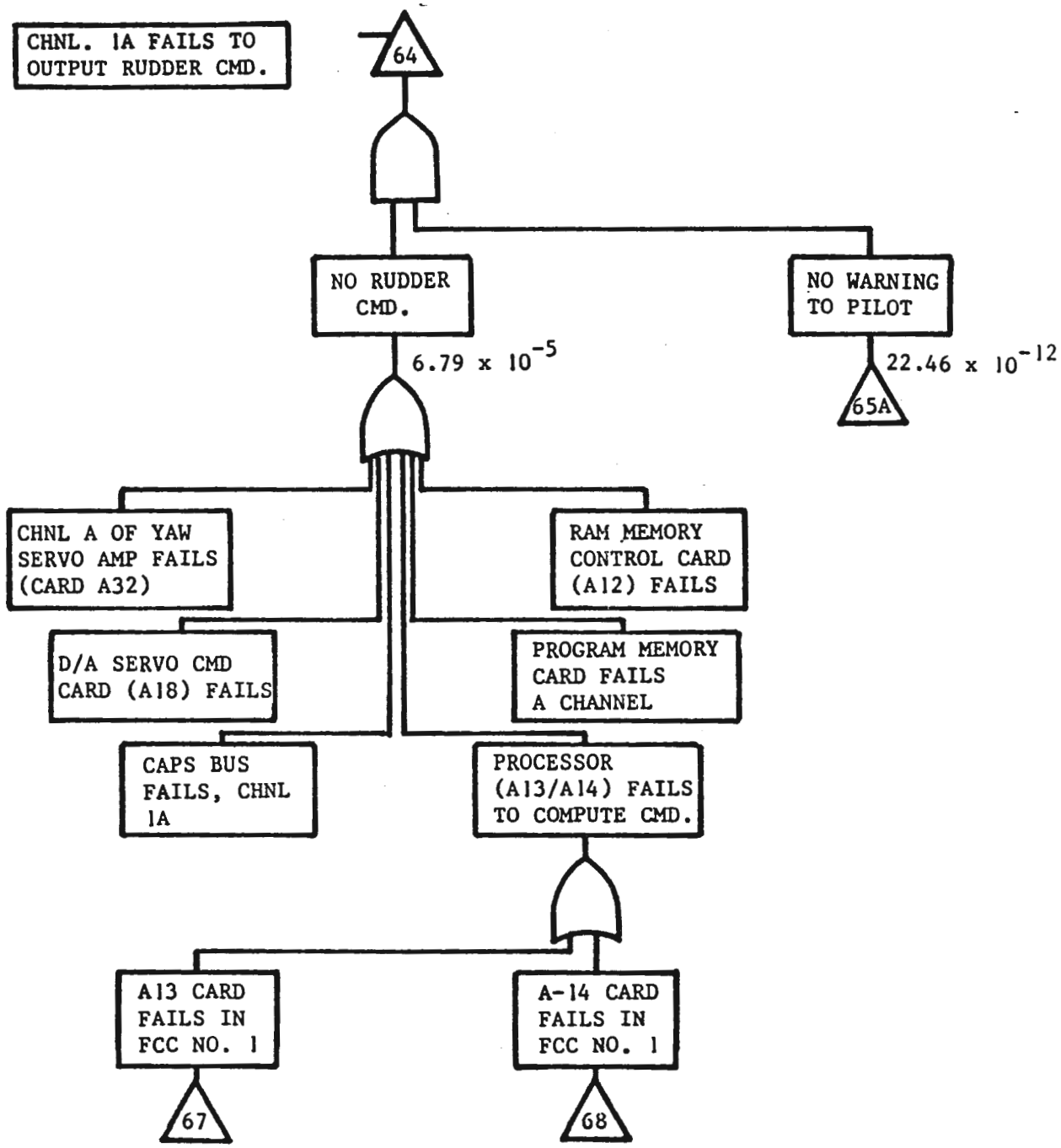



FIGURE 4.2-6. CHANNEL 1A RUDDER COMMAND

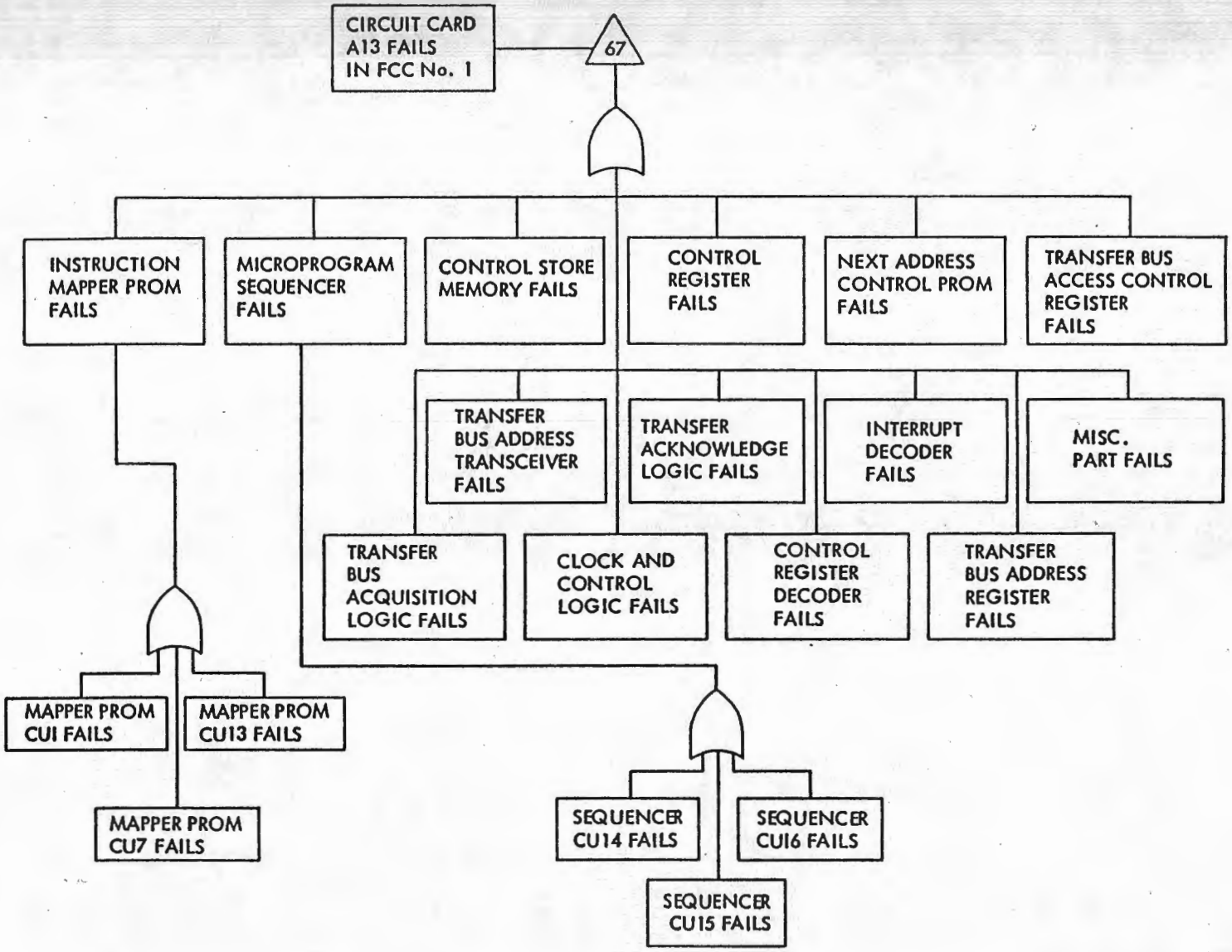



FIGURE 4.2-7. FCC PROCESSOR CONTROL CARD

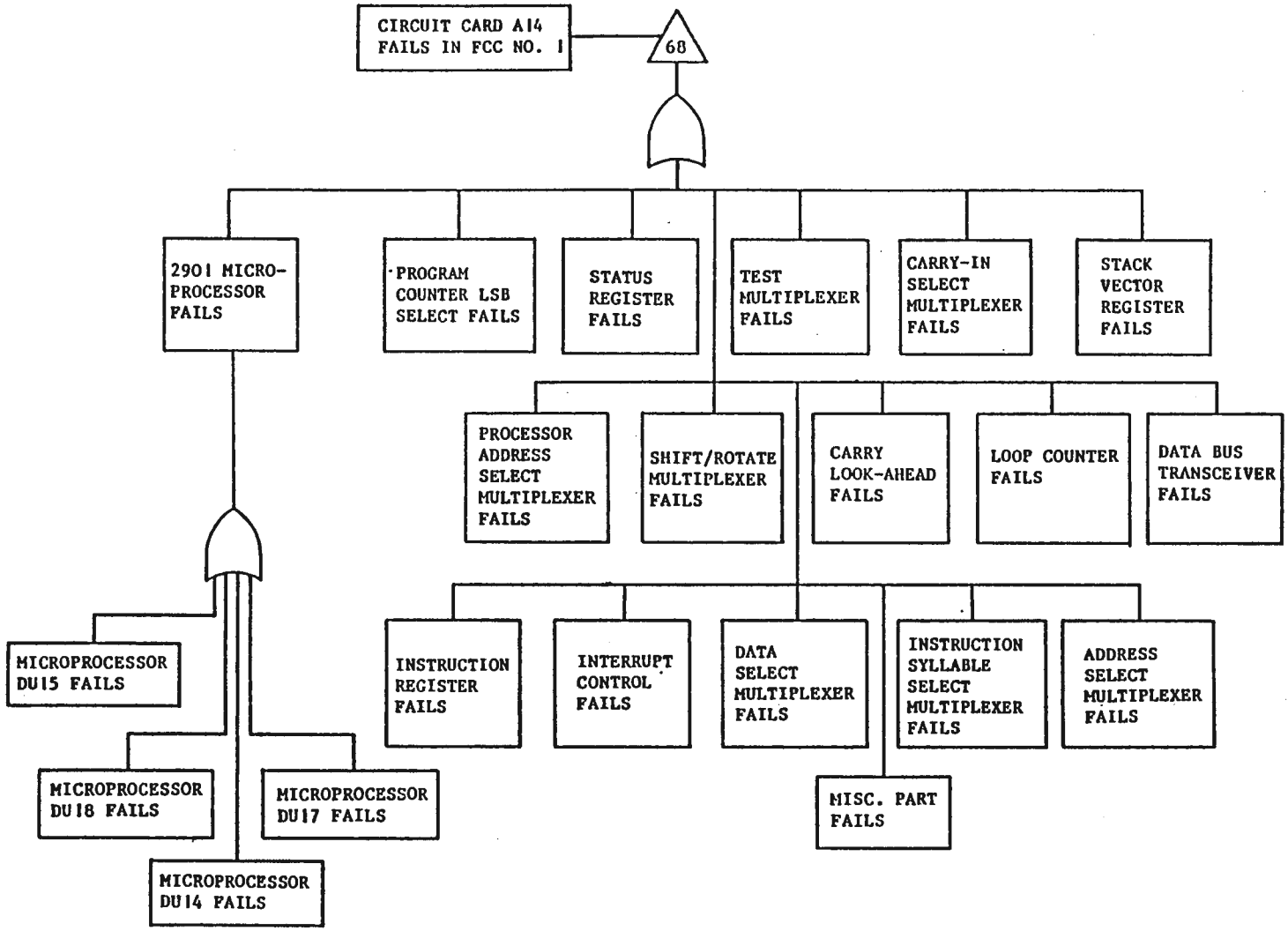



FIGURE 4.2-8. FCC PROCESSOR DATA PATH CARD

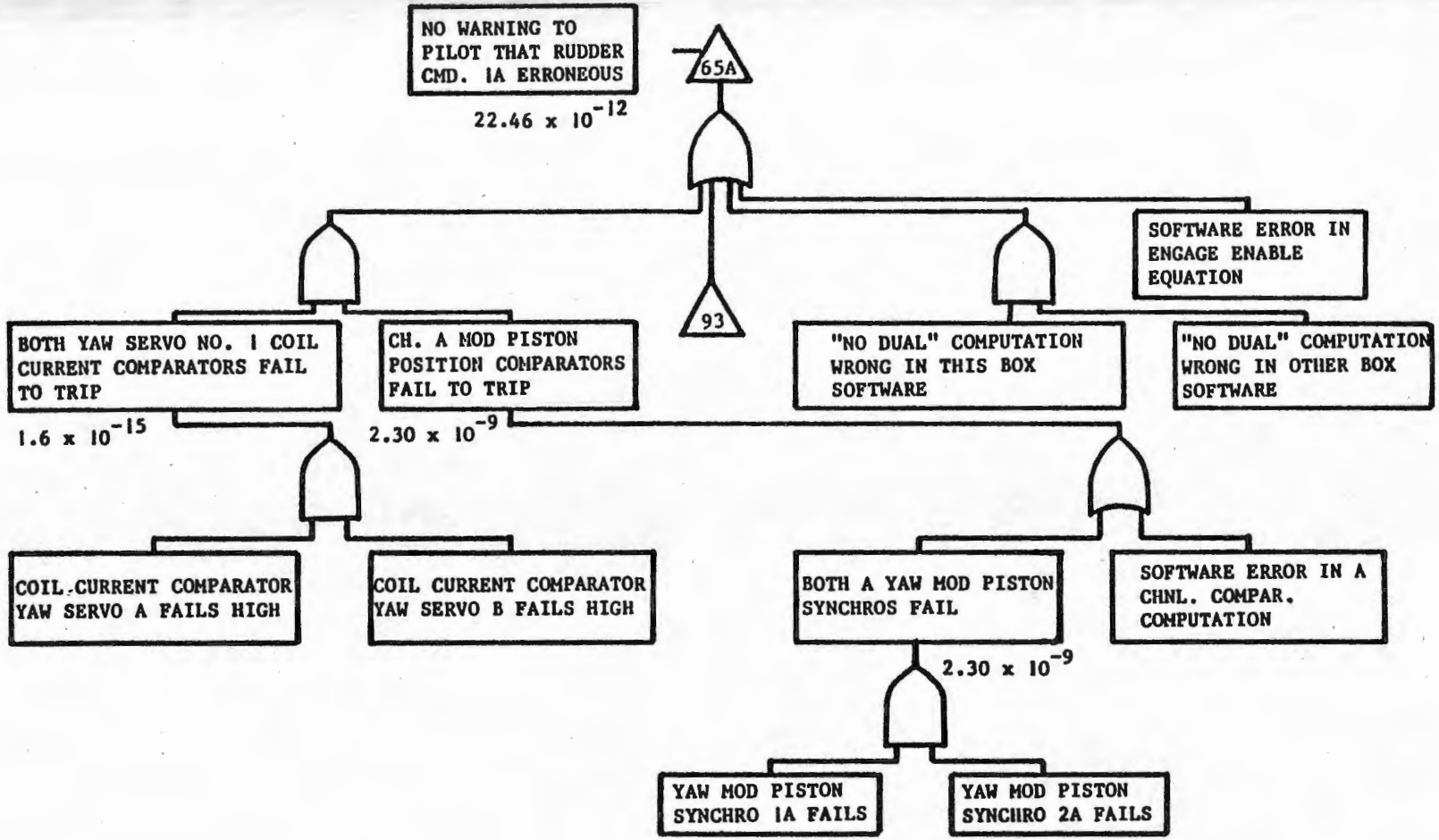



FIGURE 4.2-9. YAW AUTOPILOT SERVO COMMAND WARNING (1 of 2)

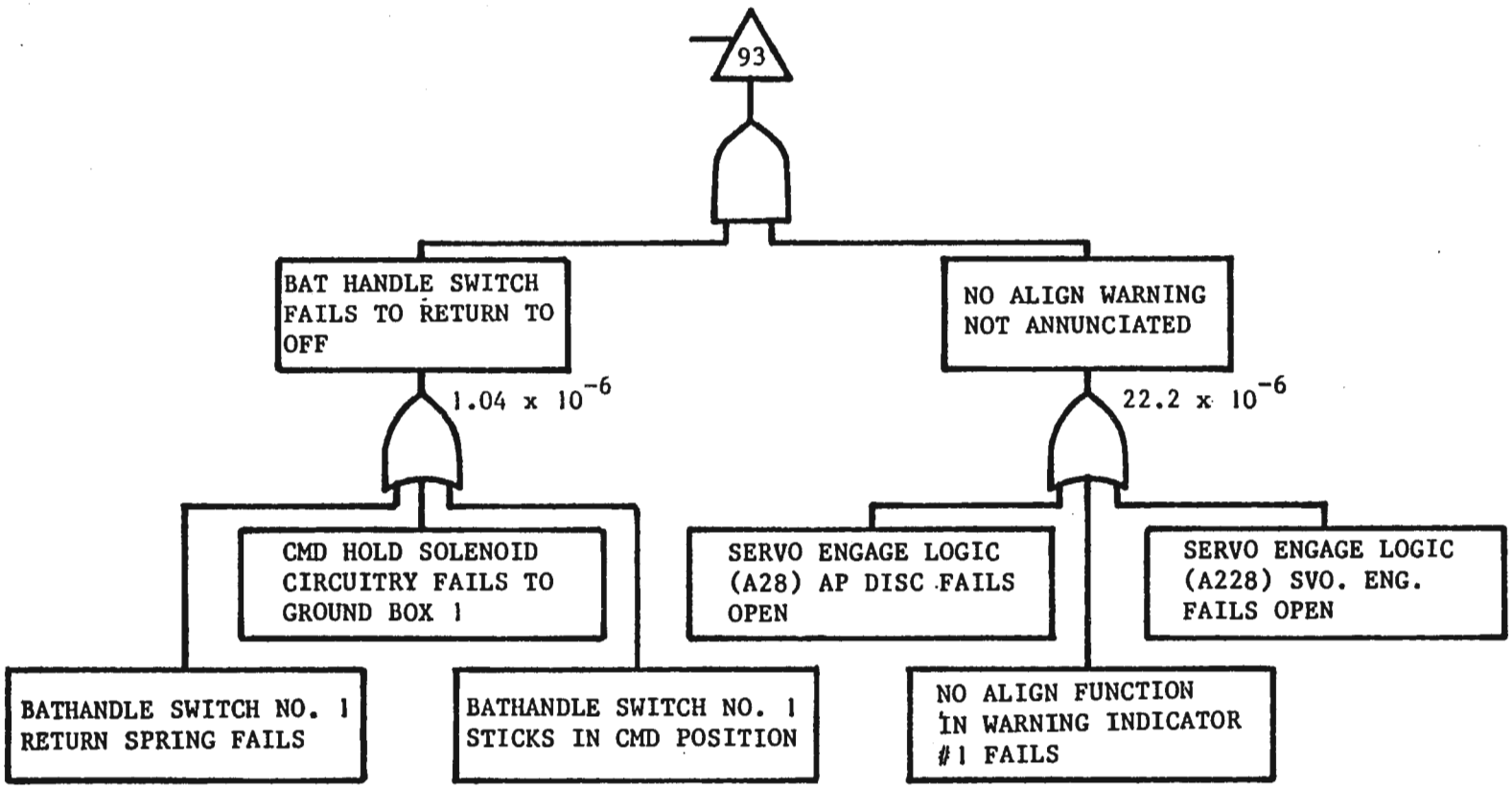



FIGURE 4.2-9. YAW AUTOPILOT SERVO COMMAND WARNING (2 of 2)

The third of the Level-2 events of figure 4.2-1 is that multiple failures occur during the crucial flight phase, and these occur in a combination which causes system failure. Figure 4.2-12 shows the initial development of this event to lower levels. Continuing this development produces a major branch of the fault tree, similar to but simpler than that for the second of the Level-2 events. It differs primarily in that the NO DUAL annunciation does not appear, because that particular warning is suppressed during the crucial phase. Since that major branch is similar to that already discussed, it is not described further here.

## 4.3. Quantitative Fault Tree Analysis

System failure probability was computed from the fault tree using the hardware failure rates presented in section 7. A failure rate of zero was used for each software function, since there is currently no acceptable way of predicting DFCS software failure rates (RTCA Document DO-178, section 2.2.1).

Considering hardware failure modes only, the probability of initiating the crucial phase with less than fail-operational capability and a second failure debilitating the system was calculated to be $2.46 \times 10^{-14}$. This is based on a flight time of 4.0 hours prior to the crucial phase and a crucial phase duration of 0.02 hours. The probability of the system failing because of multiple failures during the crucial phase was calculated to be $0.638 \times 10^{-9}$. This is based on a crucial phase duration of 0.02 hours. Table 4.3-1 compares the system failure probabilities obtained from the fault tree analysis with those obtained using the CARSRA reliability prediction program. The latter will be discussed in section 8.

The system failure probabilities computed are actually upper bounds on the actual failure probabilities. This is because the fault trees are based on the assumption, for many items, that all failure modes of the item render the item incapable of performing any of its functions. For example, certain buffers on the A26 Data Acquisition Card are used for sensor data which is not required for automatic landing. So at least some of the failures of these buffers would not prevent the card from correctly handling required data. However, the failure rates used in the analysis are for the entire card, including these buffers, so that the failure probability calculated for the card includes card failure modes which would not affect automatic landing.

The computation of the probability of system failure is accomplished by a process of "rolling up" from the lowest event levels. In computing the probability of an event's failure based on its subevents' failure probabilities, consideration must be given to several factors. Among these factors are the following:

- logical structure of the event: A OR B OR C, A AND B AND C, A OR B AND C, etc. (where A, B, and C are subevents)
- dependence or independence of subevents

TABLE 4.3-1. QUANTITATIVE RESULTS

| Probability Of | Fault Tree Result | CARSRA Result |
|---|---|---|
| Unannunciated Failure in Cruise and Second Failure in Landing | $2.46 \times 10^{-14}$ | $3.36 \times 10^{-14}$ |
| Multiple Failures in Landing | $0.64 \times 10^{-9}$ | $0.66 \times 10^{-9}$ |

The "rolling up" process continues from the lowest-level events through the top-level event. The actual number of computations can be large. Fault-tree algorithms use minimal cut sets (see section 4.2) to reduce the computational effort.

When dealing with these complexities, the approach to probability computation should always be conservative.

Poor event and subevent definition may result in excessive common failure modes within subevents. This results in having common failure modes simultaneously affect the operation of two or more separate systems, although these systems may otherwise be independent. Some common failure modes may be unavoidable. Examples of these are losses of electrical power, hydraulic power, or cooling. FMEA is useful in identifying common failure modes.

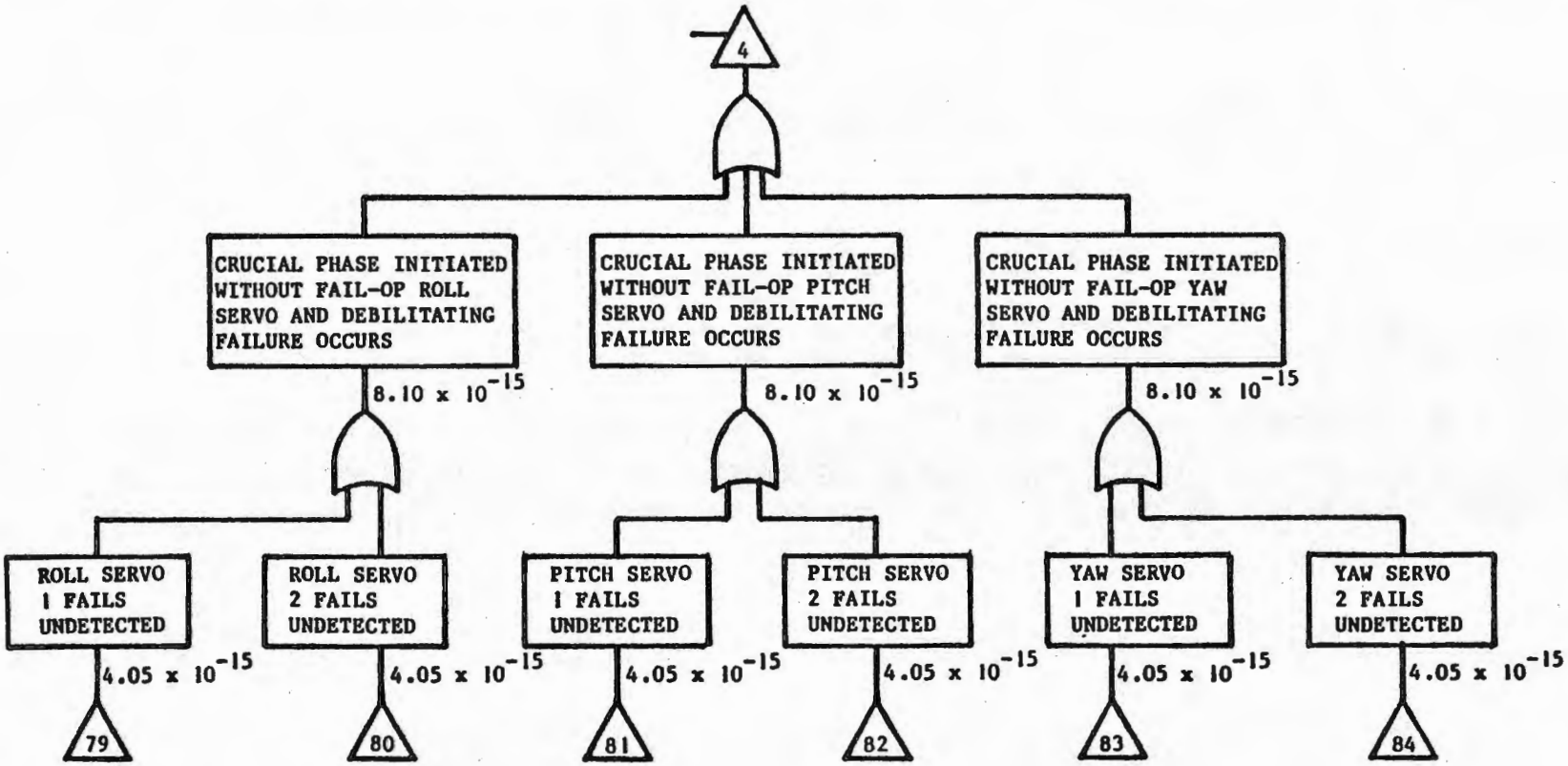



FIGURE 4.2-10. SERVO FUNCTIONS

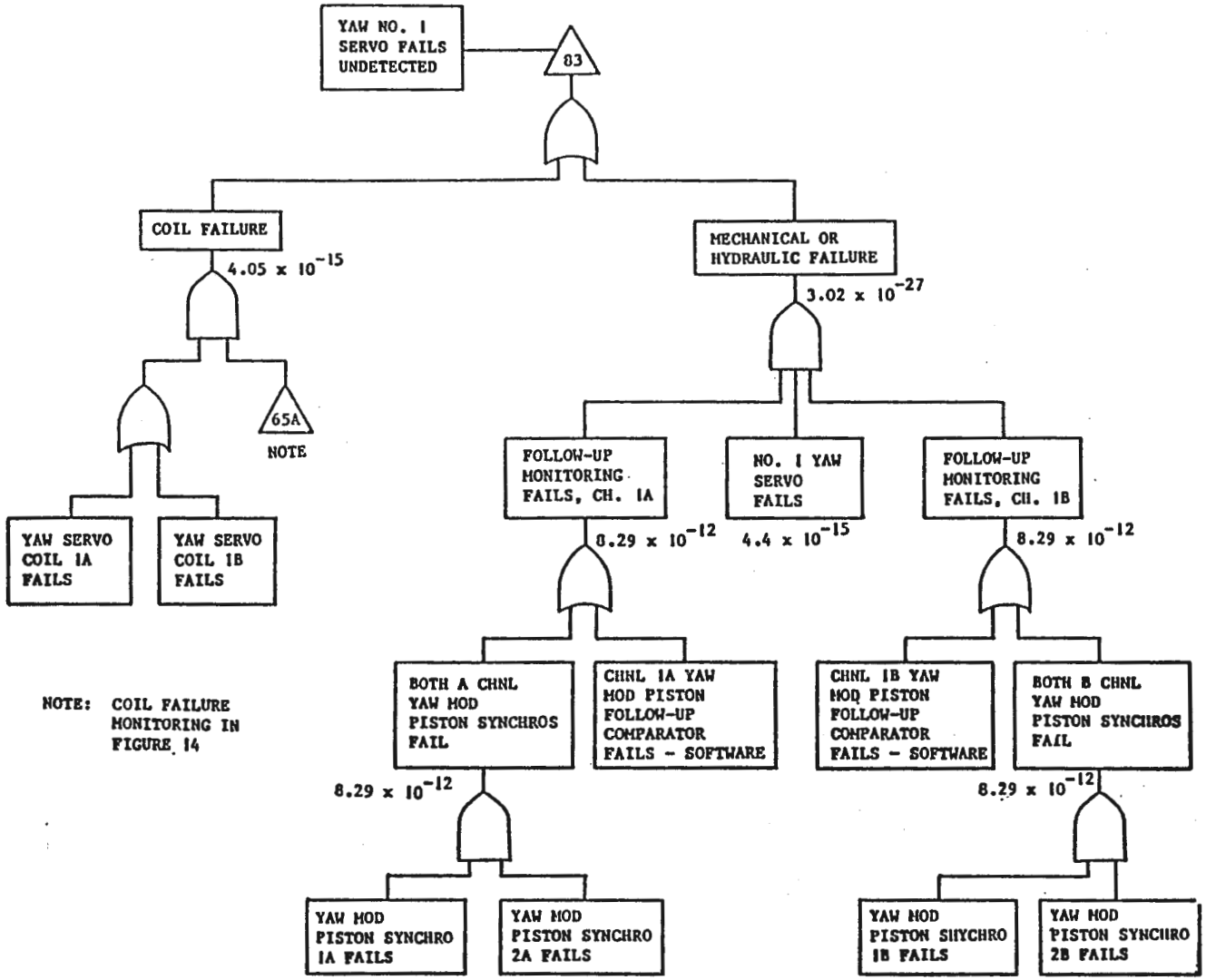



FIGURE 4.2-11. NO. 1 YAW AUTOPILOT SERVO

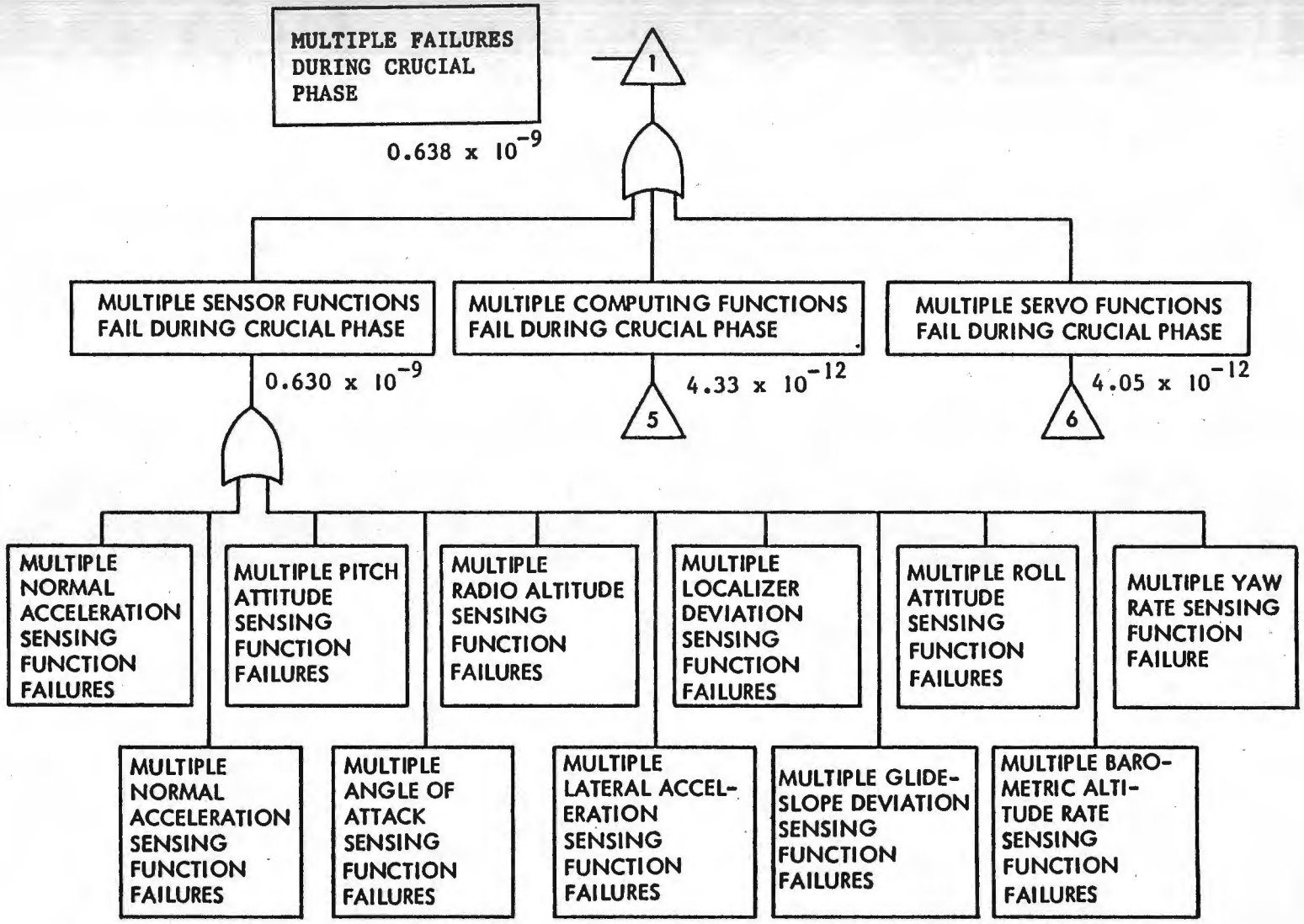



FIGURE 4.2-12. MULTIPLE FAILURES DURING CRUCIAL PHASE

# 5. FAILURE MODE AND EFFECT ANALYSIS

## 5.1. General Comments

The role of FMEA in IAA was specified in section 3.4. FMEA is a qualitative approach to be used in analyzing system failure. It is referenced in AC 25.1309-1 as a viable tool for both independent failure analysis and as an adjunct to a top-down tool, such as fault tree analysis.

AC 25.1309-1 defines FMEA as "an inductive bottom-up analysis which determines what happens to a system upon single failures of its individual parts." If the item being analyzed is a piece-part of a larger system, these identified failure modes will become the bottom events of a top-down analysis, such as fault tree analysis.

When using FMEA, all potential failure modes of a piece-part are identified. Each mode is then studied, and its impact on system performance is determined. This may be accomplished either empirically or by studying logic diagrams. The empirical approach may involve a technique such as fault insertion. Probabilities must be assigned for each of the failure modes. The development of these failure probabilities may involve some of the techniques discussed in section 7. MIL-STD-1629A is the best guidance document for FMEA.

The preceding paragraph defines the context in which FMEA was applied to RDFCS. Figures 4.2-7 and 4.2-8 depicted the subevents for two events related to the failure of Circuit Cards A13 and A14 in FCC #1. A number of these subevents dealt with microprocessor failure. Since the processor is a general purpose device, continuing the fault tree analysis, which is function oriented, was not desirable. An FMEA was performed instead.

In conducting the pin-level FMEA analysis of a processor, three factors were found to greatly reduce the effort. The first factor is that propagation of most faults under all conditions does not have to be considered. A single effect can usually be found which will totally debilitate the processor. For example, a faulted processor output pin will result in the processor trying to read about half of the data and machine level instructions from the wrong memory addresses. This will result in the coil current comparators tripping, sensor comparisons failing, and in the case of the RDFCS, the iteration monitor failing. In a system using checksums to monitor program memory integrity, these tests will fail.

The second factor which reduces the effort is that many pairs of faults will have the same effect. There are numerous instances of an output pin on one IC being connected only to one other pin. If either pin fails to open, the effect will be the same. Similarly, a ground fault in either pin will produce the same effect.

The third factor which reduces the effort is that there are many instances in which three pins are connected so that one output pin drives two input pins on different circuits. An open fault at each of the input pins can be evaluated first. An open fault at the output pin is then equivalent to both input pins failing open simultaneously. In most cases the effect is the "sum" of the effects of the input pins failing to open; that is, both effects occur. If both input pins are on the same chip, the effect of both being open is more likely to differ from the sum of the individual effects. See figure 5.1-1.

The effect of any of the three pins failing shorted-to-ground is the same in either of the two cases of figure 5.1-1.

Another frequently encountered condition involving three pins is two outputs connected to a single input (figure 5.1-2). In such a case, chips A and B will have three-state outputs, and one or both outputs should be in the high-impedance state at all times. An open fault on the output pin of chip A will then only affect chip C when A has its output enabled. Similarly, an open fault on the output pin of chip B will only affect chip C when B has its output enabled. An open fault on the chip C input pin will usually produce the sum of the effects of open faults on the two output pins. A ground fault on any of the three pins will have the same effect.

Still referring to figure 5.1-2, if a fault should occur, which results in both enable pins being in the enable state, there is a possibility of damage to the A or B chip. If one output is high and the other low, there could be a low impedence path to ground, through the output pins, which could burn out the A or B chip. This depends on the technology used in the individual chips. Frequently, the effect of the original ground fault can be judged to be a total processor failure whether or not the secondary damage occurs.

## 5.2. Application of FMEA to RDFCS

In the RDFCS analysis, three modules of the processor (figure 5.2-1) were considered at pin level.

- The instruction mapper Programmable Read-Only Memory (PROM), which consists of three PROM chips in parallel.
- The microprogram sequencer, which consists of three 2911 sequencer chips in parallel.
- The microprocessor module, which consists of four 2901A chips in parallel.

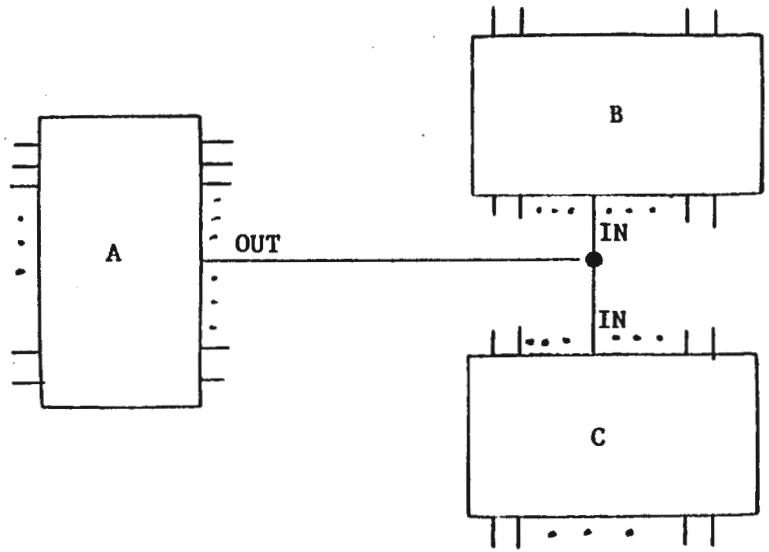
The instruction mapper PROM chips are read-only memory chips. There are two inputs to the chip: machine-level operation codes, and the depth of the stack maintained in the 2901 microprocessors. These are connected to the address pins of the mapper. The data stored in the PROM is the control store PROM address of the first microcode instruction required to execute the machine level instruction with the processor stack at a particular depth. The mapper output pins are only active at the beginning of a microcode sequence, at which time a chip enable signal is sent to the mapper from the next address control PROM.

The microcode address from the mapper PROM is routed to the microprogram sequencer module. This module generates a sequence of microcode addresses, beginning with the starting address from the mapper PROM. Some microcode routines involve jumps to a new address rather than sequential progression only. In such cases, the microprogram sequencer receives the jump address from the control store PROMs and resumes sequential generation of addresses.

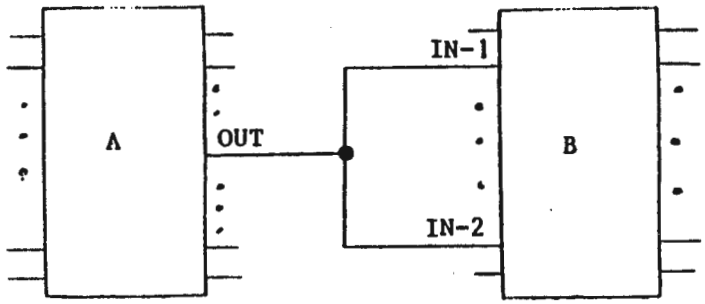
The microprocessor module is composed of four 2901A microprocessor chips. Each chip has a word size of four bits, so that the four chips in parallel are used to provide the processor 16-bit word size. This requires that carry signals be passed between 2901As during arithmetic operations. Other interconnections between 2901As are used for data shift operations. The 2901As are controlled primarily by control signals from the control store PROMs in conjunction with the outputs from various registers.

The FMEA, summarized in table C-1 (appendix C), considered three types of pin-level faults: open, grounded, and shorted-to-supply voltage. In most cases, the effect of a fault can be assessed by using the chip logic diagrams, a description of chip/module functions, and the schematic diagrams. The schematic diagrams are reproduced in appendix D.

The effect of certain pin faults cannot be determined by analysis using just the above information. In particular, the contents of specific PROM addresses was required or the machine-level code was needed along with the microcode sequences and addresses. Alternatively, the faults could be inserted and the effects observed. The Fault Insertion Approach was taken in this analysis. The results are presented in section 6. For example, it was known that failure of one of the processor pins, used in data shifts (R0, R3, Q0, or Q3 stuck high or low), would result in an immediate disconnect if certain integer words made up of packed Boolean variables were shifted. It was determined from the available information that such shifts might occur, but it was not definite. Similarly, if certain fixed-point numbers were shifted during computation, the commands to the servos would be in error, and the coil current comparators would trip. While both left and right shifts are normally used in multiplication algorithms, it could not be determined that a stuck shift bit would definitely cause such a trip. When the faults were actually inserted, the processor stopped immediately. ("Immediately," as viewed by the human observers.) In this way, fault insertion confirmed the overall effect, a massive processor failure and disengagement of the servos. The exact mechanism by which it occurred was not determined.

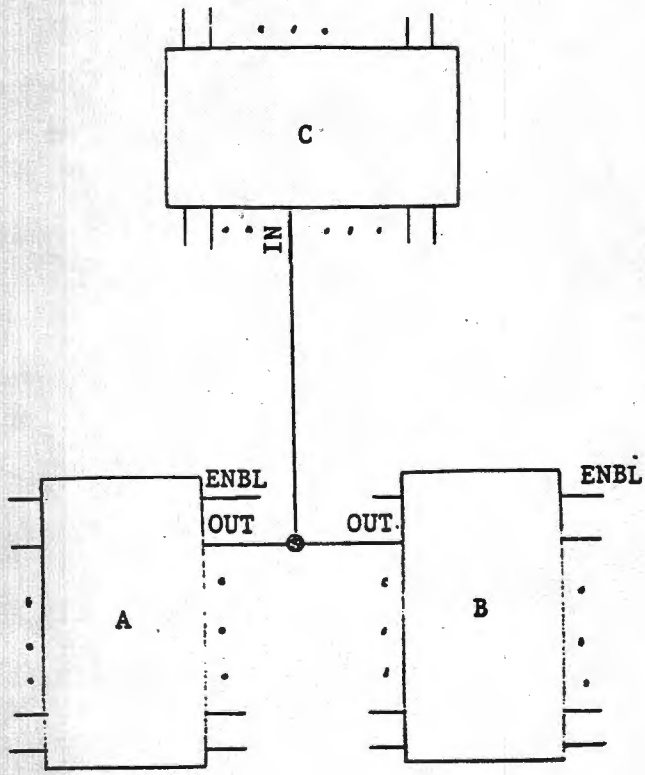



EFFECT OF OUTPUT OPEN ON "A" USUALLY SAME AS EFFECT OF INPUT ON "B" OPEN PLUS EFFECT OF INPUT ON "C" OPEN.




EFFECT OF OUTPUT OPEN ON "A" OFTEN NOT SAME AS EFFECT OF IN-1 PLUS EFFECT OF IN-2.

FIGURE 5.1-1. ONE OUTPUT, TWO INPUT CONDITIONS




BOTH "A" AND "B" ENABLED SIMULTANEOUSLY MAY DAMAGE CHIP.

FIGURE 5.1-2. TWO OUTPUT, ONE INPUT CONDITION

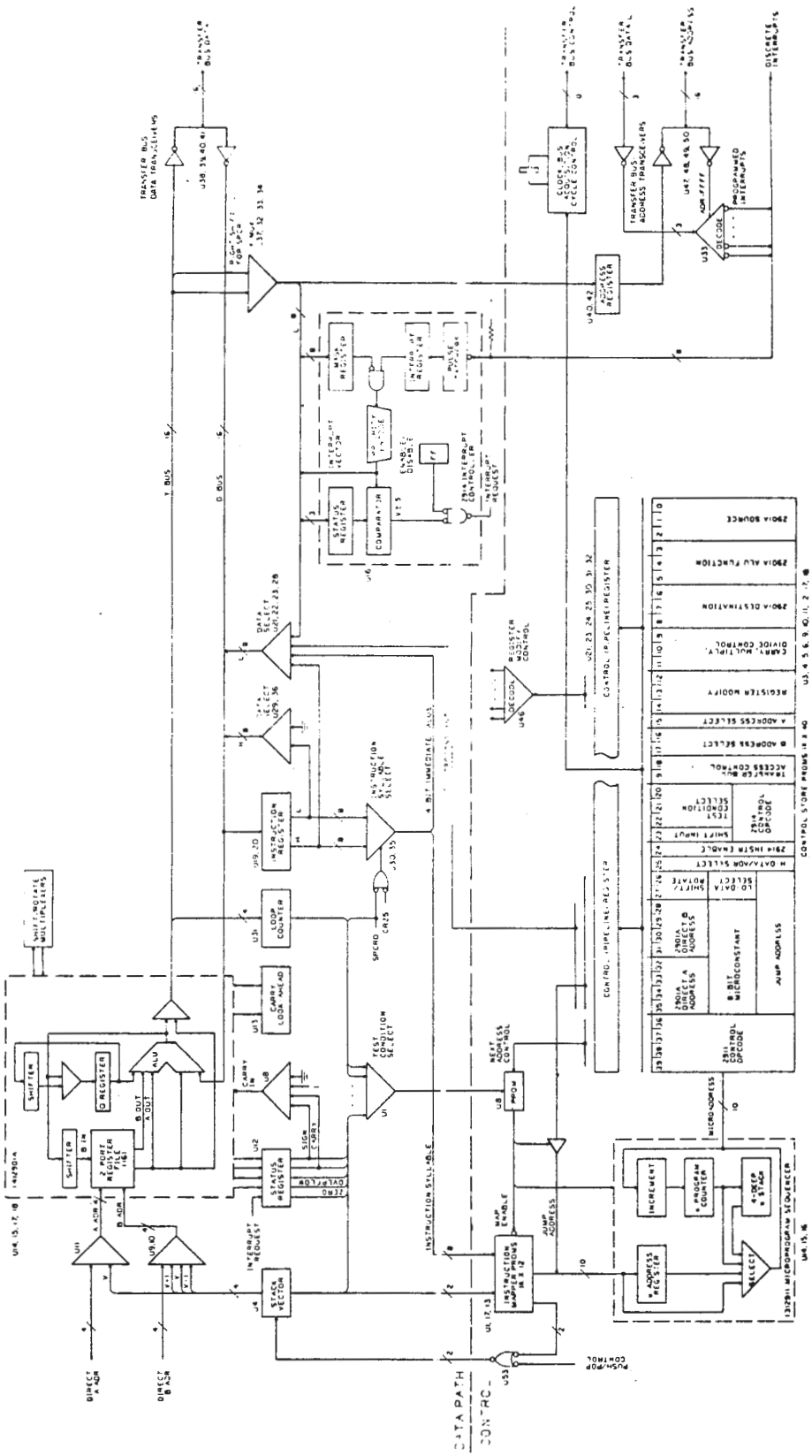


FIGURE 5.2-1. PROCESSOR BLOCK DIAGRAM

# 6. FAULT INSERTION

## 6.1. General Comments

The role of fault insertion in IAA was specified in section 3.5. This technique is required whenever the effect of certain pin faults cannot be determined analytically. The effect of the fault is determined by observing it. Fault insertion may also be used to confirm the effect of faults diagnosed analytically. Another use of fault insertion is to verify fault detection and annunciation systems. In the RDFCS analysis the technique was used for all three purposes.

## 6.2. Application of Fault Insertion to RDFCS

The RDFCS simulator at NASA-Ames was used to insert the faults shown in table E-1 (appendix E). The faults were of two general types: component level faults and IC pin faults. The component level faults were inserted using the FCC breakout panels (figure 6.2-1), the SSP (figure 6.2-2), and the MDICU. Single-sensor faults are those numbered 1 through 19 in table E-1 (appendix E).

Faults representing a dead sensor or a broken wire from the sensor to the FCC were inserted by pulling the appropriate jumper plug at the break-out panel. Faults representing missing sensor validity discretes were also inserted in this way, although they can also be inserted via the DSP (figure 6.2-3). Sensor hardovers and ramps were inserted using the MDICU. Servo faults were inserted using the SSP.

For monitoring the processor detection of sensor faults, the CTAs were used. One of the CTA address windows was set to the address of the Executive Failure (Status) Word (EFW) in each computer channel. The EFW is a 16-bit word with each bit representing a discrete piece of information. There is one EFW for each sensor type in each computer channel. The four low-order bits (0-3) represent, respectively, Executive Failure of My A (EFMA), My B (EFMB), Other A (EFOA), and Other B (EFOB) sensor signals. The data window of the CTA shows the status of the EFW as four hexadecimal characters, with the right-most character representing the bits of interest, 0-3.

The effect of a sensor signal being detected as bad by the software sensor monitor is that certain bits are changed from 0 to 1. With no failures detected, EFMA, EFMB, EFOA, and EFOB are all 0, which is represented in hexadecimal notation as 0. (0000 binary = 0 hexadecimal). When the number 1 sensor of a triple sensor complement is detected to have failed, bit 0 (EFMA) is set to 1 in both channels of FCC No. 1. Bit 1 is also set to 1 so that the comparison monitoring will work properly on the two remaining sensors. The EFW low-order bits will then be 0011, which is 3 in hexa- decimal. The net effect, then, of the number 1 sensor of a triple-sensor set failing is that the value displayed in the CTA window changes from 0000 to 0003. The left three

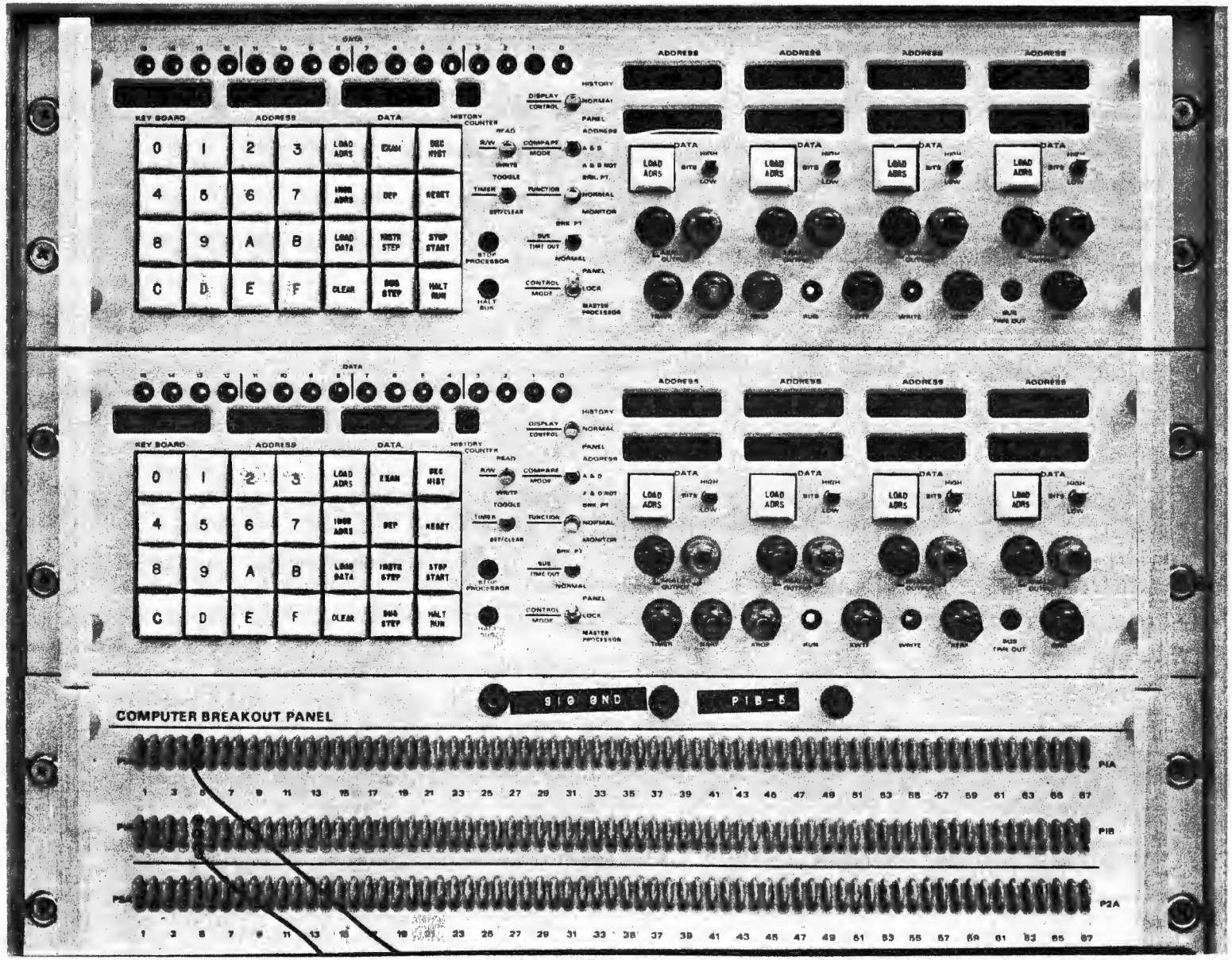



FIGURE 6.2-1. CTA AND COMPUTER BREAKOUT PANEL

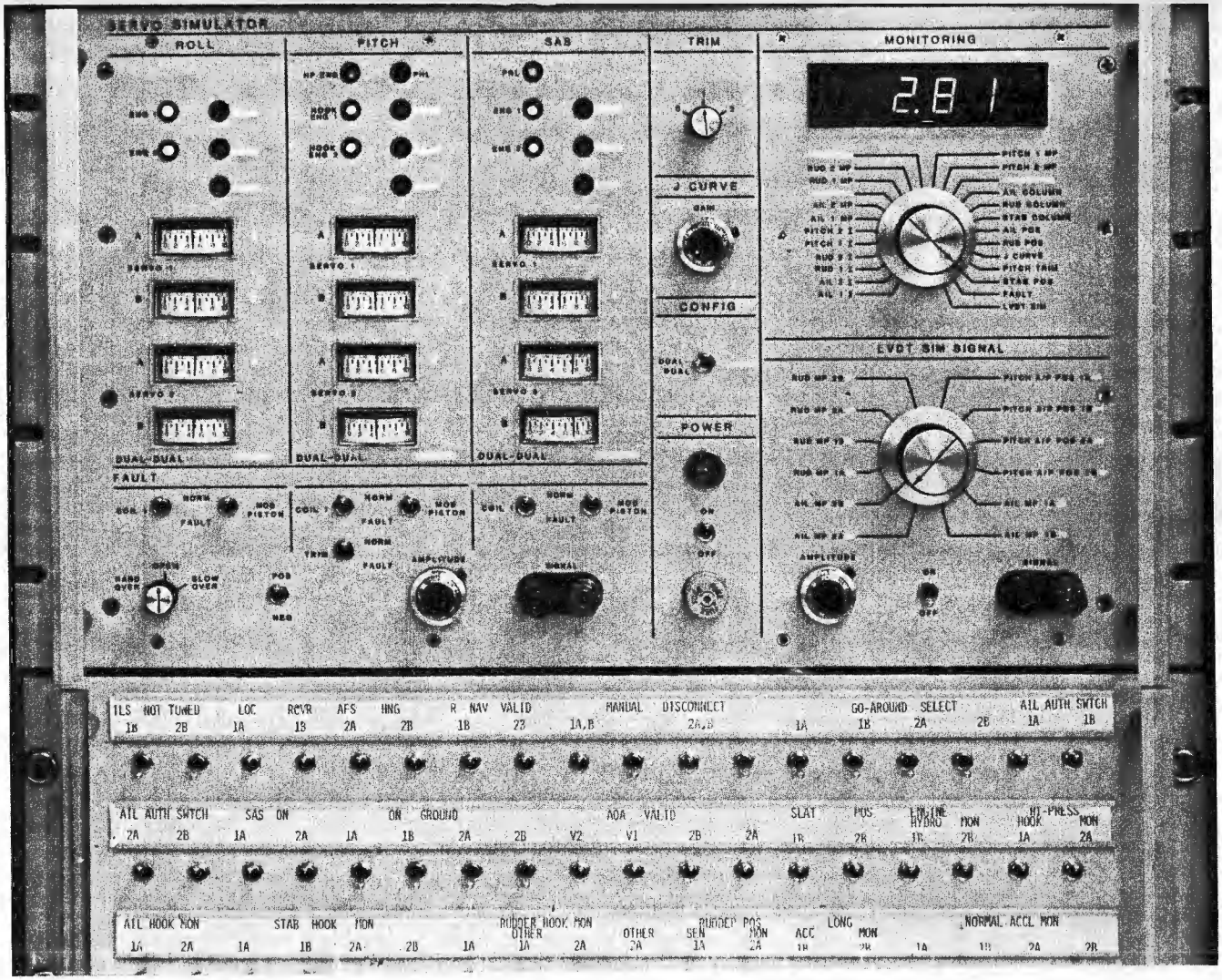



FIGURE 6.2-2. SERVO SIMULATOR PANEL

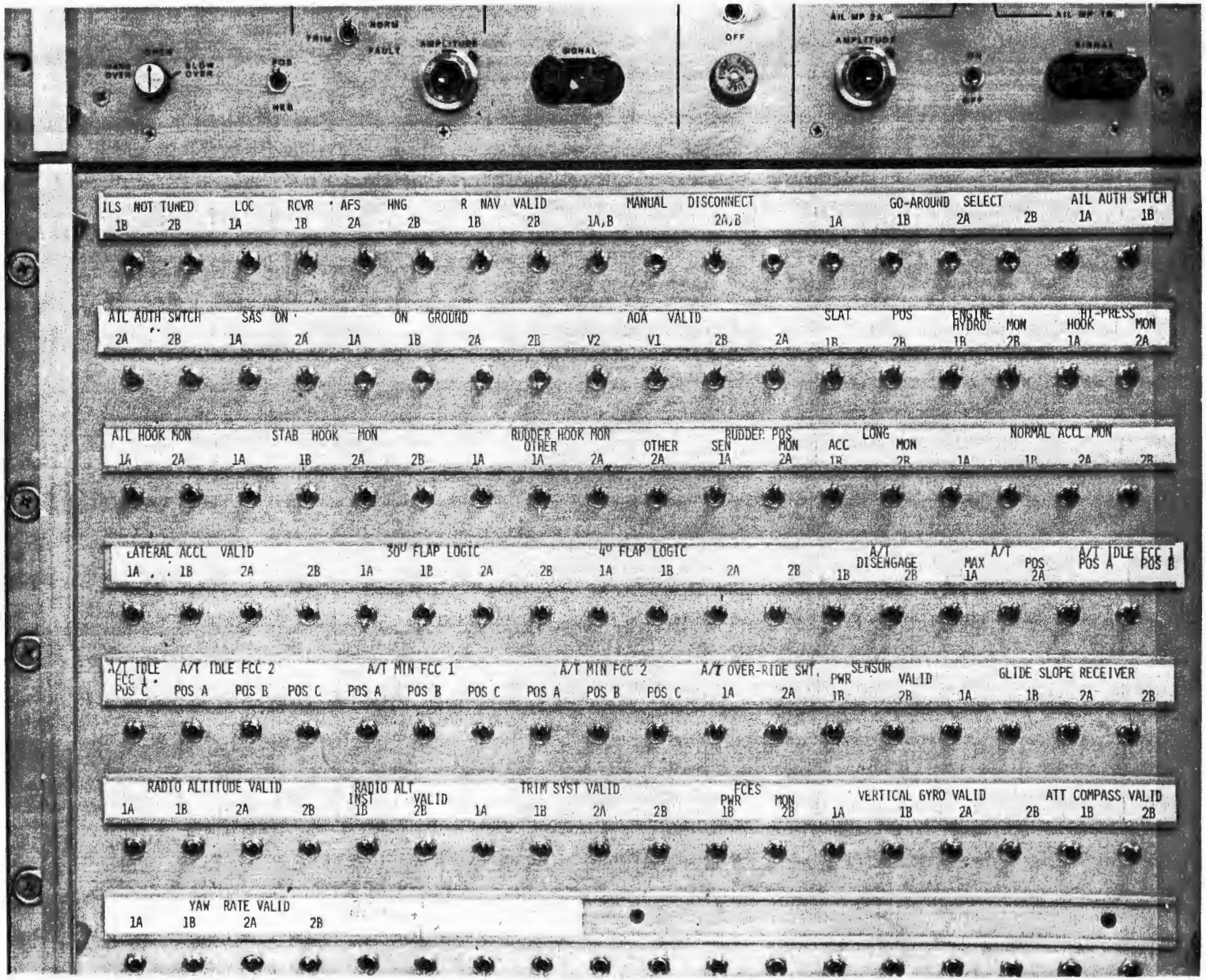



FIGURE 6.2-3. DISCRETE SWITCH PANEL

hexadecimal digits each remains at 0, since each of the corresponding binary bits (4-15) of the EFW remains at 0.

Fault cases 1 through 8 were used to show that the software sensor monitor subroutine is implemented correctly in the RDFCS by subjecting it to a number of different faults in the same sensor type. These cases were also used to show that the results of the sensor monitoring are accounted for in the implementation of the NO DUAL equation, which is also in software. Cases 9 through 16 were then used to show that the voter is involved for various sensor types. Rigorous validation of the system by testing would require that faults be inserted for all sensor types used in automatic landing. In this analysis, performed for illustrative purposes, the full complement of sensor types was not faulted.

In case 2E, NO DUAL did not annunciate even though the fault was inserted with the airplane inbound to the ILS beam intercept point. It is believed to be the result of the inbound leg being flown at an unrealistically low altitude, so that the airplane did not track the glideslope beam for 25 seconds before passing through a 150 ft altitude. A review of the NO DUAL annunciation logic shows that this is the most likely cause, since AP. ONEFAIL was set to true. Low approaches (1500 ft) were being simulated in the interest of time. Approach altitude was subsequently raised to 2000 ft.

Faults 17 through 19 were used to confirm the servo monitoring and the tie-in of the servo monitor outputs to the NO DUAL and disconnect logic. The servo monitors, in particular the coil current comparators, are quite important in ensuring that the airplane does not enter the crucial phase with a faulty computer or servo.

Fault cases 43 through 45 were used to confirm that the FCCs will both disengage upon loss of the second sensor, with the AP.DISC warning displayed, in accordance with the system description.

At the IC pin level, a number of open and ground faults were inserted to confirm the FMEA results of section 6. For this activity, one of the FCCs was removed from the pallet, and the card containing the chip to be faulted was extended for access as shown in figure 6.2-4.

Open pin faults, Cases 20 through 23, were inserted by using multiple sockets between the chip and the circuit card, with a jumper wire replacing the normal pin-to-socket connection. Each fault was inserted by physically pulling the jumper to open the connection. This is a slow procedure, since the chip must be removed and the jumper wire rigged on the desired pin.

The chip and sockets must then be installed and the processors brought back up. This method of inserting open pin faults is only marginally satisfactory. It would be much easier to do if a stack of five or six sockets could be used between the chip and the circuit card. However, the processor will not come up with more than three sockets stacked. The longer electrical paths resulting from the use of the extender card apparently come close to exhausting the avail-

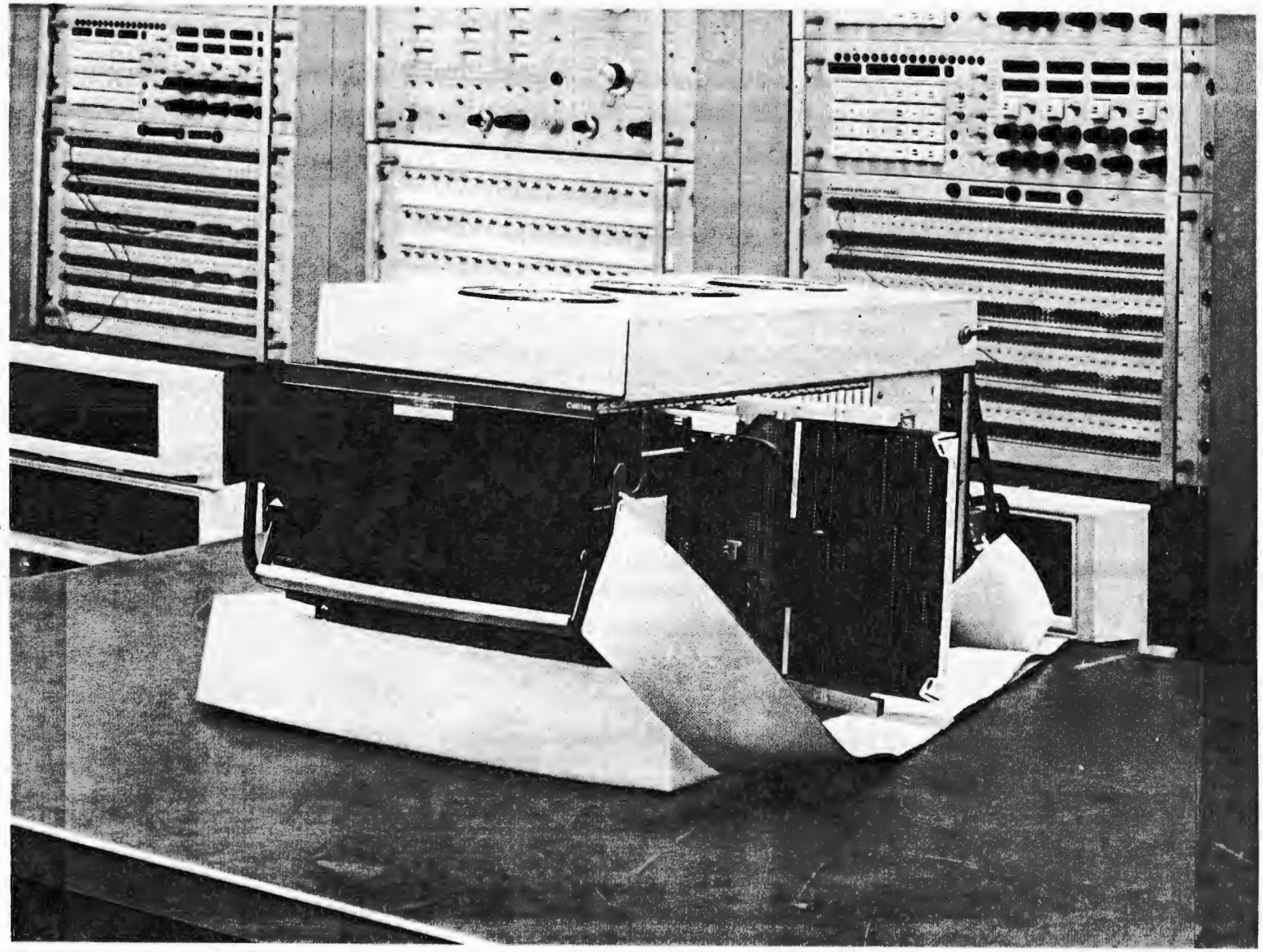


FIGURE 6.2-4. FCC WITH FLIGHT CARD EXTENDED

able tolerance in the timing of the individual microsteps, and the extra path length and capacitance caused by more than three sockets disables the processor.

Grounded pin faults are much easier to insert, since the chip does not have to be removed to set up each case. The processor does have to be brought back up each time, but this is a fairly rapid step. Before each fault was inserted, the data sheets from the chip manufacturer were reviewed (along with the card schematics) to determine that the fault would not damage any chips. No chips were damaged by the ground faults. The ground pin faults are cases 24 through 42 in table E-1.

The chip pin faults all disabled the processor, with the exception of open pin fault 21. This fault involves a pin of a quad 2-input NOR gate. The fault had no effect on the processor operation.

### 6.3. Fault Insertion Results

The faults inserted in the RDFCS simulator achieved the desired results in the assurance assessment of this analysis. More importantly, this confirmed that fault insertion is capable of providing the results required in the IAA. Specifically, the faults inserted confirmed (1) that the NO DUAL warning appears when it should, (2) that all sensor types faulted and required for automatic landing are monitored, (3) that the servo monitoring functions correctly, (4) that the effect of pin-level faults in the processor is in agreement with the FMEA, and (5) that fault insertion is a reasonable way of resolving uncertainty of the effect of open and grounded pin faults in digital hardware. While these results were obtained on a particular system, the approach is judged to be viable for validating other digital systems.

# 7. FAILURE RATE DEVELOPMENT

## 7.1. General Comments

While the development of failure rates is not a safety analysis technique, it is a task that is fundamental to the generation of system probability of failure. In order that fault tree analysis or a statistical prediction tool be able to produce a measure of reliability, individual failure rates must be known. This section discusses the different sources that were used for obtaining failure rates for the RDFCS analysis. The event trees used in the fault tree analysis in section 4 reference these failure rates; they were used in the failure probability computation. Likewise, the rates will be referenced in the reliability analysis found in section 8.

## 7.2. Rationale for Developing Failure Rates for RDFCS

The failure rates for servos, sensors, and indicators were taken from the database maintained by the Lockheed-Georgia Company Reliability Engineering Department. They are composite values for representative components of comparable complexity and construction.

The failure rates for the ICs of the Data Path and Control Cards were estimated using the formulas and tables of Military Standardization Handbook 217C. The formulas provide a means of accounting for a significant number of factors:

- Device technology
- Device complexity
- Junction temperature
- Package technology
- Application environment (voltage)
- Usage environment
- Quality level

For example, the equation for the failure rate of a monolithic bipolar device is:

$$f = K_0 \left[C_1 K_T K_V + (C_2 + C_3) K_E\right] K_L$$

Where:

f is the device failure rate

$K_Q$ is the quality factor

$K_T$ is the temperature adjustment factor for junctions

$K_V$ is the voltage derating stress factor

$K_E$ is the application environment factor

$C_1$ and $C_2$ are complexity factors based on transistor count

$C_3$ is a complexity factor based on package technology and number of pins

$K_L$ is a learning factor

The quality factor, $K_Q$, has a value of 1 for devices procured in full accordance with MIL-M-38510, Class B requirements. This value was used for all circuits in this project. It should be noted that the quality factor is a direct multiplier, so that the predicted rate is proportional to it. More or less stringent quality factors can therefore greatly influence the prediction for any individual circuit, circuit board, or entire component.

Junction temperatures are used in determining the adjustment factors $K_T$. The junction temperature is ambient temperature plus the differential resulting from power dissipation through the case. An ambient of 60° was used, with the power dissipation taken from the circuit specification.

The voltage derating stress factor is 1 for the bipolar circuits used in the CAPS processor. The application environment factor is 3.5 for the airborne, inhabited, transport environment of the aircraft underdeck avionic rack. Failure rates for the circuit cards of the FCCs were obtained by summing the failure rates for the card and its components. Table 7.2-1 summarizes the failure rate prediction for the A13 control card. Failure rates for the other cards are shown in table 7.2-2.

Table 7.2-3 presents failure rates for the system components other than the FCCs.

In using these rates in the fault tree and CARSRA analyses, an adjustment was frequently required to include only a portion of the rate, since only certain failure modes are of interest. For example, each dual current comparator has a predicted failure rate of 0.03. Each half of the comparator is given a rate of .01 for the failure mode of failing to trip when the threshold difference is exceeded. This is a very conservative rate for this mode.

TABLE 7.2-1. FCC CONTROL CARD FAILURE RATE

| ITEM | FAILURE RATE* |
|---|---|
| Integrated circuits | 1.788 |
| Resistors | .0018 |
| Capacitors | .224 |
| Oscillator | .25 |
| Coil | .0007 |
| Circuit Board | .023 |
| Edge Connector | .16 |
| Control Card Total | 2.45 |

* All failure rates in failures per million hours.

TABLE 7.2-2. PREDICTED FCC CARD FAILURE RATES

| CARD NO. | FAILURE RATE* |
|---|---|
| A1 Power Supply Monitor | 0.555 |
| A2-A5 PROM Card | .809 each |
| A6 Power Supply Monitor | .55 |
| A7-A10 PROM Card | .809 each |
| A11 Terminator/Test Access | .555 |
| A12 RAM Memory Control | 1.18 |
| A13 CAPS Control | 2.45 |
| A14 CAPS Data Path | 1.98 |
| A16 Cross-Channel Receiver | .70 |
| A17 DITS Transmitter | 1.75 |
| A18 D/A Servo Command | 1.75 |
| A19 Terminator/Time Synch | 1.40 |
| A20 Discrete Output | 2.79 |
| A21 Data Transmitter/Receiver | .70 |
| A22 Serial Digital Input No. 1 | 1.65 |
| A23 Serial Digital Input No. 2 | 1.80 |
| A24 Autoland Sensor Input | 1.80 |
| A25 Cruise Sensor Input | 1.12 |
| A26 Data Acquisition | 1.20 |
| A27 Discrete Input | 1.30 |
| A28 Servo Engage Logic | 2.61 |
| A29 Cross Channel XMTR | 1.20 |
| A30-A32 Servo Amplifier | 3.00 |
| A33 Speed Servo Amp | 1.70 |
| A300 Speed Command XMTR | 1.70 |
| A400 Power Supply | 21.0 |
| A500 Power Supply | 21.0 |

* All failure rates in failures per million hours

TABLE 7.2-3. FAILURE RATES FOR MAJOR RDFCS COMPONENTS

| COMPONENT | UNIT FAILURE RATE* |
|---|---|
| Pitch Angle Gyro | 303 |
| Roll Angle Gyro | 303 |
| Yaw Rate Gyro | 200 |
| Accelerometer | 74 |
| Radio Altimeter | 756 |
| ILS Receiver | 252 |
| Air Data System | 167 |
| Roll Autopilot Servo | 14 |
| Pitch Autopilot Servo | 15 |
| Yaw Autopilot Servo | 14 |
| EH Valve Drive Coil | 1.0 |
| LVDT | .72 |
| Dual Current Comparator (Hardware) | .03 |
| Warning Annunciator (per function) | 8.3 |

* These are not actual failure rates for any particular airplane or for any single component produced by a particular manufacturer. They are representative rates determined by a review of generic component types on a number of airplane models in a variety of commercial and military applications. All failure rates in failures per million hours.

# 8. RELIABILITY PREDICTION USING CARSRA

## 8.1. General Comments

AC 25.1309-1 emphasizes the need for probability analysis in determining system reliability. This circular defines reliability to be "the probability that a system, subsystem, unit or part will perform its intended function for a specified interval under stated operational and environmental conditions." The following three probability classifications are specified:

- Probable: Probable events may be expected to occur several times during the operational life of each airplane. A probable event has an occurance probability on the order of $1.0 \times 10^{-5}$ or greater.

- Improbable: Improbable events are not expected to occur during the total operational life of a random single airplane of a particular type, but may occur during the total operational life of all airplanes of a particular type. An improbable event has a probability on the order of $1.0 \times 10^{-5}$ or less.

- Extremely Improbable: Extremely improbable events are so unlikely that they need not be considered to ever occur, unless engineering judgment would require their consideration. An extremely improbable event has a probability on the order of $1.0 \times 10^{-9}$ or less.

Aircraft functions fall into three categories: Non-essential, Essential, and Critical. Section 3.1 discusses these three categories; AC 25.1309-1 provides a better discussion. Errors in a non-essential function may be probable. Errors in an essential function must be improbable. Errors in a critical function must be extremely improbable.

An IAA is performed on a particular aircraft system performing a function, e.g. the RDFCS. The RDFCS becomes a critical system when the airplane is executing an automatic landing, during the last 150 ft of descent.

## 8.2. RDFCS Reliability - Fault Tree Analysis

As demonstrated in section 4.3, the probability of system failure may be computed from the fault tree analysis. The two scenarios that could result in the aircraft's improperly accomplishing or losing the critical function performed by the RDFCS had a combined probability of $0.638 \times 10^{-9}$. A third scenario, that conditions encountered are outside the system design requirements or control laws are deficient, was not considered (see section 4.2). The probability of failure of the RDFCS is "extremely improbable." The concern is that an error was made producing a low system failure probability and that the error remains undetected. An independent confirmation of this reliability is needed.

## 8.3. RDFCS Reliability - CARSRA

CARSRA is an analytical reliability prediction program used in this IAA to obtain the probability of system failure (Bjurman, et al., 1976). In this analysis, the probability of failure is only considered during the crucial flight phase, which has a duration of 0.02 hours. Although CARSRA was used in this assessment, some other method (except fault tree analysis) could be used for the verification. These alternatives were discussed in section 3.5.

## 8.4. CARSRA Application

CARSRA is an application of stochastic processes. Stochastic processes concern sequences of events governed by probabilistic laws. Applications exist in many physical and social sciences, as well as engineering. The explosion of system failure into its elemental causes with probabilistic relationships (fault tree analysis) suggests that viewing the error tree as a stochastic process may be an aid in analysis. The reader need not have a strong statistical background in order to understand this discussion. There are a large number of possible references on stochastic processes. Karlin (1966) was used in the preparation of this tutorial.

### 8.4.1. Configuration Description

The CARSRA computer program requires that the user provide an input description of the system under analysis. Three levels of organization are implicit in the CARSRA inputs; the user must adhere to these levels. At the top level is the system, in this case the RDFCS. System failure probabilities constitute the primary output provided by CARSRA. The intermediate level is comprised of stages. Each stage consists of one or more identical modules, which are at the lowest level. In the RDFCS, each sensor is a module, and like sensors form stages. For example, each of the three Normal Accelerometers (NA) is a module, and the three NA together comprise a stage. CARSRA determines system reliability as a function of time; the success of the system is based on dependencies between stages, functional redundancy, and module redundancy.

### 8.4.2. Markov Models

A Markov process is a type of stochastic process. Chapter 1.3 of Karlin (1966) contains a mathematical definition of both stochastic and Markov processes. A Markov model is a Markov process used to describe a particular system or subsystem. Markov models were selected by the CARSRA developers as a major part of the program's analytical framework. The following discussion of these models includes some material on applying CARSRA to systems other than the RDFCS. This material is intended to benefit readers not familiar with the rationale of developing the input parameters for Markov models as used in CARSRA.

A Markov model is used to describe the number of failed and operating modules within each stage. The transition rates from state to state are used by CARSRA in computing state occupancy probabilities. A separate Markov model is used for each stage. State 1 is the no-failure state in each model, and the two states with the highest numbers correspond to stage failure. The Model always starts

in State 1. For example, a dual stage (one of two identical modules required for the stage to function) might have four states, as shown in figure 8.4-1. State 1 represents both modules working. State 2 represents one module failed and one working. States 3 and 4 represent both modules failed. The highest numbered state (state 4 in this case) represents undetected stage failure, while State 3 represents detected failure. Note that State 2 does not distinguish which module has failed.

State transition rates must be supplied to CARSRA by the user. These are generally functions of the module failure rates, and possibly other parameters. Returning to the example of the dual stage used previously, the Markov state diagram would be as in figure 8.4-1. Transition rate $f_{12}$ is the rate at which transitions occur from State 1 to State 2. That is, if the system is in State 1, the probability that it will transition to State 2 during a short increment of time, dt is $f_{12}dt$. The other transition rates are similarly defined.

If there is no monitoring or switching required when the first module fails, and if there is no possibility of the stage failing undetected, the transition from State 1 will always be to State 2, and the transition from State 2 will always be to State 3. Transition rate $f_{12}$ will be simply 2f, and $f_{23}$ will be f, where f is the failure rate of a single module. The other transition rates will be 0. Note that this means that State 4 will never be occupied, consistent with undetected stage failure being impossible.

In many instances encountered in real systems, digital or otherwise, a reconfiguration must occur before the redundancy can be determined. In the example dual case, an output monitor could be used on each module. If the monitor can detect 97 percent of module failures, e.g. no output or unreasonable output, the monitor provides "coverage", c, of 97 percent. The transition rate $f_{12}$ is then 2fc, so that 97 percent of the transitions from State 1 go to State 2.

Of the remaining three percent of the transitions from State 1, some fraction, e.g., 2/3, could go to State 3 and the rest to State 4. This would result in $f_{13}$ being 2f(1-c)(2/3) or 2f(.02), and $f_{14}$ being 2f(1-c)(1/3) or 2f(.01).

Note the distinctions between coverage (which relates to module failure detection) and undetected stage failure. Note also that the function of a particular stage could be such that it cannot fail undetected, even though individual modules within the stage may fail with coverage less than 1. In other cases, stage failure may be detected only by multiple module failures being detected.

The sum of transition rates out of State 1 is 2f. In general, if any state corresponds to N modules working, the sum of transition rates out of that state will be Nf.

Stages can fail for two reasons: spares exhaustion or coverage failure. In contemporary aircraft systems having critical functions to perform, coverage failures are of as much concern as spares exhaustion.

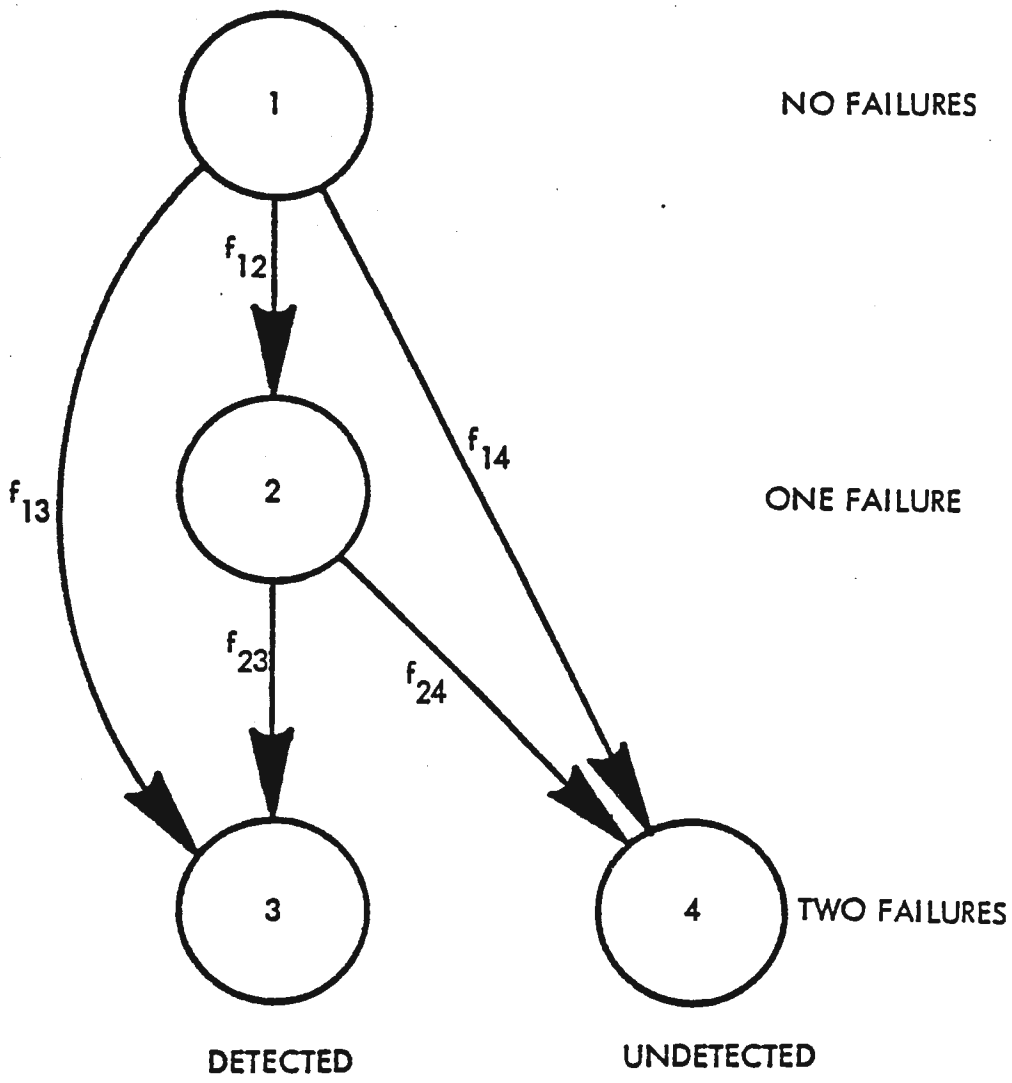



FIGURE 8.4-1. MARKOV MODEL OF DUAL STAGE

In the previous dual stage example with 97 percent coverage of the first module failure, no consideration of the failure rate of the monitor itself was included. The coverage factor of 97 percent means that 97 percent of the module faults are of such a nature that they can be detected by an unfailed monitor. The rest are outside of the monitor's capability. In cases where dedicated hardware monitors are used, it is appropriate to consider their failure rates and failure modes. A two-state monitor is the type most frequently encountered. It provides only a GOOD/BAD signal. Such a monitor has only two failure states: false indication of BAD when the module is good, and false indication of GOOD when the module is bad.

The simplest way of treating such monitors in CARSRA is to combine the monitors with the modules as a single stage. The transition rate from State 1 to State 2 is then

$$2fcr_m + 2f_m a$$

where:

- $f$ = the failure rate of a single module
- $c$ = the coverage
- $r_m$ = the reliability of the monitor over the entire flight time
- $f_m$ = the monitor failure rate
- $a$ = the fraction of monitor failures resulting in a good module being declared bad.

The other transition rates would be similarly defined, recognizing the relation between detection of stage failure and component monitors. Each instance of such a stage must be evaluated individually in determining the applicable rate formulas.

Frequently, certain terms in a rate equation can be ignored because they are numerically negligible. For example, if $f = 120 \times 10^{-6}$ and $f_m = 0.1 \times 10^{-6}$, the term $2f_m a$ can be ignored in the formula

$$f_{12} = 2fcr_m + 2f_m a$$

provided c is not absurdly small. If c is 90 percent, a is 50 percent, and the flight time is ten hours,

$$f_{12} = 2(120 \times 10^{-6})(.90)\ \exp(-.1 \times 10^{-6} \times 10) + 2\ (.1 \times 10^{-6})(.50)$$

$$= 216 \times 10^{-6} + .1 \times 10^{-6}$$

Inclusion of the term yields a rate of $216.1 \times 10^{-6}$; ignoring it yields $216.0 \times 10^{-6}$. The difference is much less than that caused by uncertainty in the module failure rate, $120 \times 10^{-6}$.

### 8.4.3. Dependencies

CARSRA permits the user to describe instances in which failures of a module in one stage will prevent a module in another stage from being used. An example of

this in the RDFCS is the portion of each FCC channel which receives sensor data and makes it available to the other channels. Data Acquisition Card A26 in FCC No. 1 receives data from the No. 1 unit of each triple sensor type and relays it to another card for transmission to the other three channels and for use by its own channel. There are five triple-sensor types involved in the autoland mode: pitch, roll, and yaw-rate gyros, and lateral and NA. (The A26 card also handles data from other sensors, but only these five will be used for discussion here.) If the A26 card fails in FCC No. 1, the data will be lost from pitch gyro No. 1, roll gyro No. 1, yaw-rate gyro No. 1, lateral accelerometer No. 1, and normal accelerometer No. 1, just as if all five of these sensors had failed. The A26 card is called a dependency module and its stage a dependency stage. Each of the affected sensors is called a non-dependency module, and the corresponding stage a non-dependency stage.

Coverage for sensor failures is provided by comparison monitoring and reconfiguration. Each channel independently performs the sensor monitoring functions on the data it will use in control law computations. When an A channel detects a failed sensor, it does not transmit the identity of the individual sensor to the other channels. When a B channel detects a failure, it does transmit a discrete variable, AP.ONEFAIL, to the A channel in the same FCC. The A channel will turn on the NO DUAL annunciation based on its receipt of AP.ONEFAIL from B or its own detection of a sensor failure. The NO DUAL indication is provided to inform the crew that the RDFCS is not fail-operational. The No. 1 FCC drives the No. 1 Warning Annunciation Indicator (WAI) and the No. 2 FCC drives the No. 2 WAI, so that warning will be provided if either channel of either FCC detects the failure.

The sensor monitoring is part of the foreground flight software. Consequently, for a channel to detect a fault, the CAPS processor must function, as must the CAPS bus and portions of the program and data memory. These are the same hardware elements which perform other functions, such as control law computations and mode logic computations. Most faults in these circuits will result in a totally debilitated processor, so that the inability to the monitor sensors is inconsequential. Note also that even if one channel does lose the ability to monitor sensors, any one of the other three channels can force the NO DUAL warning.

Therefore, the only appreciable probability that the loss of fail-operational sensor capability will not be annunciated results from loss of both WAIS. The multiple-function WAI has a unit failure rate prediction of 33 per million hours. The failure rate of any one of the eight warning messages is conservatively taken to be one-fourth the unit rate, or 8.3 per million. The FCC activates the NO DUAL message by providing a ground to the WAIs, so that a broken wire or bad connector contact would prevent annunciation. A rate of 1.3 per million hours is included for such failures. Also, the Discrete Output (A20) and Servo Engage Logic (A28) cards are involved, with failure rates of 2.79 and 2.61 per million hours, respectively. Even though only a portion of the failures of these cards will affect NO DUAL, the full rate is used. Further analysis could reduce this rate substantially. The failure rate for NO DUAL is then

| | |
|---|---|
| WAI | $8.30 \times 10^{-6}$ |
| Wiring | 1.30 |
| A20 Card | 2.79 |
| A28 Card | 2.61 |
| | $15.00 \times 10^{-6}$ |

The probability of failure in a 4-hour time period is then $60 \times 10^{-6}$. The probability of both NO DUAL warnings being lost is the square of this number, $3.6 \times 10^{-9}$. The test button on the WAI results in the FCC circuitry and the wiring being tested as well as the WAI itself. Thus latent failures are not a problem, provided the indicators are tested prior to autoland.

The factor $3.6 \times 10^{-9}$ is used as the probability that the first failure of a sensor type will not be covered. This does not constitute stage failure, either detected or undetected. Undetected stage failure is assumed to occur on second failure, provided the first failure was undetected. This is somewhat a misuse of the term "undetected"; the stage failure itself is not necessarily undetected, but the increased likelihood of its occurrence, following first failure, is not annunciated.

This treatment of sensor failures allows the availability feature of CARSRA to be used in computing the probability of loss of one sensor prior to 150 ft, failure of the NO DUAL annunciation, and another failure below 150 ft.

8.4.4. Availability

CARSRA permits system reliability to be computed for a mission phase which follows a period of operation with less stringent failure criteria. An obvious example of this is the RDFCS, which is fail-passive in cruise, but must be fail-operational in autoland below 150 ft. The availability feature allows the user to specify which modules may be failed at the beginning of autoland without forcing diversion to an alternate landing site. Each such availability configuration must provide adequate reliability for the landing, although not as much as if everything is working. The RDFCS requires all of the modules used in autoland to be operational, so that the availability feature might seem not needed in this assessment. It is needed, though, to compensate for a capability which CARSRA lacks.

The reliability of the RDFCS for automatic landing is predicated on the system being fail-operational as the alert height is passed. Therefore, the probability of the system having a latent failure at 150 ft and a second failure below that point must be quite small.

By setting up the CARSRA input to allow one sensor of each type to fail during cruise, with the transition rate from State 2 to the undetected failure state including the coverage factor of $3.6 \times 10^{-9}$, the undetected system failure probability computed by CARSRA will give the probability of an undetected latent failure at 150 ft and a second failure before touch-down. (See figure 8.4-2.)

What CARSRA will actually compute is:

$P_1 \times P_2 + P_3 \times P_4$

where,

$P_1$: probability of 0 failures at 4 hours

$P_2$: probability of undetected failure and detected failure between 4 and 4.02 hours

$P_3$: probability of 1 undetected failure at 4 hours

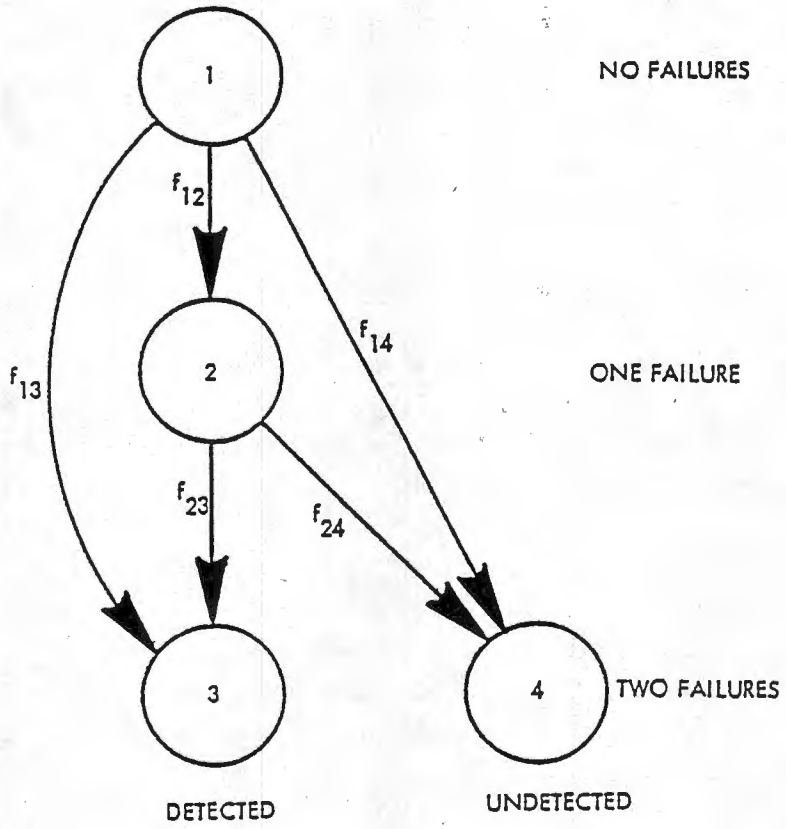
$P_4$: probability of second failure between 4 and 4.02 hours

Since the probability of both an undetected and a detected failure between 4 and 4.02 hours is very small, the first term is negligible and the output will be equal to the second term, which is the probability desired. This approach is used for the undetected (unannunciated) failures throughout the system. The definition of stages and the transition rates are shown in figure 8.4-3.

The CARSRA program is sensitive to the precision of the host computer. During this analysis, some negative probabilities for unannunciated failures were computed. It is suspected that this may have been caused by the 36-bit word length of the Univac 1100-series computer. The transition rates to the unannunciated failure states are quite small, in some cases $1.0 \times 10^{-13}$, and addition/subtraction of numbers of this magnitude with numbers close to 1.0 could produce some numerical accuracy problems on a 36-bit machine. If the program is run on a computer with a 64-bit word length, the problem is thought to be unlikely.

Because of the numerical problem encountered with the CARSRA output, the system failure probabilities reported here were actually manually calculated. This was done by manually computing the stage occupancy probabili-ties, and then combining these probabilities to account for dependencies between stages, using the same logic that the CARSRA program uses.

The probability of an undetected failure prior to the crucial phase, followed by a second failure in the crucial phase, is $3.36 \times 10^{-14}$, compared to $2.46 \times 10^{-14}$ from the fault trees. The probability of multiple failures in the crucial phase, if everything is working just prior to the phase, is $0.658 \times 10^{-9}$, compared with $0.638 \times 10^{-9}$ from the fault trees. These results are shown in table 4.3-1.




| | DUAL SENSOR | TRIPLE SENSOR |
|---|---|---|
| $f_{12}$ | 2f | 3f |
| $f_{13}$ | 0 | 0 |
| $f_{14}$ | 0 | 0 |
| $f_{23}$ | f | 2f |
| $f_{24}$ | fa | 2fa |

f = MODULE FAILURE RATE

a = ANNUNCIATION FACTOR $3.6 \times 10^{-9}$

FIGURE 8.4-2. MARKOV MODEL CODING FOR SENSOR STAGES

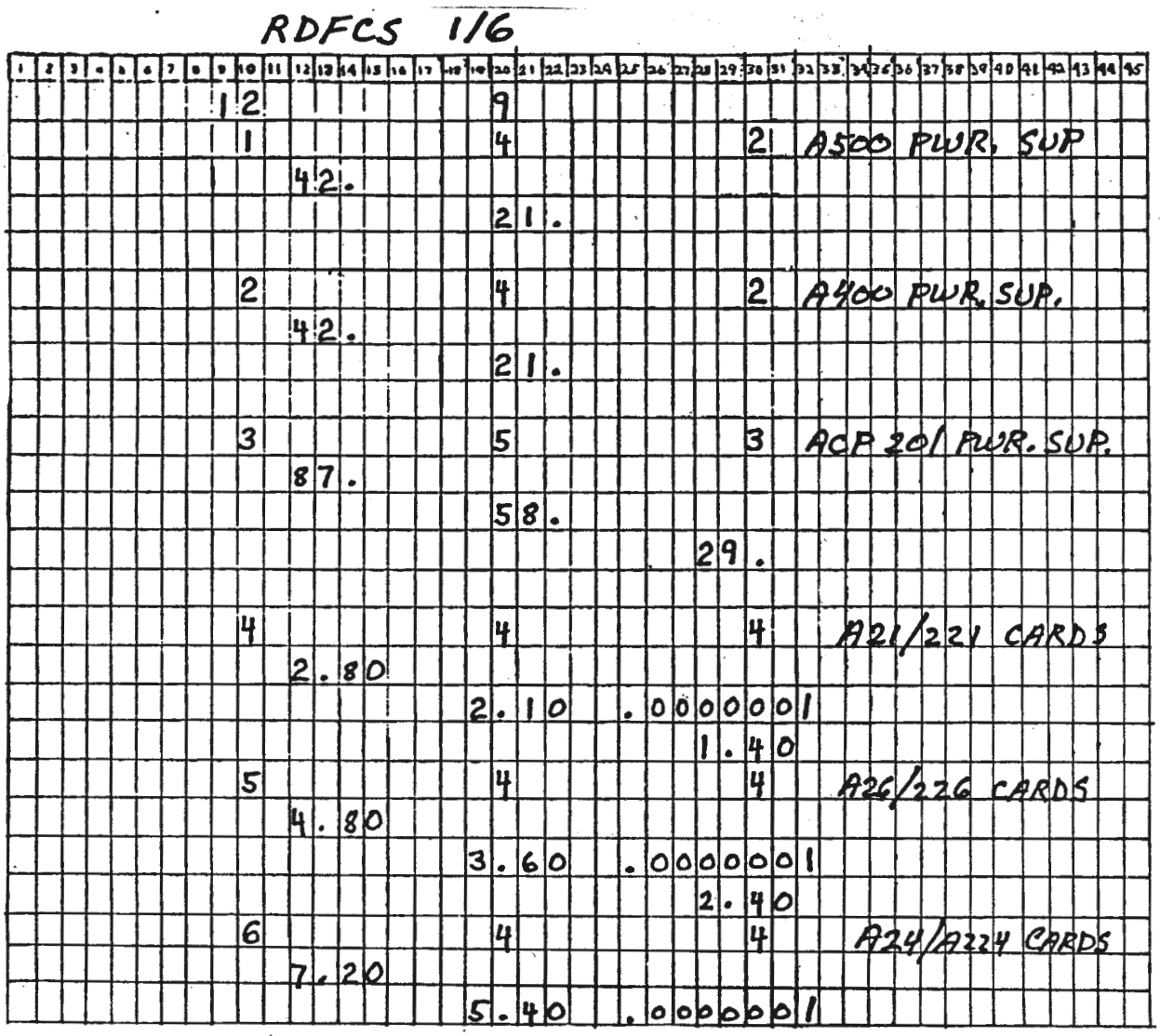



FIGURE 8.4-3. CARSRA INPUT (1 of 6)

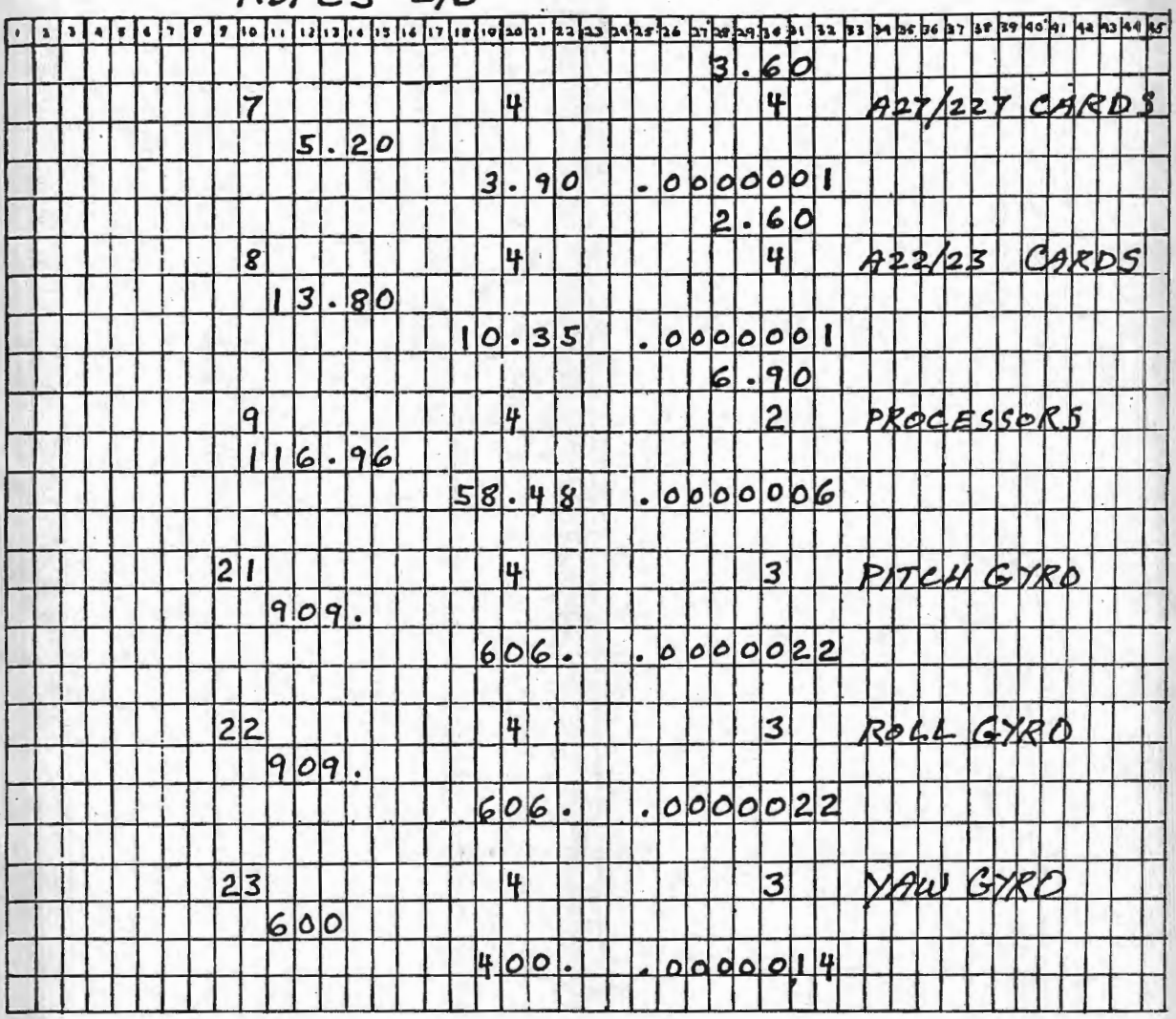



FIGURE 8.4-3. CARSRA INPUT (2 of 6)

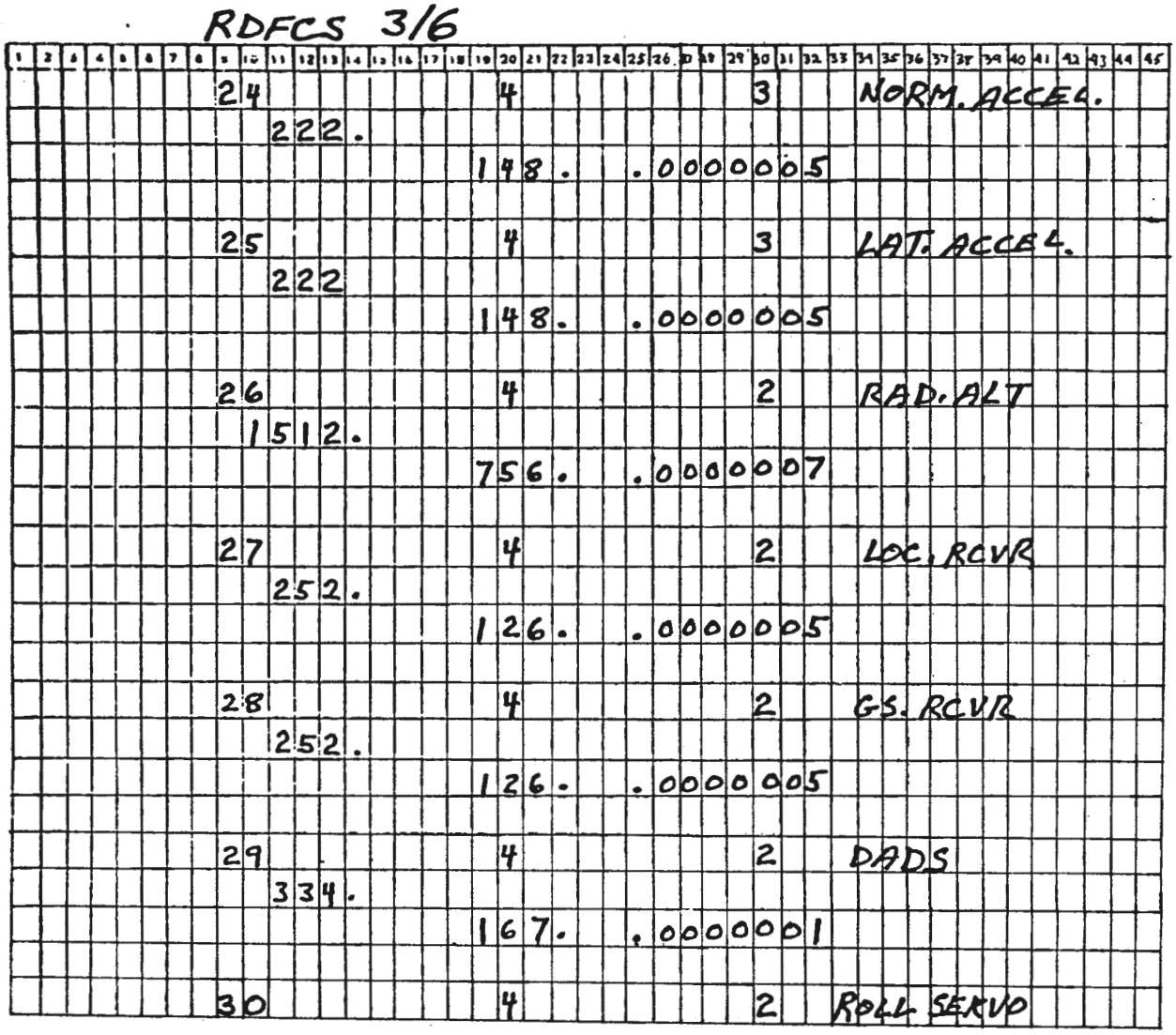



FIGURE 8.4-3. CARSRA INPUT (3 of 6)

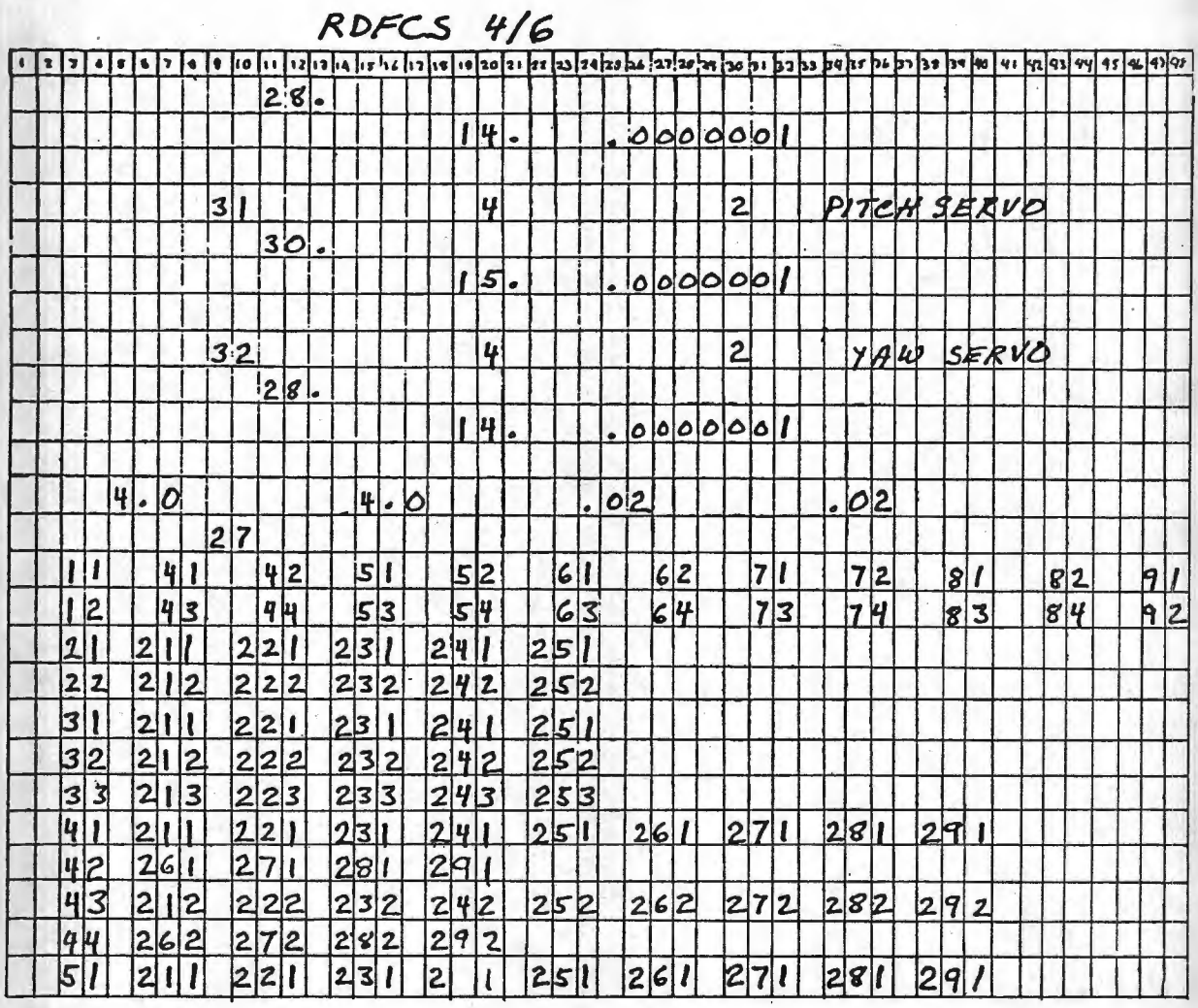



FIGURE 8.4-3. CARSRA INPUT (4 of 6)

RDFCS 5/6

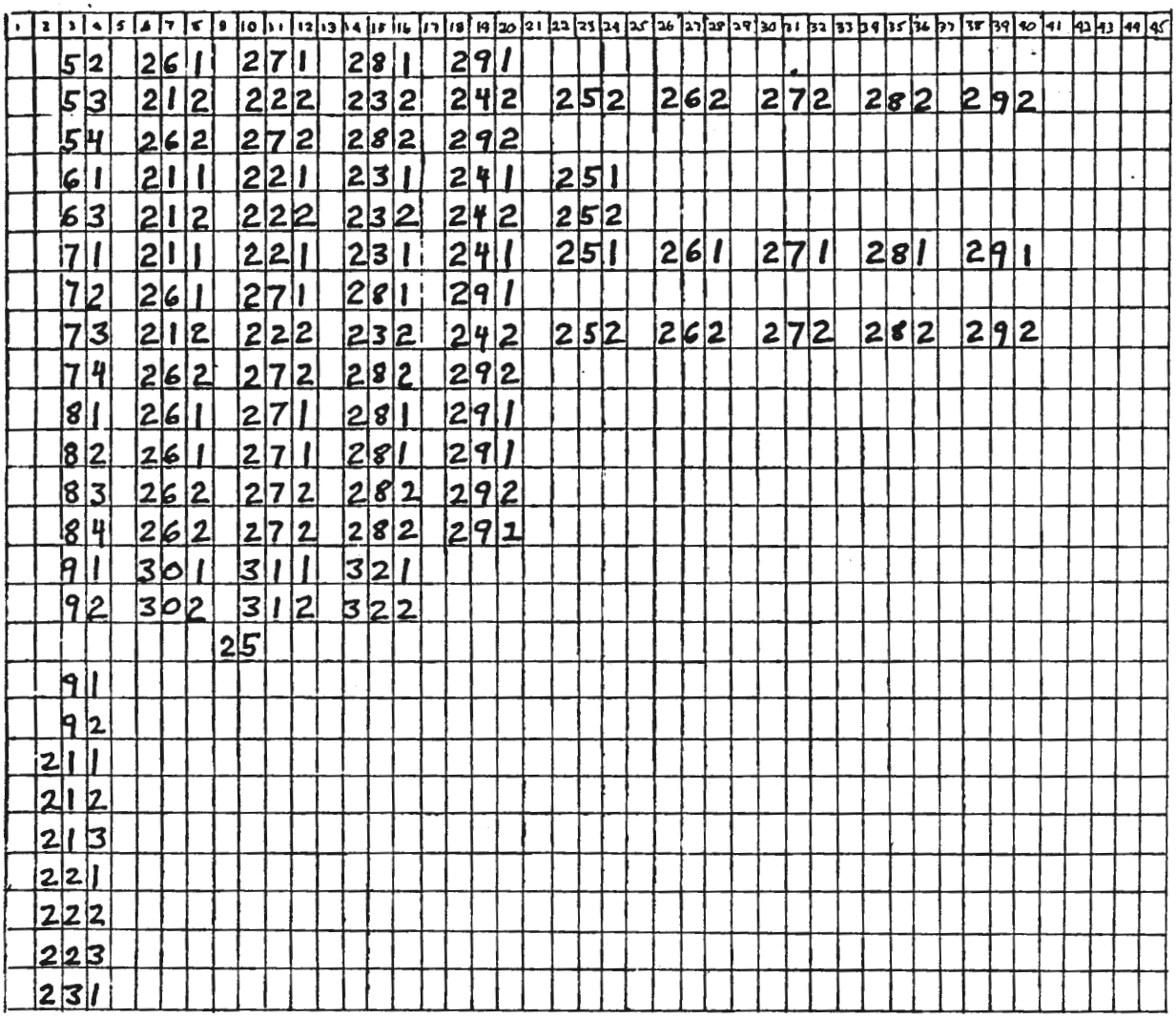



FIGURE 8.4-3. CARSRA INPUT (5 of 6)

RDFCS 6/6

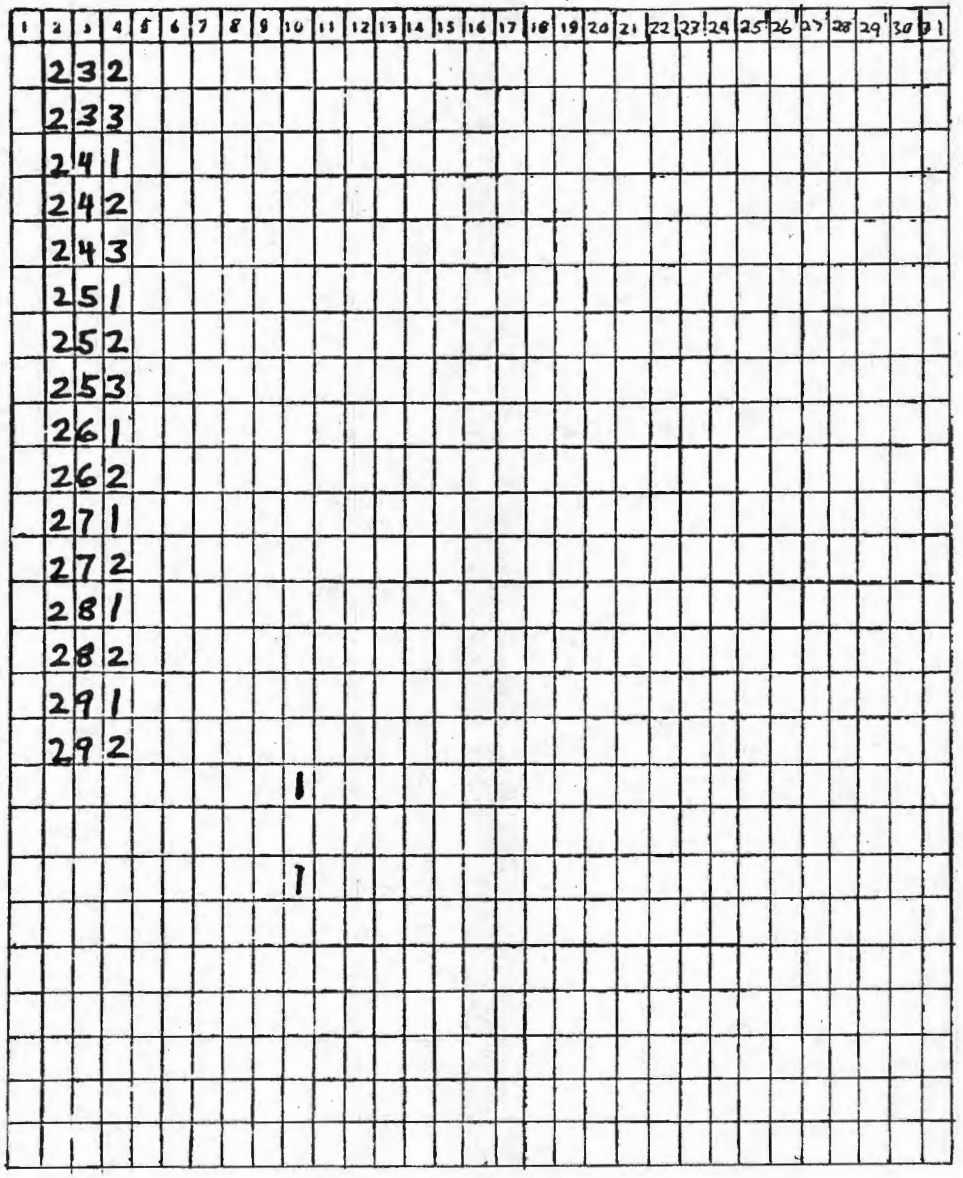



FIGURE 8.4-3. CARSRA INPUT (6 of 6)

# 9. SUMMARY

## 9.1. Background

The purpose of this tutorial is to provide certification engineers with a background in IAA. The vehicle for accomplishing this is to provide a walk-through of a system failure analysis using a critical system, the RDFCS. The complexity of airplane systems (RDFCS being a prime example) has necessitated the evolution of safety analysis techniques that permit the diagnosis of failure effects of multiple faults, both intra- and intersystem. A diagnosis of this type is important in demonstrating compliance with Part 25 of the FARs for "flight-essential" and "flight-critical" avionic systems. AC 25.1309-1 provides guidance material for complying with the above regulation. The techniques suggested in the AC were the basis for the RDFCS analysis. The analytical techniques were selected to mutually reinforce each other on the following bases:

- Fault tree analysis identifies the conditions and functional failures contributing to a defined failure condition. It is also used to compute a quantitative assessment of reliability (system failure probability).
- Analytical reliability prediction analysis confirms the system failure probability computed with fault tree analysis.
- FMEA was used to augment fault tree analysis in diagnosis below the circuit card level.
- Fault insertion was used to confirm the fault detection and tolerance capabilities of the RDFCS. The existence of the RDFCS Simulator made fault insertion possible.

The integration of the above techniques was exemplary. The selection of safety analysis techniques for integration is the responsibility of the test/certification engineer and must be customized to the system. The use of any IAA model must supplement, but not replace, the judgment of the FAA certification personnel.

## 9.2. Concluding Remarks

The concluding remarks for this tutorial relate to the benefits and limitations of the IAA model used in the RDFCS analysis. During the course of this analysis, certain enhancements were visualized that would have improved the study. These enhancements will be discussed in section 9.3. They are included in this tutorial, because of their potential application to other integrated analyses.

IAA is workable for a system, such as the RDFCS, which employs monitoring totally separate from the hardware/software being monitored. In the RDFCS, this monitoring includes the servo coil current comparators and the modulator piston followup monitoring. It also includes the warning annunciations which one FCC can generate following a failure in the other FCC. A single-string, self-monitored system might be much less amenable to this approach, depending on the monitoring approach used.

Fault tree analysis is a feasible analytical method for identifying system level faults. One benefit is that specific software failures are identified as the analysis progresses. These can be, and should be, used as a check on the validation test case selection to assure that the software function is rigorously tested. Fault trees can be extended to the circuit card level in a well organized computer such as that used in the RDFCS. In general, the analysis is facilitated by a design with clearly partitioned and identifiable functions, and an interface structure which is consistent for all card inputs and outputs.

FMEA is more easily accomplished than fault trees within the processor itself. This is because the processor is involved in a diverse set of functions defined by the flight software. Most individual pin-level faults have many effects. Usually, each fault can be traced to an effect which totally debilitates the processor. Other effects, which cause massive processor failure or erroneous results only under certain conditions, do not have to be analyzed in detail when their effects do not propagate across channels. In contrast, a fault tree analysis based on loss of required system functions would result in identification of the same hardware faults time after time. When a system simulator exists, fault insertion is an important adjunct to FMEA.

Fault tree analysis and reliability analysis tools, such as, CARSRA, provide comparable results for relatively straightforward redundancy conditions, e.g., the probability of multiple failures during the crucial phase when all components are working at the beginning of the phase. For more complicated situations, the two methods do not agree as closely. This is a result of different simplifications and assumptions being made to structure the problem to the two methods. For example, the third sensor of a triple sensor set (figure 2.1-1) has redundant input paths to the computers (the data input sections of the two computer B channels), but the other sensors have only a single data path (the A channel input sections). This is treated correctly in the fault trees, but the redundancy cannot be accounted for in CARSRA. The conservative assumption is therefore made that loss of either B channel sensor input capability will cause loss of the third sensor in all triple sensor sets. In validation work, any assumptions required can be made conservatively so that the computed failure probability is actually an upper bound on the true probability.

## 9.3. Analysis Enhancements

The RDFCS simulator was invaluable as a tool in the failure analysis function. Given sufficient access to the simulator, the FMEA and fault insertion test sessions should be scheduled on an iterative basis. After performing certain FMEAs, a fault insertion session should be used to confirm the analysis to that

point. The results should then be incorporated in the FMEA, and the entire FMEA reviewed in light of those results. This review may lead to identification of additional fault cases which should be simulated to resolve uncertainty which may have arisen. This iterative approach was not used in the subject study because of limitations on the availability of the simulator.

The RDFCS simulator has substantial capability for research investigations of DFCS validation issues. This capability would be significantly improved by an automated fault insertion and data recording capability. Such a capability should be preprogrammable with a list of faults to be inserted. It should include a means of recording the impact of each fault (e.g., changes in the values of discrete variables) for many more variables than the four accessible through the CTAs. It should allow variables in channels other than the faulted one to be accessed and recorded.

Care should be taken in selecting a computer tool for analytic reliability prediction. The nature of the analytical program requires a high degree of numerical precision when solving certain problems. In the subject study (see section 1.2) the system failure probabilities used to confirm the fault tree analysis computations were manually calculated. This was done by manually computing the stage occupancy probabilities and then combining these probabilities to account for dependencies between stages. The logic used was identical to that used by the CARSRA program. The problem was thought to be due to the small failure rates and the accompanying precision difficulties encountered on a computer with a 36-bit word length. If so, the use of CARSRA on a computer with a 64-bit word length would have remedied the problem.

## APPENDIX A - SUMMARY DESCRIPTION FOR REDUNDANT DIGITAL FLIGHT CONTROL SYSTEM ARCHITECTURE

### Redundant Digital Flight Control System

In most operational modes, the RDFCS is fail-passive, with a dual channel configuration. For automatic landings under Category IIIa conditions (see AC 120-28C), the system can be brought into a dual-dual fail-operational, fail-passive configuration. The classification dual-dual relates primarily to the four computer channels in the system. Each of the two FCCs has two channels which run frame-synchronously, with each channel driving one coil of a dual-coil servo in each axis. Any indication of disagreement between the two channels in an FCC causes the servo connected to that FCC to be disengaged by removing hydraulic pressure. Figure 2.1-1 of this report summarizes the dual-dual configuration.

### Monitoring Configuration and Implementations

Extensive monitoring is employed in the RDFCS for fault detection. Coil current comparators for each servo provide coverage of faults resulting in erroneous commands to the servo coils. They also provide coverage for broken wire faults between the FCC and the servo or failures of the coils themselves. These monitors are made more effective by the insertion of opposing 5 ma bias currents. The bias currents permit circuit integrity to be monitored even when the FCC is not commanding the servo to a new position, such as when the aircraft is flying through very calm air at a stable altitude. It may be noted that this type of monitoring is equally applicable to analog and digital systems.

Response of the autopilot servos to commands from the servo amplifiers is monitored by modulator position signals fed back to the FCC. The feedback signals are averaged and passed through a high-pass filter to get a modulator rate that is compared with cold current. This comparison is used to detect jamming of the modulator piston, runaway conditions, or loss of hydraulic power. This type of monitoring can also be applied to either analog or digital systems.

In the pitch-axis servos, modulator piston position monitoring is implemented in hardware. In the other two axes, it is implemented in software. Together, the coil current monitoring and modulator piston monitoring detect any servo fault which prevents the servo from responding to commands. They also detect any fault in a computer channel which prevents that channel from generating a reasonable command for the servos in each of the three axes. All monitors and feedback sensors are dual to increase reliability.

Each computer channel has an iteration monitor implemented in hardware. This monitor observes the state of a discrete software variable which is changed at the end of each iteration of the foreground software. Since this software executes at a 10 Hertz (Hz) rate, the result is a 10 Hz square wave. Should the processor short-loop or hang up, the 10 Hz wave will not be presented. The

iteration monitor will withdraw its input to the engage logic, and the FCC will disengage.

Sensor monitoring is primarily accomplished by comparison and by validity discretes generated by the sensors. There is no one location that sensor monitoring takes place, since all four computer channels incorporate the monitoring function. This process ensures that the circuitry involved in getting the sensor signals to each channel is included in the monitoring.

The accelerometers are tested each time the system is powered up with the airplane on the ground. The ILS receivers are checked using a square wave test. This test checks for failure of the localizer and glideslope beam deviation inputs. During landing, the outputs of both receivers are compared, with reliance on the self-monitoring to identify which receiver is bad if the signals disagree. The comparison monitoring is used to check wire integrity between the receiver and the computer channels. The other dual sensors are comparison monitored in the same way.

Even though each channel monitors sensors individually, any channel can initiate the NO DUAL annunciation, which is the primary indication that the system is not fail-operational. If any channel detects a second failure of a sensor type, it will cause its FCC to disengage, but the other FCC will remain engaged.

Although NO DUAL is the primary warning of loss of one sensor, NO ALIGN will be annunciated if the course signals from the two compass systems do not agree.

Other monitoring within the FCC involves comparison of active operating modes. If the two channels within an FCC disagree on which modes are engaged, and the disagreement lasts for more than 0.1 sec, the FCC will disengage. If the two FCCs disagree, SPLIT will be displayed on the WAIs. This monitoring, together with the sensor data transfers, will detect most faults of the cross-channel data transfer circuitry.

## APPENDIX B - DESCRIPTION OF RDFCS SIMULATOR

The RDFCS simulator is comprised primarily of a PDP 11/60 computer and the RDFCS pallet. The RDFCS pallet includes the FCCs, core memory, MDICU, SSP, DSP, CTA, and Computer Breakout Panels. The functions of these items are described in the remainder of this section. Figures 2.1-2 through 2.1-5 are pictures of these items.

### PDP 11/60 Computer/Airplane Model

The PDP 11/60 computer hosts a discrete-state model of the airplane in which the RDFCS is installed. This airplane is a representative wide-body transport, and the model coefficients are changed according to the flight case being simulated. Each flight case, then, is a point simulation of the airplane in a particular configuration and operating in a specific portion of the flight envelope. The airplane model executes at a 50 Hz rate.

During the subject study (see section 1.2) the flight cases described and discussed in Lockheed-Georgia LG81E0126 (1981) were used in the simulation. As part of this study, a go-around case was added to the library of cases available. The go-around case is characterized as follows:

| | |
|---|---|
| Airplane Weight | 314,500 lb |
| Altitude | 35 ft |
| Angle of Attack | 10.91 ° |
| Indicated Air Speed | 168 kts |
| Flap Deployment | 22 ° |

Transition capability was added to go from approach conditions to landing conditions, and from landing to the new go-around case. The transitions involve changing the model coefficients and establishing new trim values. The transition capability has been installed and checked out successfully.

### Modular Digital Interface Control Unit

The MDICU receives the output of the airplane discrete-state model through a communication link with the PDP 11/60 computer. The MDICU converts the various pieces of information into the form needed by the FCCs. For example, roll angle and pitch angle are converted to three-wire AC signals, properly scaled, while localizer deviation is coded in Aeronautical Radio Incorporated (ARINC) serial digital format. The MDICU is described more fully in the NASA RDFCS System Interface Document.

The MDICU incorporates provisions for the signal for the No. 1 sensor of each type to be ramped up or down. This facility is accessed by means of the HP 2645A terminal physically located in the pallet.

Computer Breakout Panels

Each sensor signal going from the MDICU to the FCCs can be interrupted at the Computer Breakout Panels by removing the appropriate jumper plug. Every FCC back connector pin is routed through one of these plugs. The lower portion of figure 2.1-3 shows the rows of plugs for connector P1 and the "A" half of connector P2. Each FCC has its own breakout panel.

CAPS Test Adapters

Figure 2.1-3 also shows the CTA for one of the FCCs. The upper half of the CTA includes, on the right-hand side, four address and four data windows. An address can be loaded in each address window. The corresponding data window can be used to display the data on the FCC A-side processor bus data lines every time the address appears on the address lines. The CTA also has other capabilities, such as providing a history of the last 16 bus transfers and changing the contents of a specific memory location within the FCC, but during the subject study only the address monitoring was used. Discrete variables representing sensor voter status were monitored visually via the data windows. Continuous variables, such as inputs to the servo amplifiers, were monitored by using the analog output posts below the appropriate data window to drive a strip-chart recorder.

The lower half of the CTA performs the same functions as the upper half, but for the B side of the FCC.

Servo Simulator Panel

The servo amplifier outputs from the FCCs are routed to the SSP. The SSP simulates the dynamics of the autopilot and power servos, and generates the required feedback signals such as modulator piston position. The SSP has circuits which can simulate a hardover or slowover command to a servo coil. It can also simulate a hardover or slowover of a modulator piston, including the modulator piston position feedback signal and the command to the power servo. All of these apply to the No. 1 servo of each type.

Discrete Switch Panel

The DSP is located just below the SSP. This panel provides a centralized location for switches such as hydraulic pressure switches and autopilot disconnect switches. The panel also includes switches that can be used to insert sensor validity faults. These faults can also be inserted by pulling the appropriate jumper plug on the FCC Breakout Panel.

Core Memory

The pallet also contains core memory for the FCCs. This is used for both data and program memory to provide flexibility and convenience in using the pallet to simulate other airplanes or DFCS architectures. As used in an airplane, the FCCs have the flight software stored in PROM and use Random Access Memory (RAM) chips for data memory.

## Glare-Shield Panel

The pallet also has a glare-shield panel, which is the control panel for the system as installed in an airplane. It includes the engage (bat handle) switches, mode select switches, altitude select knob, and other controls. The pallet also has a single Automatic Direction Indicator (ADI), Horizontal Situation Indicator (HSI), radio altitude display, Mode Indicator, and WAI.

| Circuit | Function | Pin | Fault | Effect |
|---|---|---|---|---|
| Instruction Mapper Prom CU1 | Produce direct input bits A0-A3 for control store memory micro-program start address | A0-A9 | Open | Address bit low, wrong address read. Wrong output passed to control store proms as starting address bits A0-A3. Massive processor failure. |
| | | | Low | Address bit sticks, wrong address read. Wrong output passed to control store proms as starting address. Massive processor failure. |
| | | | High | Same as above |
| | | CS1, CS2 | Open | Output pins remain in high-impedence state. Input pins to microprogram sequencer CU16 low. Wrong starting address bits A0-A3 to control store proms. Massive processor failure. |
| | | | Gnd | Don't care. |
| | | | High | Same as open. |
| | | O1-O4 | Open | Prom output bit not fed to microprogram sequencer input bit. Input bit low, resulting in wrong microprogram starting address. Massive processor failure. |
| | | | Low | Corresponding bit (A0-A3) of microprogram start or jump address is always low. Massive processor failure. |
| | | | High | Corresponding bit (A0-A3) of microprogram start or jump address is always high. Massive processor failure. |
| Instruction Mapper Prom CU7 | Produce direct input bits A4-A7 for control atore memory microprogram start address. | | | The fault of any pin of CU7 has the same effect as the same fault occurring in CU1, except that the affected address bits are A4-A7. |

| Circuit | Function | Pin | Fault | Effect |
|---|---|---|---|---|
| Instruction Mapper Prom CU13 | Produce direct input bits A8-A9 for control store memory micro-program start address; produce push/pop signals to stack vector register DU4. | A0-A9 | Open | Address bit low, wrong address read. Wrong output passed to control store proms as starting address bits A8-A9. Wrong output may also include wrong push or pop signal to stack vector. |
| | | | Low | Address bit stuck low, wrong address read. Bits A8-A9 of microprogram start address wrong. Push or pop signal to stack vector register DU4 may be wrong. Massive processor failure. |
| | | | High | Address bit stuck high, wrong address read. Bits A8-A9 of microprograms start address wrong. Push or pop signal to stack vector register DU4 may be wrong. Massive processor failure. |
| | | CS1, SS2 | Open | Control register cannot pull down enable, so that output pins are at high impedance. Start address bits A8-A9 always low. Massive processor failure. |
| | | | Low | Chip CU13 can pull down data input to microprogram sequencer when control register is trying to set it high as part of a jump address. |
| | | | High | Same as open. |
| | | O1-O2 | Open | Start address bit A8, A9 to control store always low. Massive processor failure. |
| | | | Low | Same as open. |
| | | | High | Start address bit A8, A9 high; address bad when bit should be low. Massive processor failure. |

| Circuit | Function | Pin | Fault | Effect |
|---|---|---|---|---|
| CU13 con't. | | 03-04 | | |
| | | | Low | POP commanded (pin 03 faulted) or PUSH commanded (pin 04 faulted) on each clock pulse, so that both commands will go to stack vector register when only non-faulted should be. Stack vector register will do nothing. Massive processor failure. |
| | | | High, Open | Fault in pin 03 causes stack vector register to broadside load instead of left shift when mapper prom tries to pop stack. Fault in pin 04 causes broadside load instead of right shift when mapper prom tries to push stack. Stack pointer not pointing to top of stack. Massive processor failure. |
| Microprogram Sequencer CU14 | Generate sequence of microcode Code addresses for control store proms using starting address from mapper or control register. U14 generates microcode address bits A8,A9 | D0, D1 | Open | Address bit A8, A9 to control store proms always low when starting microcode sequence or on microcode jump. Massive processor failure. |
| | | | Gnd | Same as above |
| | | | High | Same as above, except that affected bit is always high. |
| | | CN | Open | Carry-in from microprogram sequencer CU15 is always low. Wrong address will be sent to control store when address increment causes overflow in CU15. Effect depends on allocation of control store store addresses to microcode sequences. |
| | | | Gnd | Same as above |

| Circuit | Function | Pin | Fault | Effect |
|---|---|---|---|---|
| CU14 con't. | | | High | In incrementing address register, bit A8 will be toggled on each clock pulse. Wrong microcode address will be generated during most microcode sequences. Massive processor failure. |
| | | OE | Open | Address bits A8, A9 always low. |
| | | | Gnd | No effect during operation. Maintenance troubleshooting affected. |
| | | RE | Open | Initial microsequence address from mapper prom cannot be loaded. Massive processor failure. |
| | | | Gnd | Same as above. |
| | | | High | No effect. |
| | | SO | Open | Sequencer will not jump to proper address when SO should be high. Massive processor failure. |
| | | | Gnd | Same as above. |
| | | | High | Sequencer will execute erroneous jump when SO should be low. Massive processor failure. |
| | | SI | | Same effect as pin SO faulted. |
| | | FE | Open | Microprogram counter will always be pushed onto stack or stack will be popped, depending on PUP. Massive processor failure. |
| | | | Gnd | Same as above. |
| | | | High | Microprogram counter cannot be pushed on stack and stack cannot be popped. Massive processor failure. |

| Circuit | Function | Pin | Fault | Effect |
|---|---|---|---|---|
| Microprogram Sequencer CU14 con't. | | PUP | Open | Stack will be popped when microprogram counter should be pushed on stack. Massive processor failure. |
| | | | Gnd | Same as above. |
| | | | High | Microprogram counter will be pushed onto stack when stack should be popped. Massive processor failure. |
| | | CP | Any | Chip disabled. CP is clock pulse input, and all state changes occur on low-to-high transition of CP. Massive processor failure. |
| | | ZERO | Open | Address bits A8, A9 to control store proms always low. Massive processor failure. |
| | | | Gnd | Same as above. |
| | | | High | Address bits A8, A9 not forced low when commanded by control register CU21. Effect depends on implementation of microcode. |
| | | Y0, Y1 | Open | Corresponding address bit to control store is always low. Massive processor failure. |
| | | | Gnd | Same as above. |
| | | | High | Corresponding address bit to control store is always high. Massive processor failure. |
| | | Y2 CN+4, D2,D3 | Any | No effect. Pins not connected. |
| | | VCC | Open | Chip dead. Massive processor failure. |
| | | Gnd | Open | Chip dead. Massive processor failure. |

| Circuit | Function | Pin | Fault | Effect |
|---|---|---|---|---|
| Microprogram Sequencer CU15 | Generate address bits A4-A7 to control store proms. Effects of most pin faults are the same as for CU14, except the affected address bits are A4-A7. Only pins with different fault are discussed. | D2,D3 | Open | Corresponding address bit A6 or A7 is always when address is from mapper prom or micro-coded jump address. Massive processor failure. |
| | | | Gnd | Same as above. |
| | | | High | Corresponding address bit A6 or A7 is always high when address is from mapper prom or microcoded jump address. Massive processor failure. |
| | | CN | Open | Carry-in from microprogram sequencer CU14 is always low. Wrong address will be sent to control store when address increment causes overflow in CU14. Massive processor failure. |
| | | | Gnd | Same as above. |
| | | | High | Carry-in from microprogram sequencer CU14 is always high. Microcode address incremented on bit A4 each clock cycle. Massive processor failure. |
| | | CN+4 | Open | Same as CN open on CU14. |
| | | | Gnd | Same as above. |
| | | | High | Same as CN high on CU14. |
| | | Y2,Y3 | Open | Corresponding bit A6 or A7 always low in address to control store. Massive processor failure. |
| | | | Gnd | Same as above. |
| | | | High | Corresponding bit A6 or A7 always high in address to control store. Massive processor failure. |

| Circuit | Function | Pin | Fault | Effect |
|---|---|---|---|---|
| Microprogram Sequencer CU16 | Generate address bits A0-A3 to control store proms. Effects of most pin faults are the same as for CU15, except that affected address bits are A0-A3. Only pins with different effects are discussed. | CN | Open | Microprogram address is not incremented during execution of microcode sequence. Massive processor failure. |
| | | | Gnd | Same as above. |
| Micro-processor DU17 | Processes the four low-order bits of the 16-bit CAPS word in response to instructions from control registers. | A0-A3 | Open | Wrong A pointer address when failed bit should be high. Massive processor failure. |
| | | | Gnd. | Same as above. |
| | | | High | Wrong A pointer address when failed bit should be low. Massive processor failure. |
| | | B0-B3 | Open | Wrong B pointer address when failed bit should be high. Massive processor failure. |
| | | | Gnd | Same as above. |
| | | | High | Wrong B pointer address when failed bit should be low. Massive processor failure. |
| | | I0-I2 | Open | Wrong data source selected when failed bit should be high. Massive processor failure. |
| | | | Gnd | Same as above. |
| | | | High | Wrong data source selected when failed bit should be low. Massive processor failure. |

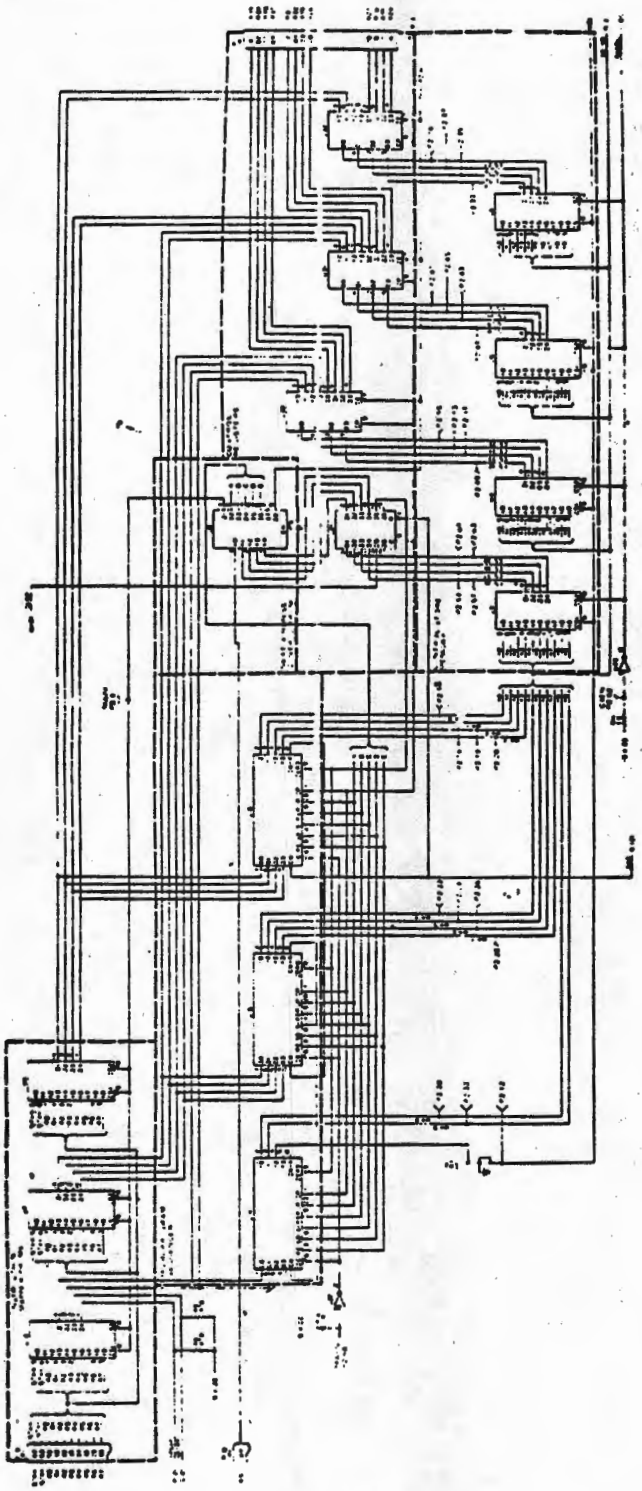
| Circuit | Function | Pin | Fault | Effect |
|---|---|---|---|---|
| Microprocessor DU17 | | I3-I5 | Open | Wrong operation performed when failed bit should be high. Massive processor failure. |
| | | | Gnd | Same as above. |
| | | | High | Wrong operation performed when failed bit should be low. Massive processor failure. |
| | | I6-I8 | Open | Wrong destination code when failed bit should be high. In most cases, the immediate effect will be internal to the chip involving load or shift of data in registers. Massive processor failure. |
| | | | Gnd. | Same as above. |
| | | | High | Wrong destination code when failed bit should be low. Massive processor failure. |
| | | CP | Any | Chip dead. Massive processor failure. |
| | | DO-D3 | Open | Input to processor is wrong when failed bit should be high. Major effect caused by incorrect bit in packed Boolean data. Massive processor failure. |
| | | | Gnd. | Same as above. |
| | | C | Open | Carry-in always low. Program counter not incremented on instruction fetch. Massive processor failure. |
| | | | Gnd. | Same as above. |
| | | | High | Carry-in always high. Foreground loop of flight software cannot execute paths 2 and 4; iteration monitor test bit not toggled; iteration monitor trips. FCC disconnects. |

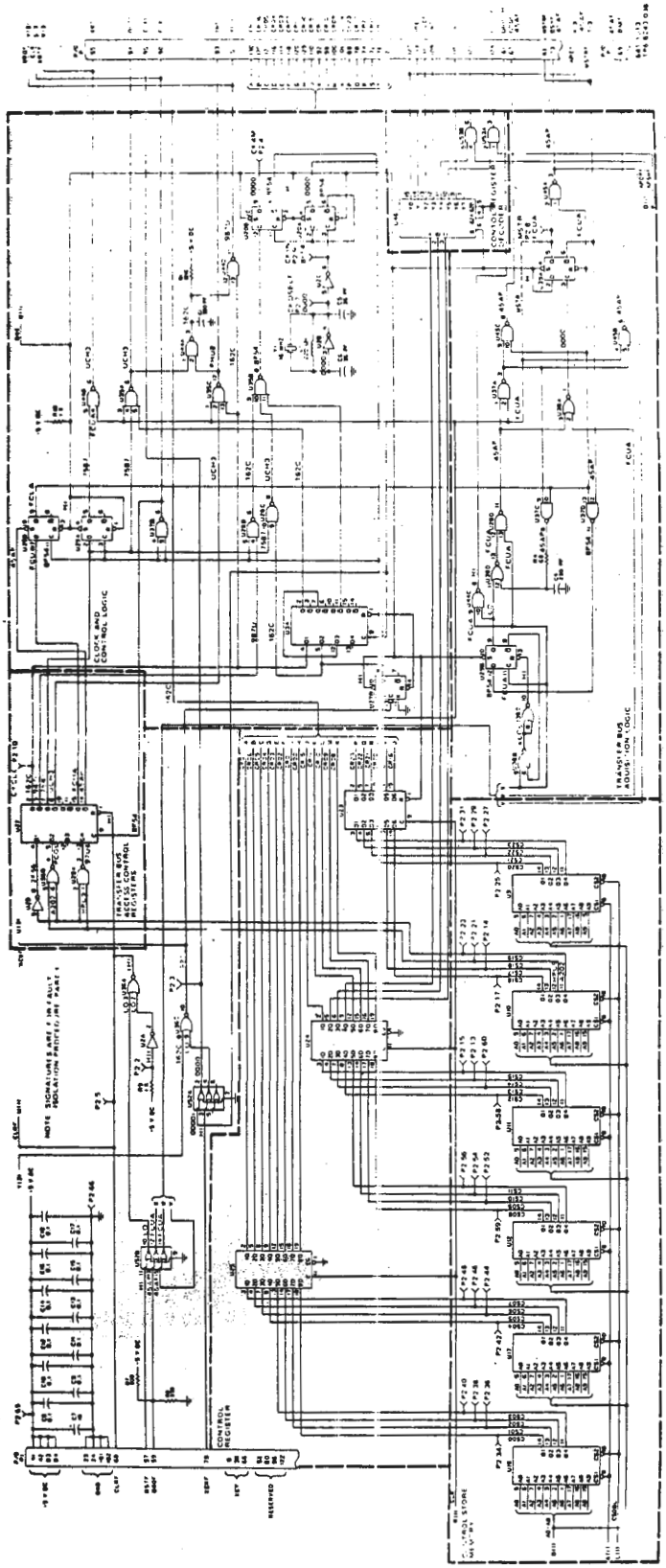
| Circuit | Function | Pin | Fault | Effect |
|---|---|---|---|---|
| Micropro-cessor DU17 con't. | | Y0-Y3 | Open | Wrong address gated on CAPS address lines. Massive processor failure. |
| | | P | Open | Carry propagate always sent to carry look-ahead logic. Massive processor failure. |
| | | | Gnd | Same as above. |
| | | | High | Carry propagate never sent to carry look-ahead logic. Double-precision integrators drift. |
| | | G | Open | Carry generate always sent to carry look-ahead logic. Massive processor failure. |
| | | | Gnd | Same as above. |
| | | | High | Carry generate signal never sent to carry look-ahead logic. |
| | | F=0 | Open | DU17 cannot pull down F=0 line to status register, yielding false results for some logic tests. Massive processor failure. |
| | | | Gnd | DU17 always pulls down F=0 line to status register yielding false results for some logic tests. Massive processor failure. |
| | | | High | Same as open. |
| | | Vcc | Open | Chip dead. Massive processor failure. |
| | | OE | Open | Chip dead. Massive processor failure. |
| | | Gnd | Open | Chip dead. Massive processor failure. |
| | | R3 | Open | Bit left-shifted into DU14 or right shifted into DU17 always low. Multiplication results erroneous. FCC disconnect. |
| | | | Gnd | Same as above. |

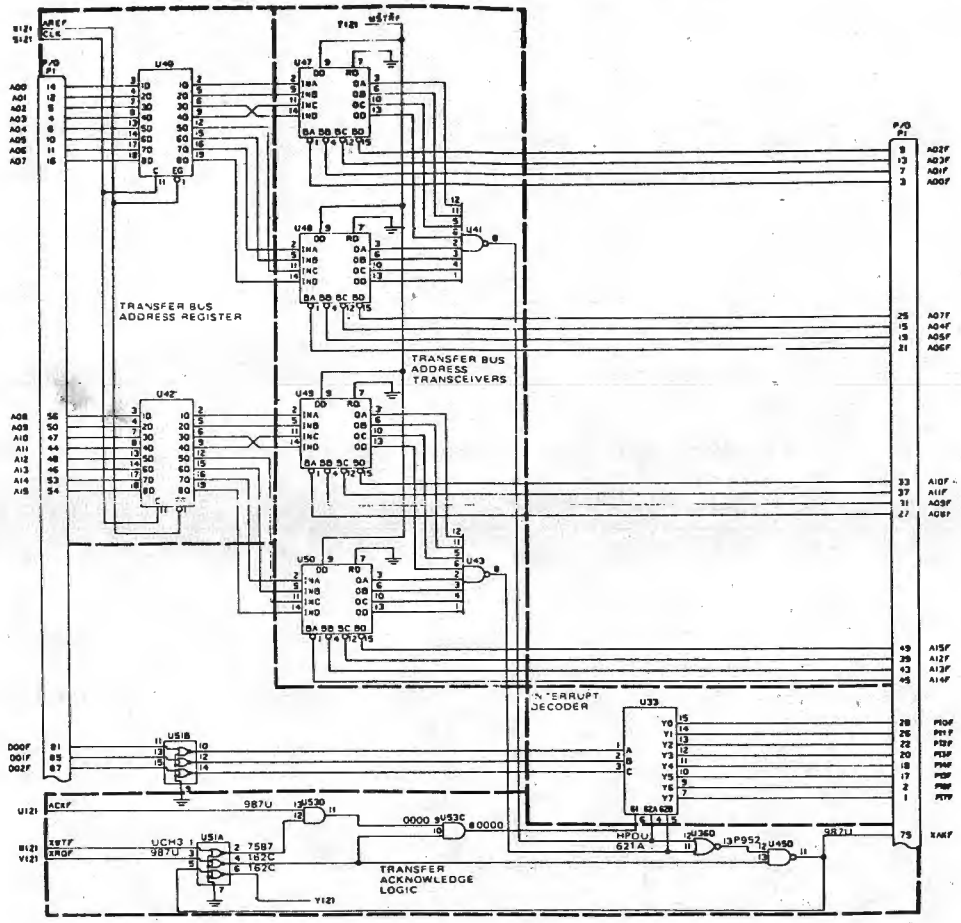
| Circuit | Function | Pin | Fault | Effect |
|---|---|---|---|---|
| Microprocessor DU17 cont't. | | | High | Bit left-shifted into DU14 or right-shifted into DU17 always high. Multiplication results erroneous. FCC disconnect. |
| | | Ro | Open | Bit right-shifted to shift/rotate multiplexer or input from shift/rotate multiplexer always low. Multiplication results erroneous. FCC disconnects. |
| | | | Gnd | Same as above. |
| | | RO | High | Bit right-shifted to shift/rotate multiplexer or input from shift/rotate multiplexer always high. |
| | | Q3 | Open | Bit right-shifted into DU17 or left-shifted to DU14 always low. Multiplication results erroneous. FCC disconnects. |
| | | | Gnd | Same as above. |
| | | | High | Bit right-shifted into DU17 or left-shifted to DU14 always high. Multiplication results erroneous. FCC disconnects. |
| | | Q0 | Open | Bit right-shifted to shift/rotate multiplexer or left shifted from shift/rotate multiplexer always low. Multiplication results erroneous. FCC disconnects. |
| | | | Gnd | Bit right-shifted to shift/rotate multiplexer. or left-shifted from shift/rotate multiplexer always high. Multiplication results erroneous. FCC disconnects. |
| | | F3 | Any | No effect. Pin not connected. |
| | | CN+4 | Any | No effect. Pin not connected. |
| | | OVR | Any | No effect. Pin not connected. |

| Circuit | Function | Pin | Fault | Effect |
|---|---|---|---|---|
| Micro-processor DU14 | Microprocessor DU14 handles bits 4-7 of the 16-bit APS word in response to instructions from the control registers. Effect of fault as if fault had occurred on DU17, except that different bit positions in the CAPS word are affected. | | | |
| Micro-processor DU18 | Microprocessor DU18 handles bits 8-11 of the 16-bit CAPS word in response to instructions from the control registers. Effect of pin faults is the same as if the fault had occurred on DU17, except that different bit positions in the CAPS word are affected. | | | |
| Micro-processor DU15 | Microprocessor DU15 handles bits 12 - 15 of the 16-bit CAPS word in response to instructions from the control registers. Effect of pin faults is the same as if the fault had occurred in DU14, except that different bits are affected. Some differences result from the use of bit 15 as the sign bit in numerical computation. | | | |

## APPENDIX D - PROCESSOR SCHEMATIC DIAGRAMS

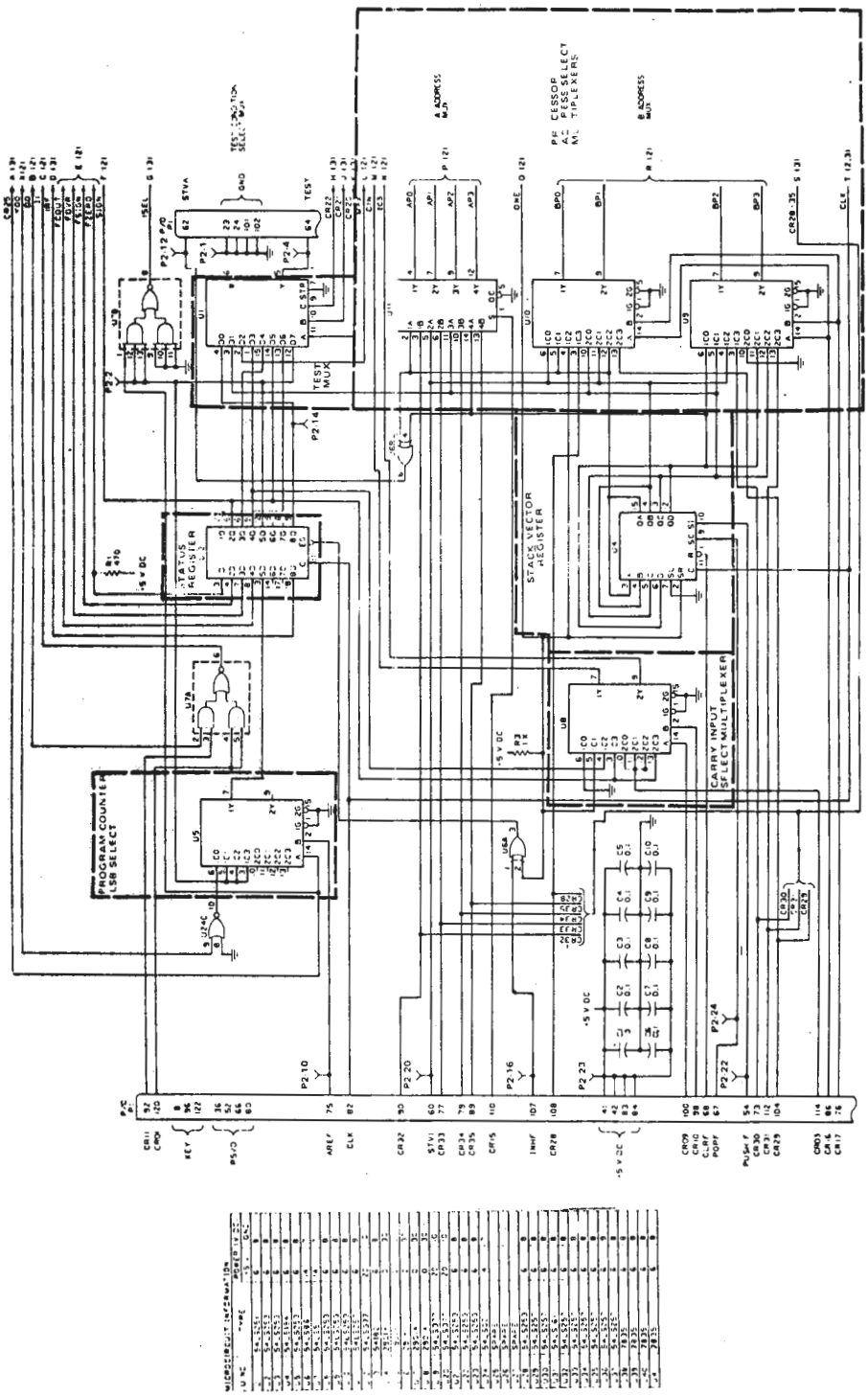


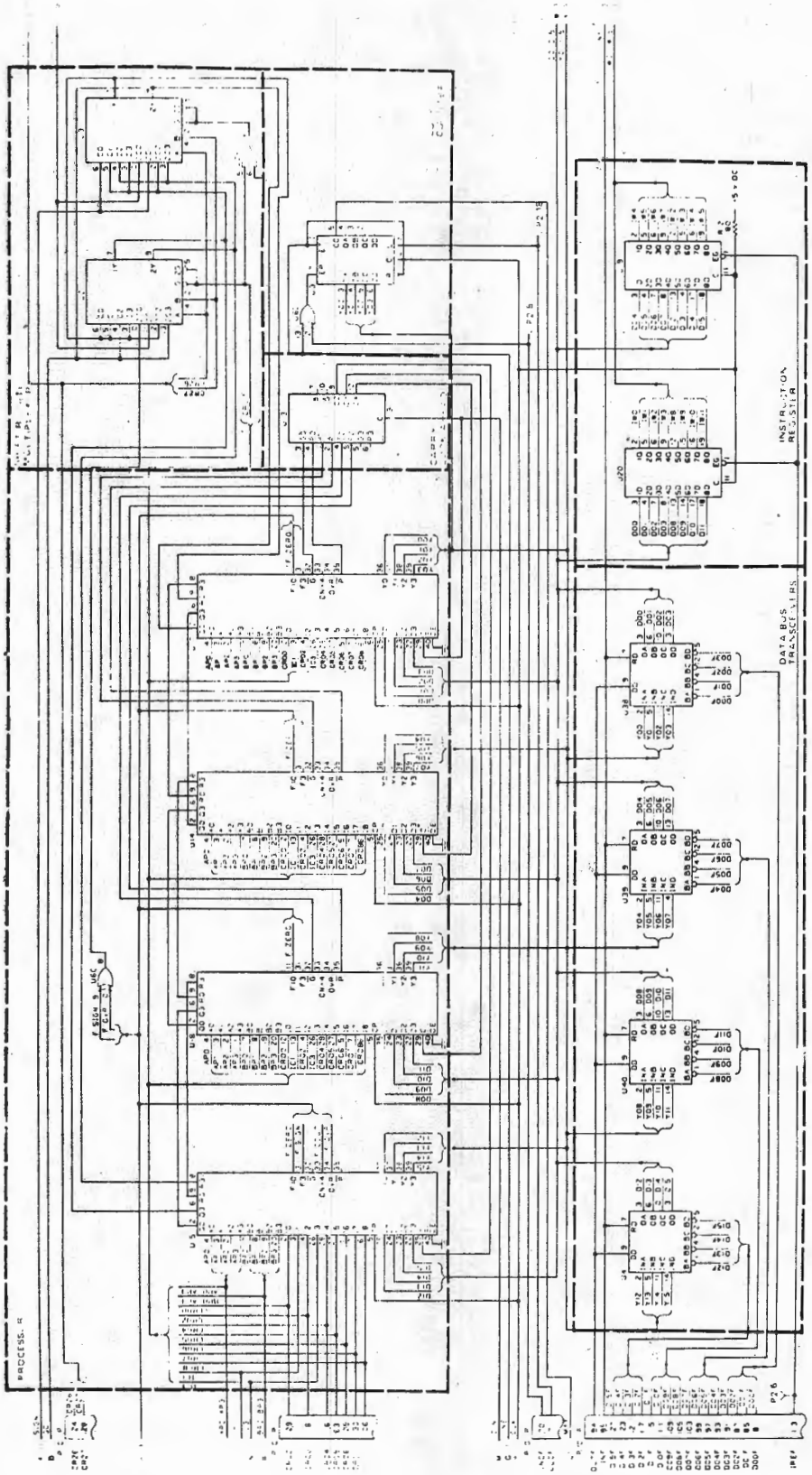


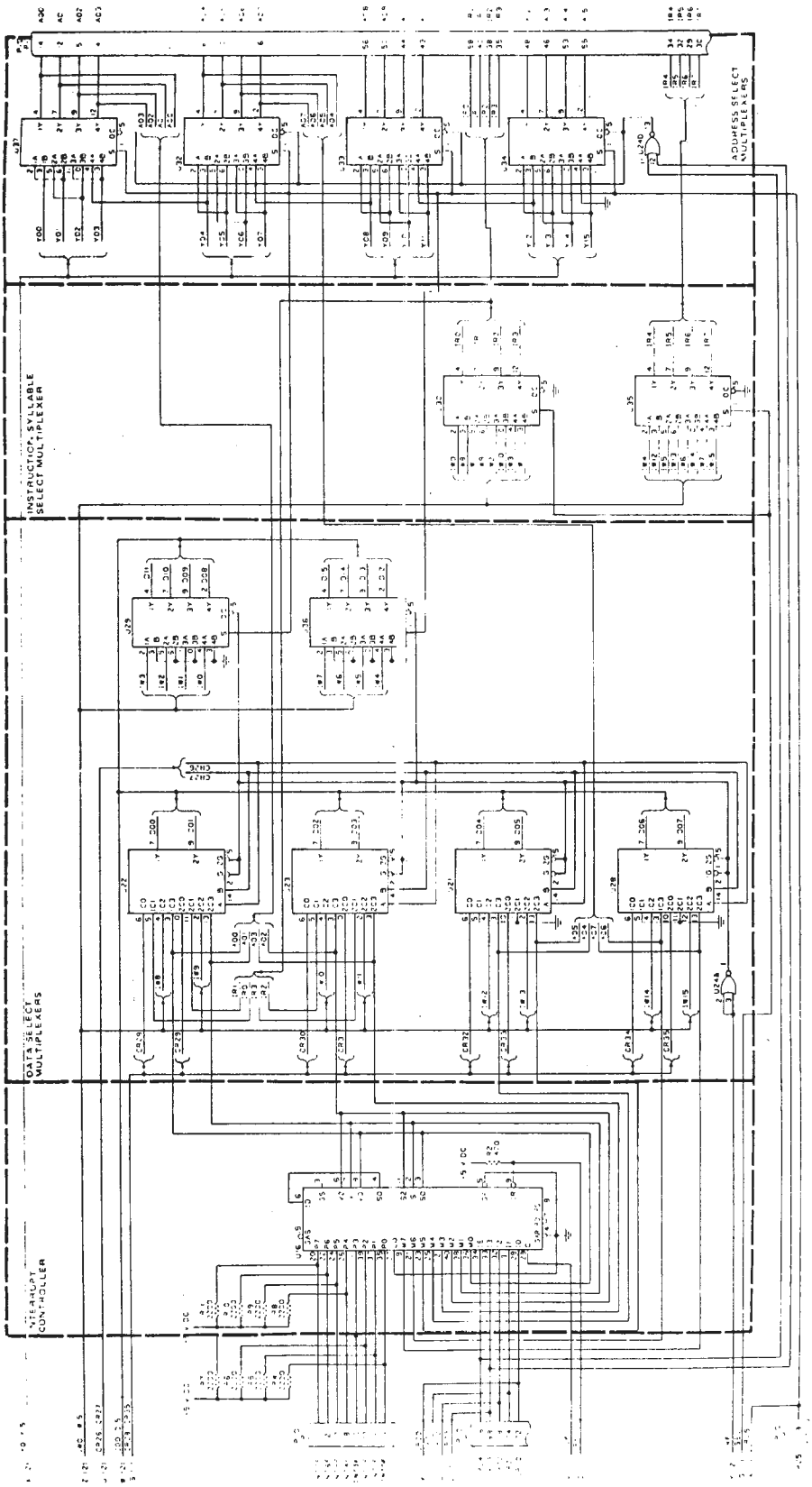



| U NO. | TYPE | POWER (DC) +5 | GND |
|---|---|---|---|
| U1 | am7643 | 18 | 9 |
| U2 | 5404 | 14 | 7 |
| U3 | am7643 | 18 | 9 |
| U4 | am7643 | 18 | 9 |
| U5 | am7643 | 18 | 9 |
| U6 | am7643 | 18 | 9 |
| U7 | am7643 | 18 | 9 |
| U8 | am7643 | 18 | 9 |
| U9 | am7643 | 18 | 9 |
| U10 | am7643 | 18 | 9 |
| U11 | am7643 | 18 | 9 |
| U12 | am7643 | 18 | 9 |
| U13 | am7643 | 18 | 9 |
| U14 | 2911 | 2 | 8 |
| U15 | 2911 | 2 | 8 |
| U16 | 2911 | 2 | 8 |
| U17 | am7643 | 18 | 9 |
| U18 | am7643 | 18 | 9 |
| U19 | NOT USED | | |
| U20 | 5474 | 14 | 7 |
| U21 | 54LS174 | 16 | 8 |
| U22 | 54LS175 | 16 | 8 |
| U23 | 54LS174 | 16 | 8 |
| U24 | 54LS377 | 20 | 10 |
| U25 | 54LS377 | 20 | 10 |
| U26 | NOT USED | | |
| U27 | 54LS112 | 16 | 8 |
| U28 | 54LS00 | 14 | 7 |
| U29 | 54LS74 | 14 | 7 |
| U30 | 29LS18 | 16 | 8 |
| U31 | [illegible] | [illegible] | [illegible] |
| U32 | 29LS18 | 16 | 8 |
| U33 | 54LS138 | 16 | 8 |
| U34 | 54LS175 | 16 | 8 |
| U35 | 54LC | 14 | 7 |
| U36 | 54LS02 | 14 | 7 |
| U37 | 5453 | 14 | 7 |
| U38 | 54LS22 | 14 | 7 |
| U39 | 5474 | 14 | 7 |
| U40 | 54LS377 | 20 | 10 |
| U41 | 54LS32 | 14 | 7 |
| U42 | 54LS377 | 20 | 10 |
| U43 | 54LS30 | 14 | 7 |
| U44 | 5438 | 14 | 7 |
| U45 | 5438 | 14 | 7 |
| U46 | 54LS38 | 16 | 8 |
| U47 | 7835 | 16 | 8 |
| U48 | 7835 | 16 | 8 |
| U49 | 7835 | 16 | 8 |
| U50 | 7835 | 16 | 8 |
| U51 | 7832 | 16 | 8 |
| U52 | 7832 | 16 | 8 |
| U53 | 54LS08 | 14 | 7 |

NOTE SIGNATURES ARE FOR FAULT ISOLATION PROCEDURE PART 1


STACK VECTOR REGISTER
CARRY INPUT SELECT MULTIPLEXER
PROGRAM COUNTER LSB SELECT
STATUS REGISTER
TEST MUX
A ADDRESS MUX
B ADDRESS MUX
P2.10
P2.20
P2.16
P2.23
P2.24
P2.22
-5 V DC


INSTRUCTION REGISTER
DATA BUS TRANSCEIVERS


INSTRUCTION SYLLABLE SELECT MULTIPLEXER
ADDRESS SELECT MULTIPLEXERS
DATA SELECT MULTIPLEXERS
INTERRUPT CONTROLLER

## COMPONENT LEVEL FAULTS

| CASE | ADDRESS / VARIABLE | FB03 VG #1 | FB01 $\dot{\theta}$ #1 | 3635 AP. ONE FAIL | 3318 EXEC FAIL $\theta$ | FB03 VG #3 | FB01 $\dot{\theta}$ #3 | 32E6 EXEC FAIL $\theta$ | 331A EXEC FAIL $\dot{\theta}$ | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Strip Chnl | 1 | 2 | | | 3 | | | | |
| 1A | Vert. Gyro #1 X-Leg open inbound | | | 0000<br>0001 | 0000<br>0003 | | | 0000<br>0003 | 0000<br>0003 | Pin P1A-31, X-Leg V.G. #1 Open On Pin removal; ATS Disconnect warning - NO DUAL, NO ALIGN at 1500 ft. |
| 1B | Vert. Gyro #1 Y-Leg open inbound | | | 0000<br>0000<br>0000 | 0000<br>0003<br>0000 | | | 0000<br>0003<br>0000 | 0000<br>0000<br>0000 | On pin removal ATS disconnect; prior to A/L track AP.ONEFAIL was cleared. |
| 2A | Vert. Gyro #1 hard shift to fixed value inbound | | | 0000<br>0001 | 0000<br>0003 | | | 0000<br>0003 | 0000<br>0000 | First fault was not of sufficient magnitude to fault sys., fault level was increased resulting in the CTA values shown, NO DUAL was annunciated upon engagement of A/L TRK. |
| 2B | Same as 2A but in Al. Arm | | | 0000<br>0001<br>0000 | 0000<br>0003<br>0003 | | | 0000<br>0003<br>0003 | 0000<br>0000<br>0000 | ATS Disc. WRN. on insertion NO DUAL did not light. AP.ONEFAIL reset. |
| 2C | Same as 2A but in AL. TRK above 150 ft. | | | 0000<br>0000 | 0000<br>0003 | | | 0000<br>0003 | 0000<br>0000 | ATS. dropped out on insertion; no ATS warning; NO DUAL did not annunciate |
| 2D | Same as 2A but in AL. TRK below 150 ft. | | | 0000<br>0000 | 0000<br>0003 | | | 0000<br>0003 | 0000<br>0000 | NO DUAL did not annunciate |
| 2E | Vert. Gyro #1 Validity false inbound | | | 0000<br>0001 | 0000<br>0000 | | | 0000<br>0000 | 0000<br>0000 | NO DUAL. did not annunciate |

## COMPONENT LEVEL FAULTS

| CASE | ADDRESS / VARIABLE | FB03 VG #1 | FB01 $\dot{\theta}$ #1 | 3635 AP. ONE FAIL | 3318 EXEC FAIL $\theta$ | FB03 VG #3 | FB01 $\dot{\theta}$ #3 | 32E6 EXEC FAIL $\theta$ | 331A EXEC FAIL $\dot{\theta}$ | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Strip Chat | | | | | | | | | |
| 3A | Vert. Gyro #1<br>Open X-Leg in<br>AL.ARM | | | 0000<br>1 | 0000<br>0000 | | | 0000<br>0000 | 0000<br>0000 | Disconnect of ATS on insertion;<br>No dual annunciated. |
| 3B | Vert. Gyro #1<br>Open Y-Leg in<br>AL.ARM | | | 0000<br>1<br>0 | 0000<br>3<br>3 | | | 0000<br>3<br>3 | 0000<br>3<br>0 | ATS Disc. on insertion.<br>No dual annunciated.<br>Exec. Fail Resets |
| 4A | Vert. Gyro #1<br>Open X-Leg in<br>AL.TRK above<br>150 Ft. | | | 0000<br>1 | 0000<br>3 | | | 0000<br>3 | 0000<br>3 | No dual annunciation, ATS<br>Disc. Annun. ATS Disc. |
| 4B | Vert. Gyro #1<br>Open Y-Leg<br>AL.TRK above<br>150 Ft. | | | 0000<br>1 | 0000<br>3 | | | 0000<br>3 | 0000<br>3 | No dual & ATS Disc.annunciated,<br>ATS Disc. |
| 5A | Same as 4A but<br>below 150 Ft. | | | 0000<br>1 | 0000<br>3 | | | 0000<br>3 | 0000<br>3 | ATS Disconn. without annunciation.<br>No dual not indicated. |
| 5B | Same as 4B but<br>below 150 Ft. | | | 0000<br>0 | 0000<br>3 | | | 0000<br>3 | 0000<br>0 | No dual did not indicate; ATS<br>Disc. & Ind. |

## COMPONENT LEVEL FAULTS

| CASE | ADDRESS / VARIABLE | FB03 VG #1 | FB01 $\dot{\theta}$ #1 | 3635 AP. ONE FAIL | 3318 EXEC FAIL $\theta$ | FB03 VG #3 | FB01 $\dot{\theta}$ #3 | 32E6 EXEC FAIL $\theta$ | 331A EXEC FAIL $\dot{\theta}$ | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Strip Chnl | | | | | | | | | |
| 6 | Vert. Gyro #1 | | | 0000 | 0000 | | | 0000 | 0000 | No dual flashed but did not |
| | Ramp Up | | | 1 | 3 | | | 3 | 0 | latch. |
| | Inbound | | | 0 | 3 | | | 3 | 0 | 3635 reset when no dual flashed. |
| 7 | Vert. Gyro #1 | | | 0000 | 0000 | | | 0000 | 0000 | No dual indicated. |
| | Ramp Down | | | 1 | 3 | | | 3 | 0 | |
| | Inbound | | | | | | | | | |
| 8A | Vert. Gyro #3 | | | 0000 | 0000 | | | 0000 | 0000 | |
| | Open X-Leg | | | 1 | 2 | | | 2 | 0 | |
| | Inbound | | | 0 | 2 | | | 2 | 0 | At 1500 Ft. No dual lit |
| 8B | Vert. Gyro #3 | | | 0000 | 0000 | | | 0000 | 0000 | No dual flashed, Reset 3635, |
| | Open Y-Leg | | | 1 | 2 | | | 2 | 0 | ATS Disc. Warning but ATS stayed. |
| | Inbound | | | 0 | 2 | | | 2 | 0 | |

## COMPONENT LEVEL FAULTS

| CASE | ADDRESS VARIABLE | FB1F N.A.#1 | 3380 N.A.EXEC. FAIL | 3635 ARONE FAIL | FB1F N.A.#3 | 334E EXEC. FAIL | |
|---|---|---|---|---|---|---|---|
| | Strip Chnl | 1 | | | 3 | | |
| 9A | Norm. Accel. #1 Open signal Inbound | | 0000<br>0 | 0000<br>0 | | 0000<br>0 | ATS held in; ATS Disc. warning at A/L TRK. |
| | | | 0000<br>3 | 0000<br>1 | | 0000<br>3 | Case 9A repeated with turbulence; Fault detected & NO DUAL |
| 9B | Norm. Accel. #1 Open Gnd. Inbound | | 0000 | 0000 | | 0000 | Case 9B involved signal gnd. Pulling pin has no effect. |

## COMPONENT LEVEL FAULTS

### FCC # 1 CTA WINDOW

| CASE | ADDRESS VARIABLE<br>STRIP CHNL | A1<br>FB07<br>ROLL<br>RATE 1 | A2<br>FB05<br>ROLL<br>RATE 1 | A3<br>3635<br>A.P.ONE<br>FAIL | A4<br>33E8<br>EXEC.<br>FAIL ROLL | B1<br>FB07<br>ROLL<br>RATE 3 | B2<br>FB05<br>ROLL<br>RATE 3 | B3<br>3386<br>EXEC.<br>FAIL ROLL | B4<br>33EA<br>EXEC.<br>FAIL ROLL | |
|---|---|---|---|---|---|---|---|---|---|---|
| 10A | Roll Gyro #1<br>Open X-Leg<br>in AL..ARM | | | 0000<br>1 | 0000<br>3 | | | 0000<br>3 | 0000<br>3 | No dual at A/L TRK |
| 10B | Roll Gyro #1<br>Open Y-Leg<br>in AL.ARM | | | 0000<br>1 | 0000<br>3 | | | 0000<br>3 | 0000<br>0 | No dual at A/L TRK |
| 11A | Roll Gyro #1<br>Ramp Up<br>Inbound | | | 0000 | 0000 | | | 0000 | 0000 | No dual at A/L TRK |
| 11B | Same as 11A<br>but in AL..ARM | | | 0000<br>1 | 0000<br>3 | | | 0000<br>3 | 0000<br>0 | No dual at 1100 FT.<br>(When Comparators<br>Tripped). |
| 11C | Same as 11A | | | 0000<br>1 | 0000<br>3 | | | 0000<br>3 | 0000<br>0 | No dual at 1100 FT.<br>(When Comparators<br>Tripped). |
| 11D | Same as 11A<br>but in AL..TRK<br>below 150 Ft. | | | 0000<br>1 | 0000<br>3 | | | 0000<br>3 | 0000<br>0 | Landing Completed<br>Without No Dual<br>Indication. |

## COMPONENT LEVEL FAULTS

FCC # 1 CTA WINDOW

| CASE | ADDRESS VARIABLE<br><br>STRIP CHNL | A1<br>FB1B<br>LOC.DEV.<br># 1 | A2<br>339A<br>EXEC.FAIL<br>LOC.DEV. | A3 | A4<br>33E8 | B1<br>FB1B<br>LOC.DEV.<br># 1 | B2<br>3368 | B3 | B4<br>33EA | |
|---|---|---|---|---|---|---|---|---|---|---|
| 12A | No Localizer Output In Inbound | | 0000<br>6003<br>4003 | | 0000<br>0 | | 0000<br>6003<br>4003 | | 0000<br>0 | No dual at AL.TRK |
| 12B | Same as 12A but In AL.ARM | | 0000<br>4003 | | 0000<br>0000 | | 0000<br>4003 | | 0000<br>0000 | No dual indicated when fault detected (at 1000 Ft.) |
| 12C | Same as 12A but In AL.TRK above 150 Ft. | | 0000<br>4003 | | 0000<br>0000 | | 0000<br>4003 | | 0000<br>0000 | No dual indicated when fault detected (at 1000 Ft.) |
| 12D | Same as 12A but in AL.TRK below 150 Ft. | | 0000<br>6003 | | 0000<br>0000 | | 0000<br>6003 | | 0000<br>0000 | Fault detected; Land completed; No dual did not illuminate. |
| 14A | Localizer No. 1, Validity 1 in AL. ARM | | 0000<br>4013 | | 0000 | | 0000<br>4013 | | 0000 | No dual at AL.TRK |
| 14B | Localizer No. 1, Validity 2 in AL. ARM | | 0000<br>4023 | | 0000 | | 0000<br>4023 | | 0000 | No dual at AL.TRK |
| 16 | Lateral Accel. No. 1 Ramp Up in AL. ARM | FB1D | 33B4 | 33B4<br>EXEC.FAIL<br><br>0000<br>400A | 3635<br>AP. ONE FAIL<br>0000 | FB1D | | 3382<br>EXEC.FAIL<br><br>0000<br>400A | 3635<br>AP.ONE FAIL<br>0000 | No dual at 1100 Ft. when comparator tripped. |

## COMPONENT LEVEL FAULTS

### FCC #1 CTA WINDOW

| CASE | ADDRESS VARIABLE<br><br>STRIP CHNL | A1 | A2 | A3<br>3364<br>AP.TWO<br>FAIL | A4<br>3365<br>AP.ONE<br>FAIL | B1 | B2 | B3<br>3364<br>AP.TWO<br>FAIL | B4<br>3365<br>AP.ONE<br>FAIL | |
|---|---|---|---|---|---|---|---|---|---|---|
| 1/A | Pitch Servo Coil Discrete Fault Inbound | | | 0000 | 0000 | | | 0000 | 0000 | Instant disengage; No dual at A/L. TRK. Servo simulator panel pitch coil switch to fault. Box 2 engaged first. |
| 1/B | Same as 1/A but In AL.ARM | | | 0000<br>0000 | 0000<br>0000 | | | 0000<br>0000 | 0000<br>0000 | Both bathandles dropped. |
| 1/B | Rerun | | | | | | | | | On 3rd retry, only affected channel disengaged, no dual at A/L. TRK. |
| 1/C | Same as 1/A, but In AL.TRK above 150 Ft. | | | | | | | | | Affected FCC disengaged, No dual |
| 1/D | Same as 1/A, but In AL.TRK below 150 Ft. | | | | | | | | | Affected FCC disengaged; No dual not indicated. |
| 18A | Pitch Servo Coil Current Ramp Up Inbound | | | | | | | | | Affected channel disengaged; No dual at A/L.TRK |
| 19A | Roll Servo Discrete Open Inbound | | | | | | | | | |
| 19B | Same as 19A but In AL.ARM | | | | | | | | | Affected bathandle dropped; No dual indicated. |
| 19C | Same as 19A but In AL.TRK above 150 Ft. | | | | | | | | | Affected bathandle dropped; No dual indicated. |
| 19D | Same as 19A but In AL.TRK below 150 Ft. | | | | | | | | | Affected bathandle dropped; No dual not indicated. |

| Case | Circuit | Pin No. | Pin Function | Open Pin Faults |
|---|---|---|---|---|
| 20a | DU18 2901 No.3 (data bits 8-11) | 24 | Data Input Bit 1 | Upon opening, both FCC's disengaged. NORMAL-STANDBY switch was in STANDBY. |
| 20 | | 11 | F = 0 | Faulted FCC disengaged. SPLIT and NO DUAL annunciated. Faulted pin transmits signal to the status register. |
| 20b | | 4 | Address bit $A_0$ | Faulted FCC disengaged. SPLIT, NO DUAL, and NO ALIGN annunciated at initiation of AL TRK. |
| 20c | | 17 | Address bit $B_0$ | Both FCC's disengaged. AP DISC and SPLIT annunciated. |
| 20d | | 37 | Data Output bit 1 | Faulted FCC disengaged. AP DISC and SPLIT annunciated on disengagement. NO DUAL annunciated at AL TRK. |
| 21 | CU16 | 1 | | Fault had no effect on computer operation. Pin used only in reset of stack vector and transfer bus access control registers. |
| 22a | CU30 Control Register 29LS18 | 1 | Data Input bit 0 | Faulted FCC disengaged. NO DUAL, NO ALIGN, and SPLIT annunciated. Pin 1 is coupled to Pin 2 (CR35, see fault following) and, when the next address control prom sets OE low, to direct input bit DI of microprogram sequencer CU14. |
| 22b | | 2 | Data Output bit $Q_0$ | Faulted FCC disengaged. SPLIT, NO DUAL annunciated. Faulted bit drives control line CR35, which is address bit A3 of the 2901's when the processor address multiplexer couples CR35 to address line A03. |
| 22c | | 10 | Data Output bit Y2 | Faulted FCC disengaged. SPLIT, NO DUAL, and NO ALIGN annunciated. Faulted pin is direct input bit D3 to microprogram sequencer CU15 when bit D3 is not being controlled by instruction mapper prom CU7. In turn, CU15 outputs this bit as address bit A7 to the control store proms when CU1E is in direct address mode. |

| Case | Circuit | Pin No. | Pin Function | Open Pin Faults |
|---|---|---|---|---|
| 22a | CU10 | 4 | Data Input bit D1 | Faulted FCC disengaged; other FCC went to CWS. CMD DISC annunciated. Control line CR34 is latched to bit D1 on rising clock pulse, and when next address control prom sets OE low, to direct input bit D0 of microprogram sequencer CU14. When selected by data select multiplexer DU28, CR34 is used as data bit D06. |
| 22c | | 4 | | In a repeat of previous case, faulted FCC disengaged. Other FCC stayed in CMD. SPLIT and NO DUAL annunciated. |
| 22f | | 12 | Data Input bit D2 | Faulted FCC disengaged, other FCC stayed in CMD. NO DUAL and SPLIT annunciated. Control line CR33 is latched to Pin 12 on rising clock pulse. CR33 is used as processor A address bit AP1 when connected by the A address multiplexer. Also, CR33 can be coupled to processor input data bit D05 by data select multiplexer DU21. |
| 23a | CU15 Micro-program Sequencer 2911 | 9 | Not Zero | Faulted FCC disengaged. SPLIT and NO DUAL annunciated at ALIGN point in landing. Pin 9 forces all outputs of CU15 to zero when it is low. Open pin prevented HI signal from control register CU21 from reaching CU15, zeroing all outputs and causing erroneous address to control store memory. |
| 23b | | 19 | Not FE | Faulted FCC disengaged. Processor halted. System failed to capture glide slope. The FE signal is one of four used to control the operation of the 2911 micro-program sequencer. In most combinations of the signals, the absence of the FE signal causes a push or pop of a counter stack in addition to a jump. |
| 23c | | 19 | Not FE | Repeat of previous fault. Faulted FCC disengaged. Other FCC stayed in CMD. SPLIT and NO DUAL annunciated. |
| 23d | | 20 | PUP | Faulted FCC disengaged. NO DUAL and SPLIT annunciated. PUP is Push/Pop control signal. Open pin prevents pushing the microprogram counter contents onto the internal stack. |

| Case | Circuit | Pin No. | Function | Open Pin Faults |
|---|---|---|---|---|
| 23e | CU15 Microprogram Sequencer 2911 | 10 | S0 Address Source Selection Control | Faulted FCC disengaged. NO DUAL and SPLIT annunciated. S0 is one of the four signals used in selecting the source of the next address. S0 open generally results in wrong source producing a jump to the wrong address. |
| 23f | | 5 | D2 Direct Input bit D2 | Faulted FCC disengaged. NO DUAL and SPLIT annunciated. D2 is one of four bits which can be selected as the output of the 2911. Fault would cause the wrong control store memory address on selection of direct input when the bit should be high. |
| 23g | | 15 | Y3 Output bit | Faulted FCC disengaged. NO DUAL and SPLIT annunciated. Y3 is one of four output bits of the 2911. This bit being open causes the wrong microinstruction to be selected whenever this bit should be high. |

| Case | Circuit | Pin No. | Pin Function | Grounded Pin Faults |
|---|---|---|---|---|
| 24 | CU2 Hex Inverter | 8 | Inverter Output | Faulted FCC disengaged. Processor stopped. This signal fans out to several points, including NAND gate CU35C, which outputs the Read Enable signal to the data bus transceivers. NAND output is stuck high so that processor cannot read the CAPS data bus. |
| 25 | CU36 Quad. NOR Gate | 4 | Gate Output | Faulted FCC disengaged. NO ALIGN, SPLIT, and NO DUAL annunciated. Faulted pin being stuck low results in NAND gate U35A being stuck high, disabling the data bus transceivers from writing on the CAPS data bus. |
| 26 | CU28 Quad. NAND Gate | 3 | Gate Output | Faulted FCC disengaged. NO ALIGN, SPLIT, and NO DUAL annunciated. Processor stopped. Fault results in BROF being stuck high, so that processor cannot access CAPS bus. Also, all interrupt inputs to the interrupt controller are set high. |
| 27 | CU36 Quad. NOR Gate | 11 | Gate Output | Faulted FCC disengaged. NO ALIGN, SPLIT, and NO DUAL annunciated. Processor stopped. Fault results in the processor being unable to transmit XAKF (transfer acknowledge) on the CAPS control bus. XAKF is stuck high. |
| 28 | DU2 Shift/ Rotate Multi-plexer | 2 | Control Input | Faulted FCC disengaged. SPLIT, NO ALIGN, NO DUAL annunciated. Faulted processor stopped. Fault causes wrong data to be inserted into microprocessor in some shift operations. |
| | | 14 | Control Input | Faulted FCC disengaged. Fault causes wrong data to be inserted into microprocessors during some shift operations. |

## Grounded Pin Faults

| Case | Circuit | Pin No. | Function |
|---|---|---|---|
| 29 | DU17 Micro-processor | 36 | Y00 |
| | | 37 | Y01 |
| | | 38 | Y02 |
| | | 39 | Y03 |
| 30 | DU14 Micro-processor | 36 | Y04 |
| | | 37 | Y05 |
| | | 38 | Y06 |
| | | 39 | Y07 |
| 31 | DU18 Micro-processor | 36 | Y08 |
| | | 37 | Y09 |
| | | 38 | Y10 |
| | | 39 | Y11 |
| 32 | DU15 Micro processor | 36 | Y12 |
| | | 37 | Y13 |
| | | 38 | Y14 |
| | | 39 | Y15 |

In each case, the faulted FCC disengaged. The faulted processor halted immediately. The y pins are the processor output pins for computed data. Under certain conditions, processor output is a memory address which is connected to the CAPS address bus, rather than data. Corruption of addresses is apparently the cause of the immediate processor halts.

| Case | Circuit | Pin No. | Function | Grounded Pin Faults |
|---|---|---|---|---|
| 33 | D017 Micro-processor | 32 | $\bar{G}$ | |
| | | 35 | $\bar{P}$ | |
| 34 | D014 Micro-processor | 32 | $\bar{G}$ | Faulted FCC disengaged. Faulted processor halted immediately. |
| | | 35 | $\bar{P}$ | |
| 35 | D018 Micro-processor | 35 | $\bar{P}$ | |
| 36 | D016 Interrupt Controller 2914 | 16 | V2 | Faulted FCC disengaged. Faulted processor stopped immediately. V2 is the most significant bit of the interrupt vector output of the 2914. This bit is also address bit A02 of the CAPS address bus when the vector output is enabled, and is hard-wired to address line A02. |
| 37 | | 17 | V1 | Faulted FCC disengaged. Faulted processor stopped immediately. V1 is the middle bit of the three-bit interrupt vector of the 2914. This pin is hard-wired to CAPS address bus line A01. |
| 38 | | 18 | V0 | Faulted FCC disengage. Faulted processor stopped immediately. V0 is the least significant bit of the interrupt vector output of the 2914. This pin is hard-wired to CAPS address bus line A00. |
| 39 | | 28 | I0 | Faulted FCC disengaged. Faulted processor stopped immediately. I0 is a micro-instruction bit to the 2914. |
| 40 | | 31 | I1 | Faulted FCC disengaged. Faulted processor stopped immediately. I1 is a micro-instruction bit to 2914. |
| 41 | | 32 | I2 | Faulted FCC disengaged. Faulted processor stopped immediately. I2 is a micro-instruction bit to the 2914. |

## GROUNDED PIN FAULTS

| Case | Circuit | Pin No. | Function | |
|---|---|---|---|---|
| 42a | DU16 Interrupt Controller 2914 | 34 | Instruction Enable | Faulted FCC disengaged. Faulted processor halted. Pin 34 is a logic-low instruction enable which should only go low when the instruction lines I0-I3 have been set. The pin stuck low causes the 2914 to read erroneous instructions. |
| 42b | | 26 | P4 Interrupt Request | Faulted FCC disengaged. Faulted processor halted. Pin 34 is a logic-low interrupt request. With the fault inserted, an interrupt request at priority 4 is generated whenever the corresponding mask bit is not set and a higher priority unmasked interrupt is not present. |
| 42c | | 39 | P2 Interrupt Request | Faulted FCC disengaged. Faulted processor halted. This is the same situation as in the previous case, except at a lower priority level. |
| 42d | | 20 | P7 Interrupt Request | Faulted FCC disengaged. Faulted processor halted. This is the same situation as in the previous two cases, except at the highest priority level. |
| 42e | | 25 | M4 Mash Bit | Faulted FCC disengaged. Faulted processor halted. This fault prevents priority Level 4 interrupts from being masked. |
| 42f | | 19 | M7 Mask Bit | Faulted FCC disengaged. Faulted processor halted. This fault prevents highest priority interrupts from being masked. |

FCC #1 CTA WINDOW

| CASE | ADDRESS VARIABLE<br>STRIP CHNL | A1 | A2 | A3<br>AP. ONE FAIL<br>3635 | A4<br>AP.TWO FAIL<br>3634 | B1 | B2 | B3 | B4 | |
|---|---|---|---|---|---|---|---|---|---|---|
| 43 | Open Roll Gyros 1 & 2 in AL.ARM | | | 0000<br>0001 | 0000<br>0001<br>0000 | | | | | Disconnected on Second Fault. 3634 flashed 0001 before reverting to 0000. |
| 44 | Open roll Gyros 1 & 3 to FCC #1 in AL.ARM | | | | | | | | | Sensors 2 & 3 still valid into box 2. No disconnect; No dual at AL.TRK. |
| 45 | Ramp Up Vert. Gyro #1 in APP; Open Vert. Gyro #2 in AL.ARM | | | | | | | | | Two sensors lost; both boxes disengaged. |

BIBLIOGRAPHY

AC 25, 1309-1, System Design Analysis, Federal Aviation Administration, September 7, 1982.

AC 120-28C, Criteria for Approval of Category III Landing Weather Minima, Federal Aviation Administration, March 9, 1984.

Bjurman, B. E., et al., Airborne Advanced Reconfigurable Computer System (ARCS), NASA-CR-145024, Prepared for Langley Research Center, NASA by Boeing Commercial Airplane Company under contract NAS1-13654, August 1976.

Conn, R. B., et al, "CAST - A Complementary Analytic - Simulative Technique for Modeling Complex, Fault-Tolerant Computing Systems," AIAA Computers in Aerospace Conference, Los Angeles, California, November 1977.

DOT/FAA/CT-82/140, Digital Flight Control System Validation Technology Assessment, July 1982.

DOT/FAA/CT-82/154, Integrated Assurance Assessment of a Reconfigurable Digital Flight Control System, Volume I, April 1983.

Hitt, E. and D. Eldredge., "A Review and Application of Analytical Models in Fault Tolerant Avionics Validation," Proceedings of IEEE/AIAA 5th Digital Avionics Systems Conference, Fall 1983.

Karlin, S., A First Course in Stochastic Processes, Academic Press, 1966.

Lockheed-Georgia Company Engineering Report LG81E0126, Simulator Investigation Plan for Digital Flight Controls Validation Technology, as revised 10 April 1981.

MIL-M-38510, Microcircuits, General Specification for.

Military Standardization Handbook 217C, Reliability Prediction of Electronic Equipment, United States Air Force, Rome Air Development Center, 9 April 1979, with Notice 1, Supplement, 1 May 1980.

MIL-STD-1629A, Procedures for Performing a Failure Mode, Effects and Criticality Analysis.

Mulcare, D. B., et al., Industry Perspective on Simulation Methods for Validation and Failure Effects Analysis of Digital Flight Control Avionics, NASA CR-152234, Moffett Field, California, February 1979.

NASA RDFCS System Interface Document, April 8, 1981.

NASA Report prepared for Langley Research Center by Raytheon Company, Contract NAS1-12668, Reliability Model Derivation of a Fault-Tolerant, Dual, Spare-Switching Digital Computer System, March 1974.

Ng, Y. W., and A. Avizienis., ARIES 76 User's Guide, National Science Foundation Report No. NSF-MCS-7203633-78944, University of California, Los Angeles, Report No. UCLA-ENG-7894, December 1978.

Nuclear Regulatory Commission Report NUREG-0492, Fault Tree Handbook.

RTCA Document DO-178, Software Considerations in Airborne Systems and Equipment Certification, November 1981.

## ACRONYMS AND ABBREVIATIONS

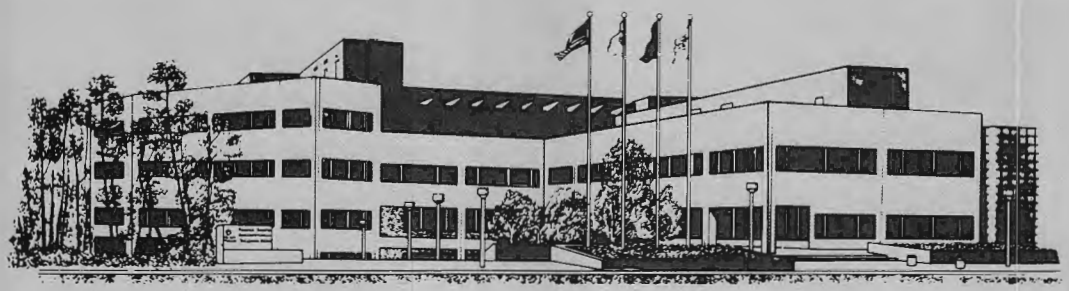
| | |
|---|---|
| A/L | Approach/Land |
| AC | Alternating Current |
| AC | Advisory Circular |
| ADI | Automatic Direction Indicator |
| ALU | Arithmetic Logic Unit |
| ARIES | Automated Reliability Interactive Estimation System |
| ARINC | Aeronautical Radio Incorporated |
| CAPS | Computer Aided Production Simulator |
| CAST | Complementary Analytic Simulative Technique |
| CARE | Computer Aided Reliability Evaluator |
| CARSRA | Computer-Aided Redundant System Reliability Analysis |
| CTA | CAPS Test Adapter |
| DFCS | Digital Flight Control System |
| DOT | Department of Transportation |
| DSP | Discrete Switch Panel |
| EFMA | Executive Failure My A |
| EFMB | Executive Failure My B |
| EFOA | Executive Failure Other A |
| EFOB | Executive Failure Other B |
| EFW | Executive Failure Word |
| FAA | Federal Aviation Administration |
| FAR | Federal Acquisition Regulation |
| FCC | Flight Control Computer |
| FMEA | Failure Mode and Effect Analysis |
| HSI | Horizontal Situation Indicator |
| Hz | Hertz |
| IAA | Integrated Assurance Assessment |
| IC | Integrated Circuit |
| ILS | Instrument Landing System |
| LVDT | Linear Voltage Differential Transducer |
| mA | Milliampere |
| MDICU | Modular Digital Interface Control Unit |
| NA | Normal Accelerometers |
| NASA | National Aeronautics and Space Administration |
| PROM | Programmable Read-Only Memory |
| RAM | Random Access Memory |
| RBDCP | Reliability Block Diagram Computer Program |
| RDFCS | Redundant Digital Flight Control System |
| REL | Reliability |
| REL COMP | Reliability Computers |
| RTCA | Radio Technical Commission for Aeronautics |
| SSP | Servo Simulation Panel |
| TASRA | Tree Aided System Reliability Analysis |
| WAI | Warning Annunciation Indicator |
| XMTR | Transmitter |

U.S. Department of Transportation
**Federal Aviation Administration**

DOT/FAA/CT-88/10

# HANDBOOK-VOLUME II
# DIGITAL SYSTEMS VALIDATION

## CHAPTER 4
## QUADRUPLEX DIGITAL FLIGHT CONTROL SYSTEMS



**PREPARED BY:**

**COMPUTER RESOURCE MANAGEMENT, INC.**
**950 HERNDON PARKWAY, SUITE 360**
**HERNDON, VIRGINIA 22070**

**PREPARED FOR:**

**FEDERAL AVIATION ADMINISTRATION**
**TECHNICAL CENTER**
**ATLANTIC CITY INTERNATIONAL AIRPORT, NEW JERSEY 08405**

## NOTICE

This document is disseminated under the sponsorship of the U.S. Department of Transportation in the interest of information exchange. The United States Government assumes no liability for the contents or use thereof.

The United States Government does not endorse products or manufacturers. Trade or manufacturers' names appear herein solely because they are considered essential to the objective of this report.

TABLE OF CONTENTS


| Section | | Page |
|---|---|---|
| 1. | INTRODUCTION | 4-1 |
| 2. | BACKGROUND | 4-3 |
| 3. | THE ASSURANCE ASSESSMENT PLAN | 4-13 |
| 4. | THE USE OF SIMULATION AS AN ASSURANCE TOOL | 4-19 |
| | 4.1. Assurance Assessment | 4-19 |
| | 4.2. Digital Flight Control System Assurance Methodology | 4-19 |
| | 4.3. Simulation Development | 4-19 |
| | 4.4. Basic System Architecture | 4-20 |
| | 4.5. Developing the Flight Control System | 4-24 |
| | 4.6. Establishing System Reliability | 4-29 |
| | 4.7. The Airplane Simulation | 4-35 |
| 5. | VERIFYING THE SIMULATION | 4-41 |
| 6. | SUMMARY | 4-49 |
| BIBLIOGRAPHY | | 4-53 |
| GLOSSARY | | 4-55 |
| ACRONYMS AND ABBREVIATIONS | | 4-58 |

## LIST OF ILLUSTRATIONS


| Figure | | Page |
|---|---|---|
| 2-1 | BASELINE SYSTEM ARCHITECTURE | 4-4 |
| 2-2 | QUADRUPLEX DIGITAL FLIGHT CONTROL SYSTEM ARCHITECTURE | 4-5 |
| 2-3 | DIGITAL FLIGHT SYSTEM LIFE CYCLE | 4-6 |
| 2-4 | RECONFIGURABLE DIGITAL FLIGHT CONTROL SYSTEM FACILITY SETUP (FLIGHT TEST PALLET) | 4-8 |
| 2-5 | RECONFIGURABLE DIGITAL FLIGHT CONTROL SYSTEM FACILITY SETUP (SIMULATION SETUP) | 4-9 |
| 2-6 | INTERDEPENDENCE OF ASSURANCE METHODS | 4-10 |
| 2-7 | MULTILEVEL TESTING | 4-11 |
| 3-1 | ARCHITECTURAL DESIGN TASKS | 4-14 |
| 3-2 | MULTILEVEL-TESTING CLOSURE | 4-16 |
| 3-3 | RECONFIGURABLE DIGITAL FLIGHT CONTROL SYSTEM ALL-UP SIMULATION COMPONENTS | 4-17 |
| 4.3-1 | CONTROL LAW USED IN REDUNDANT FLIGHT CONTROL SYSTEM CHANNELS | 4-23 |
| 4.5-1 | TOP-LEVEL DIGITAL FLIGHT CONTROL SYSTEM CHANNEL LOGIC DESIGN | 4-25 |
| 4.5-2 | SYNCHRONIZATION PREDICATE/TRANSITION NETWORK | 4-27 |
| 4.5-3 | PARTIAL PREDICATE/TRANSITION NETWORK | 4-28 |
| 4.5-4 | TYPICAL DISCRETE-EVENT SIMULATION RESULTS | 4-30 |
| 4.5-5 | CONTROL GRAPH OF TOP-LEVEL SOFTWARE | 4-31 |
| 4.6-1 | RELIABILITY MODEL FOR IMPLEMENTED ARCHITECTURE | 4-32 |
| 4.7-1 | DIAGRAM OF STATE-VARIABLE AIRPLANE SIMULATION | 4-36 |
| 5-1 | AUTOMATED TEST SCENARIO | 4-43 |
| 5-2 | IMPLEMENTATION OF DIGITAL FLIGHT CONTROL SYSTEM SIMULATOR FACILITY | 4-45 |
| 5-3 | MECHANIZATION OF SIMULATOR TESTING | 4-46 |
| 6-1 | CROSS-CHANNEL SYNCHRONIZATION TIME-HISTORIES | 4-50 |


## LIST OF TABLES


| Table | | Page |
|---|---|---|
| 3-1 | DIGITAL FLIGHT CONTROL SYSTEM TESTING SCENARIO | 4-15 |
| 3-2 | RECONFIGURABLE DIGITAL FLIGHT CONTROL SYSTEM FACILITY CAPABILITIES | 4-18 |
| 4.3-1 | DEVELOPMENT PHASE ACTIVITIES AND PRODUCTS | 4-21 |
| 4.3-2 | STABILITY AUGMENTATION GAIN SCHEDULING | 4-24 |
| 4.5-1 | CYCLING STATE LOOP TIMING | 4-26 |
| 4.6-1 | DIGITAL FLIGHT-CONTROL SYSTEM RELIABILITY ASSESSMENTS | 4-33 |
| 4.6-2 | DEGRADATION OF FLYING-QUALITIES: 5-HOUR MISSION | 4-34 |
| 4.6-3 | DEGRADATION OF FLYING-QUALITIES: 10-HOUR MISSION | 4-34 |
| 4.7-1 | RELAXED STATIC STABILITY FLIGHT CASES | 4-37 |
| 5-1 | AUTOMATED TESTING TOOLS | 4-44 |
| 5-2 | MULTILEVEL TEST CASES AND MODES | 4-47 |

# 1. INTRODUCTION

The purpose of this chapter is to provide background information pertinent to assurance assessments of quadruplex Digital Flight Control Systems (DFCS), to identify and discuss some of the current problems associated with assurance assessments, and to present related evaluation approaches and techniques.

Assurance assessment is a collective phrase that covers design correctness, system verification, and system validation. By ensuring that a proposed system functions according to design specifications, assurance assessments serve as tracking guidelines during system development.

The material contained in this chapter relies heavily on the development and validation of a double fail-operational DFCS architecture reported in Quadruplex Digital Flight Control Assessment, DOT/FAA/CT-86/30 (D. B. Mulcare, L. E. Downing, and M. K. Smith). This report serves as the basis for this presentation, and generous use has been made of its data, text, and illustrations.

## 2. BACKGROUND

The first DFCSs were introduced in the mid-1970s. They used fairly conventional architectures and were largely copies of prior analog systems. These systems used parallel replicated lanes with sensors fanning into the processors, which in turn fanned out to actuators or displays. Figure 2-1 shows an expansion of this architecture for the pitch-axis of an Augmented Fly-By-Wire (AFBW) function. Except for cross-channel communication, all signal paths are dedicated analog paths. Fan-in is minimized by having one set of sensors feed directly into each computational channel. In figure 2-1, each channel broadcasts its received sensor inputs to the other three channels.

These early Flight Control Systems (FCS) were flight-critical for a very limited time (e.g., autoland). By 1975, the Federal Aviation Administration (FAA) recognized the trend from analog systems to their digital implementations. The FAA also recognized the challenges that would accompany this transition to a new technology (Reed and Boothe, 1977). More recently, the FAA has identified a need to become familiar with the emerging full-time, full-authority, "flight-critical" fly-by-wire/light systems (Larsen and Carro, 1986). Besides the functional criticality of the systems, the concepts of envelope limiting, flying-qualities management, and system reconfiguration without loss of redundancy during the processing of hard and soft faults (transparent recovery) have brought new issues to the forefront. As a result, new areas of concern have emerged as fault-tolerant systems come online.

Advanced systems become interdisciplinary as their complexity increases. The overall aspects of certification become more system-integrity oriented. As a result, there is a need to formulate a uniform approach to assurance assessments. From a reliability point of view, analytical assessments of critical functions are demanding and tedious. They include the evaluation of the dissimilar versions of hardware and software used in the various lanes of redundancy, the need for backup systems, faulted recovery techniques, status instrumentation, and pilot workload.

Quadruplex Digital Flight Control Assessment (D. B. Mulcare, L. E. Downing, and M. K. Smith) defines a double fail-operational DFCS that was designed, analyzed, implemented, and validated relative to system fault-tolerance (figures 2-2 and 2-3). In this chapter, a subset of the DFCS system, the pitch-axis control functions, will be used to demonstrate important portions of an integrated approach to assurance assessment. The system-level, fault-tolerant design was developed around an engineering development model of the L-1011 FCS and will be verified using a predicate/transition network simulation (Barton, Mulcare, and LeBlanc, AIAA Paper, 1985).

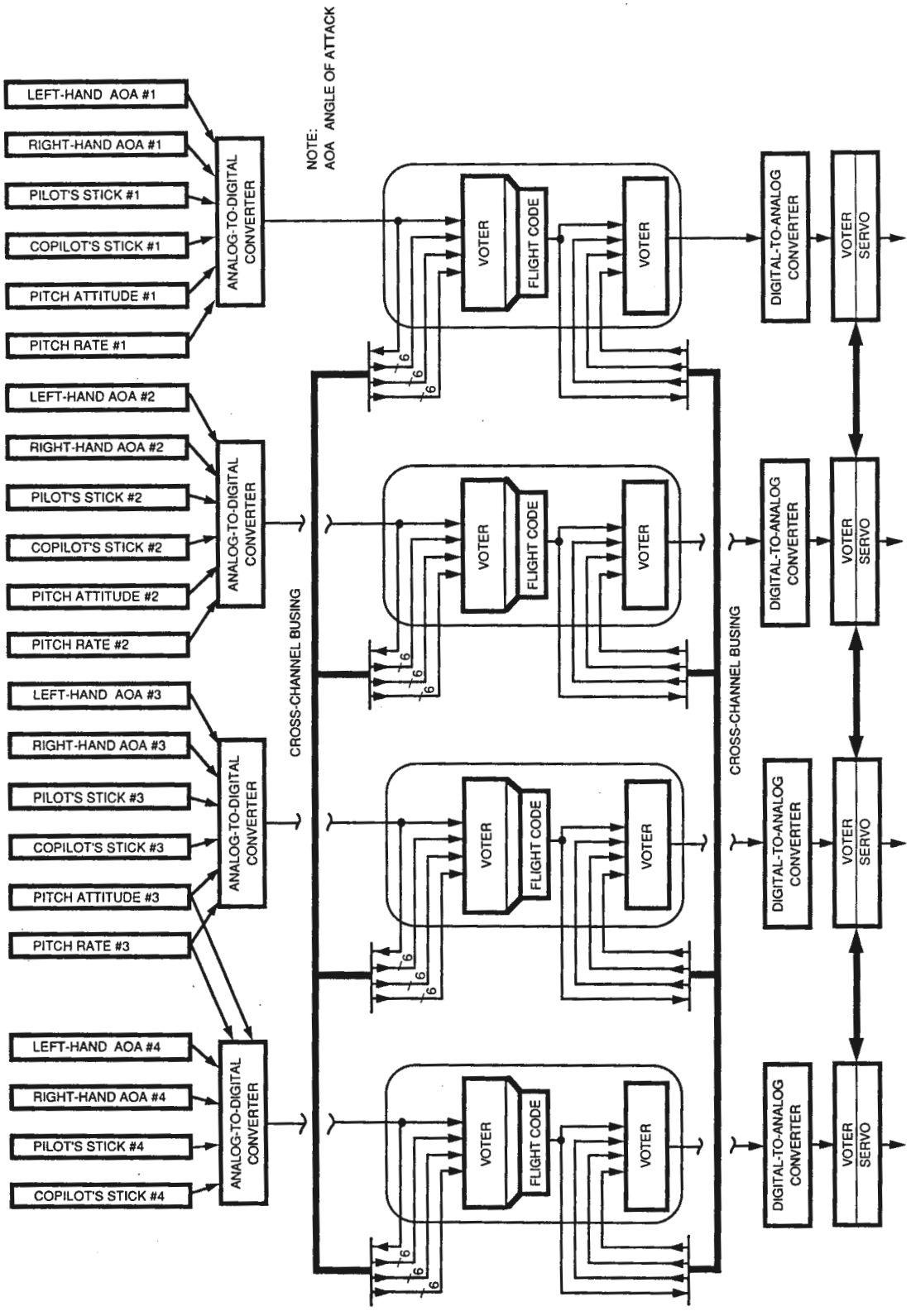



FIGURE 2-1. BASELINE SYSTEM ARCHITECTURE

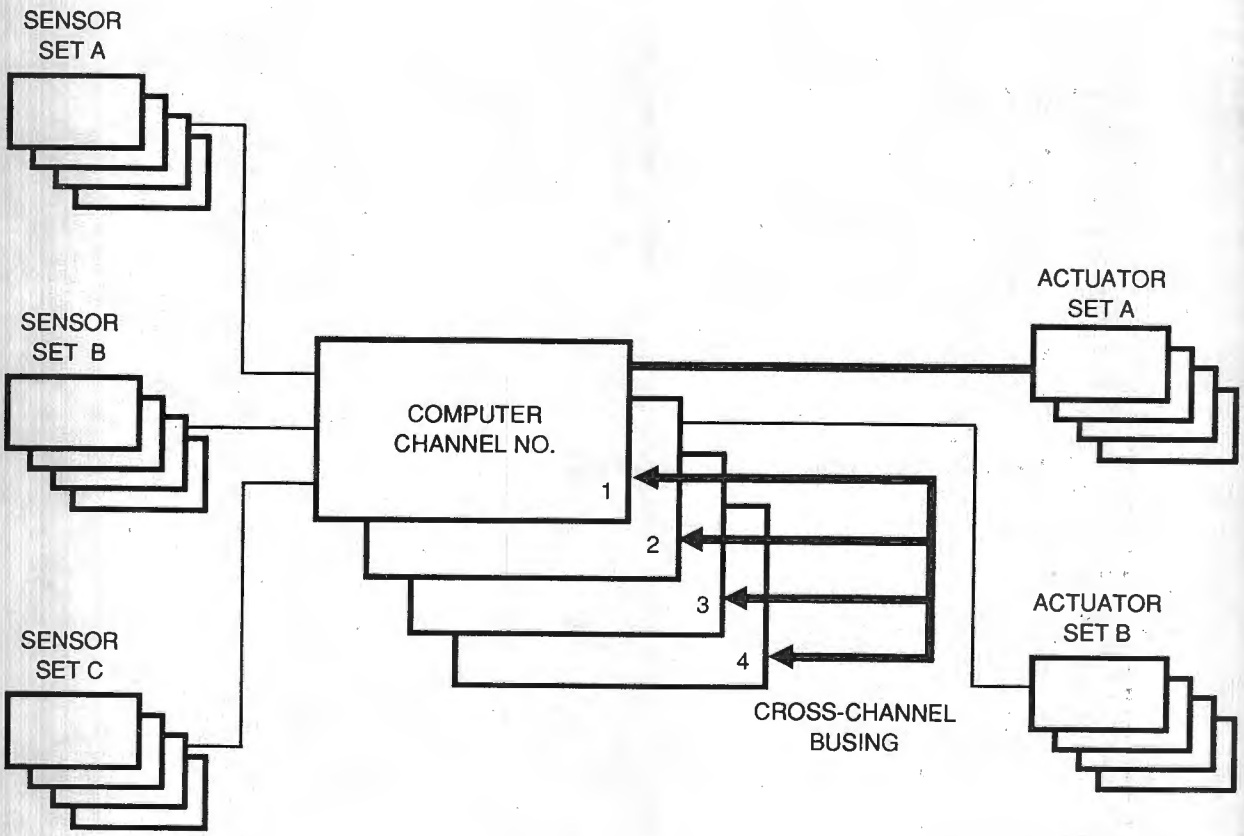



FIGURE 2-2. QUADRUPLEX DIGITAL FLIGHT CONTROL SYSTEM ARCHITECTURE

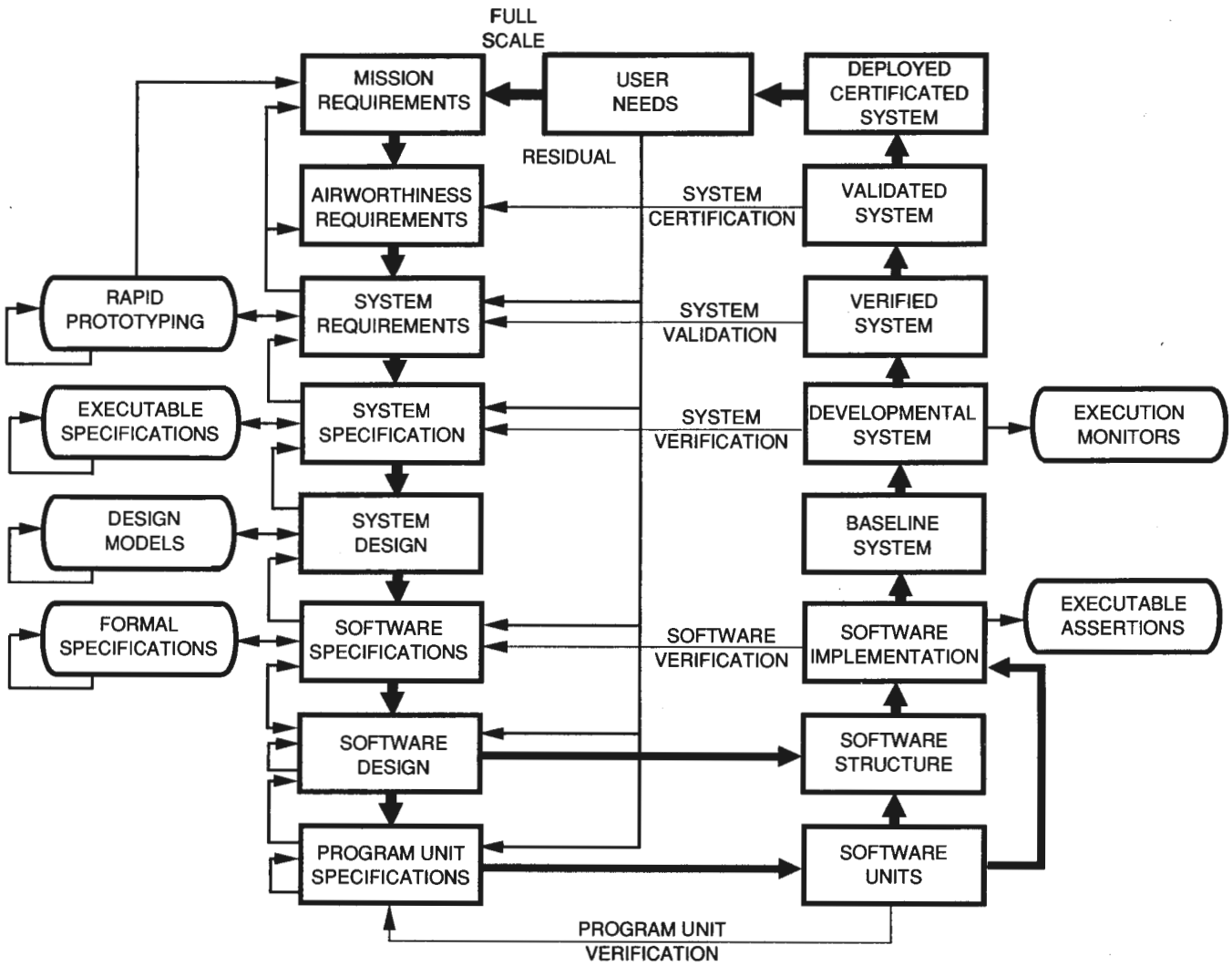



FIGURE 2-3. DIGITAL FLIGHT SYSTEM LIFE CYCLE

The data contained in this chapter have been developed from the DFCS and systems simulator shown in figures 2-4 and 2-5. (This system was located at Ames Research Center, Moffett Field, California.) The original dual-dual architecture was transformed into a quadruplex architecture by making software changes that extended the fault-tolerance capability to full-time flight-critical operation. The resulting implementation was not optimal for flight application, but it worked well for demonstrating a verified design and served to illustrate the integrated assurance methodology necessary throughout the development process for flight-critical applications.

With the introduction of full-time flight-critical control functions, substantially more effort must be directed toward guaranteeing airworthiness than was necessary for flight-phase critical functions. A tremendous increase in system complexity has also been incurred with (1) the advent of fault-tolerant (error-correcting) architectures, (2) the myriad assurance tools necessary to confirm system airworthiness, and (3) digital implementation. To attain requisite airworthiness of the flight-critical control system, lower levels of digital implementation must be examined. An integrated assurance methodology is essential for compliance with the provisions of AC 25.1309.1.

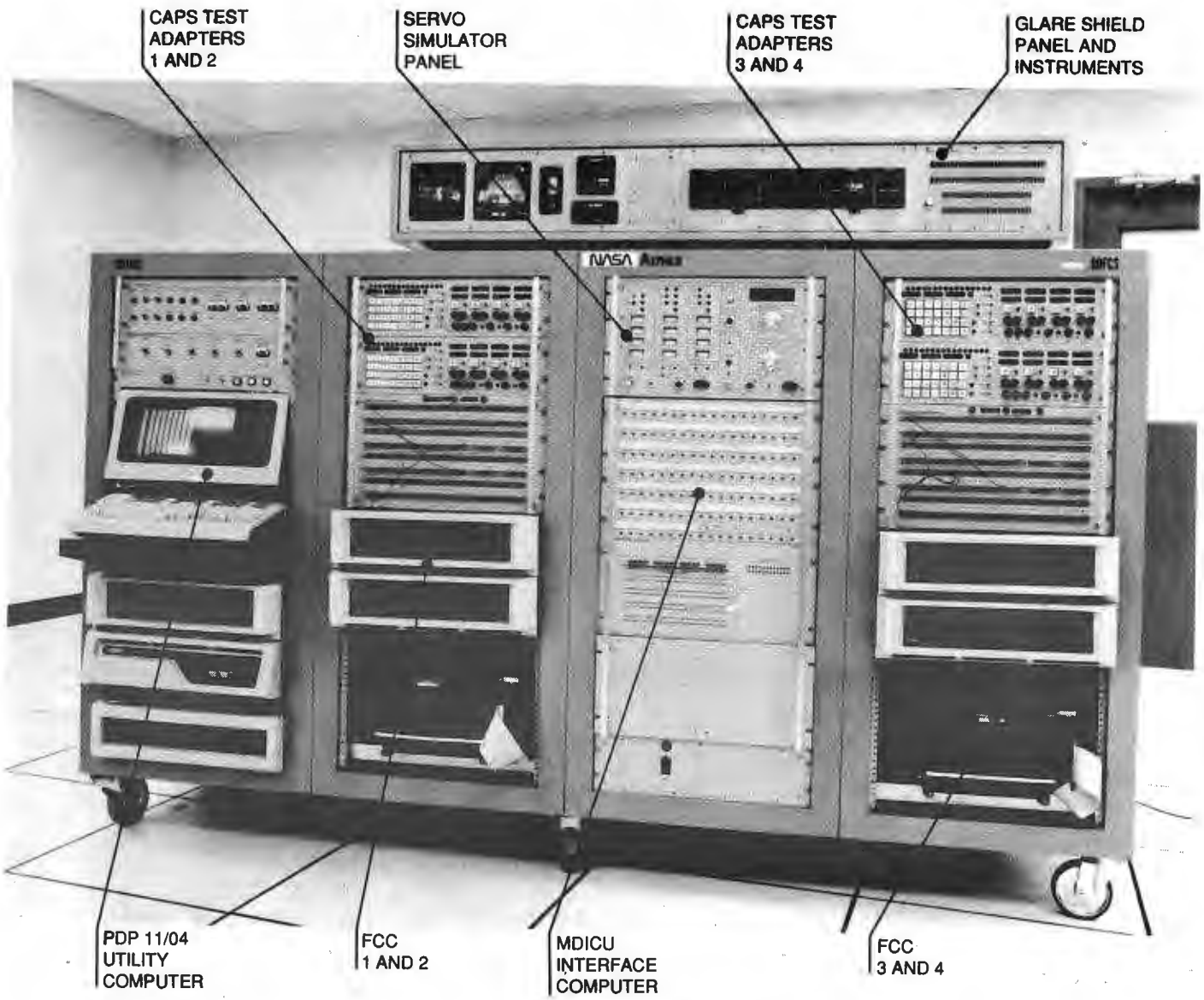
The assurance tools and methods required for use with these advanced systems have not been fully developed. It is not clear that their use ensures that the required level of reliability is achieved. In civil aircraft, current reliability levels are set at $10^{-9}$ or less.

In a comprehensive assessment process, several dimensions of an integrated assurance method should be reviewed and acknowledged. These dimensions include the performance, analysis, testing, and inspection of reliability analyses, failure effects analyses, simulation and functional performance assessment methods. The integration assurance methodology is shown in figure 2-6. Analysis is the dominant method for reaching high levels of assurance in the early development of an FCS, particularly when only models or abstractions are available. At this stage, a systems walk-through can be especially valuable.

Analysis forms the basis of conclusive testing. It must be accomplished at various stages of the development process to maximize the effectiveness of testing, as shown in figure 2-7. Testing based on proper analysis assures the system designer that the implementation is in accord with design specifications and requirements. The coincident application of various test cases derived through analysis provides an increased level of confidence in the correctness of the design and its implementation.

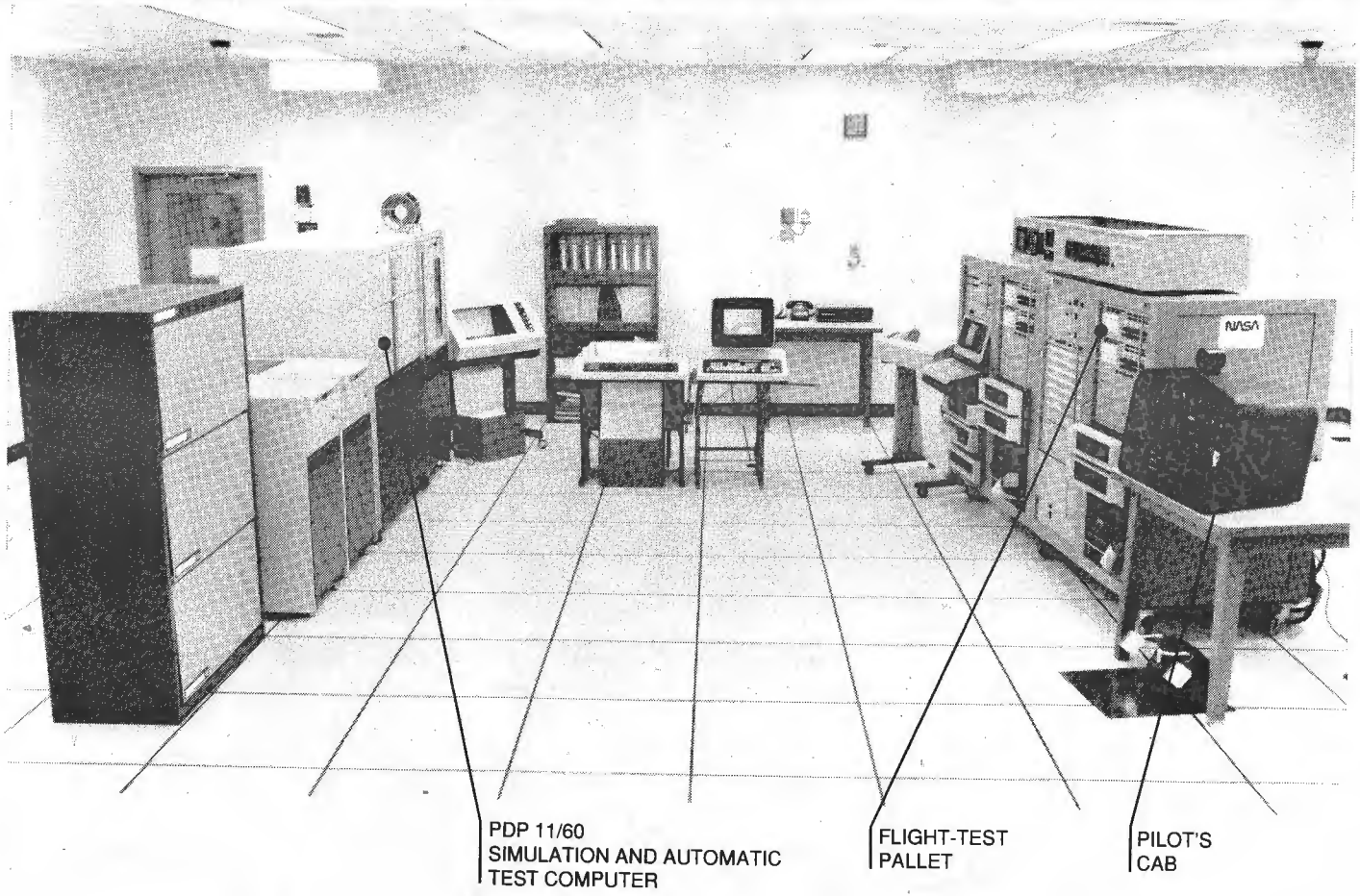
An integrated approach to proving implementation correctness is essential because testing can only be applied to the actual article. Testing primarily examines the validity of analytical results and the underlying assumptions.

The assessment methods presented are not necessarily the best or only techniques for every flight-critical DFCS use. The methods do, however, provide a representative example of the airworthiness assurance of the full-time, flight-critical DFCSs that will be making their appearances in the near future. Although the pitch-axis example discussed here is scaled, it is representative of the challenges in certifying double fail-operational quadruplex DFCSs.




**(a) Flight-test Pallet**

FIGURE 2-4. RECONFIGURABLE DIGITAL FLIGHT CONTROL SYSTEM FACILITY SETUP (FLIGHT TEST PALLET)




**(b) Simulation Setup**

FIGURE 2-5. RECONFIGURABLE DIGITAL FLIGHT CONTROL SYSTEM FACILITY SETUP (SIMULATION SETUP)

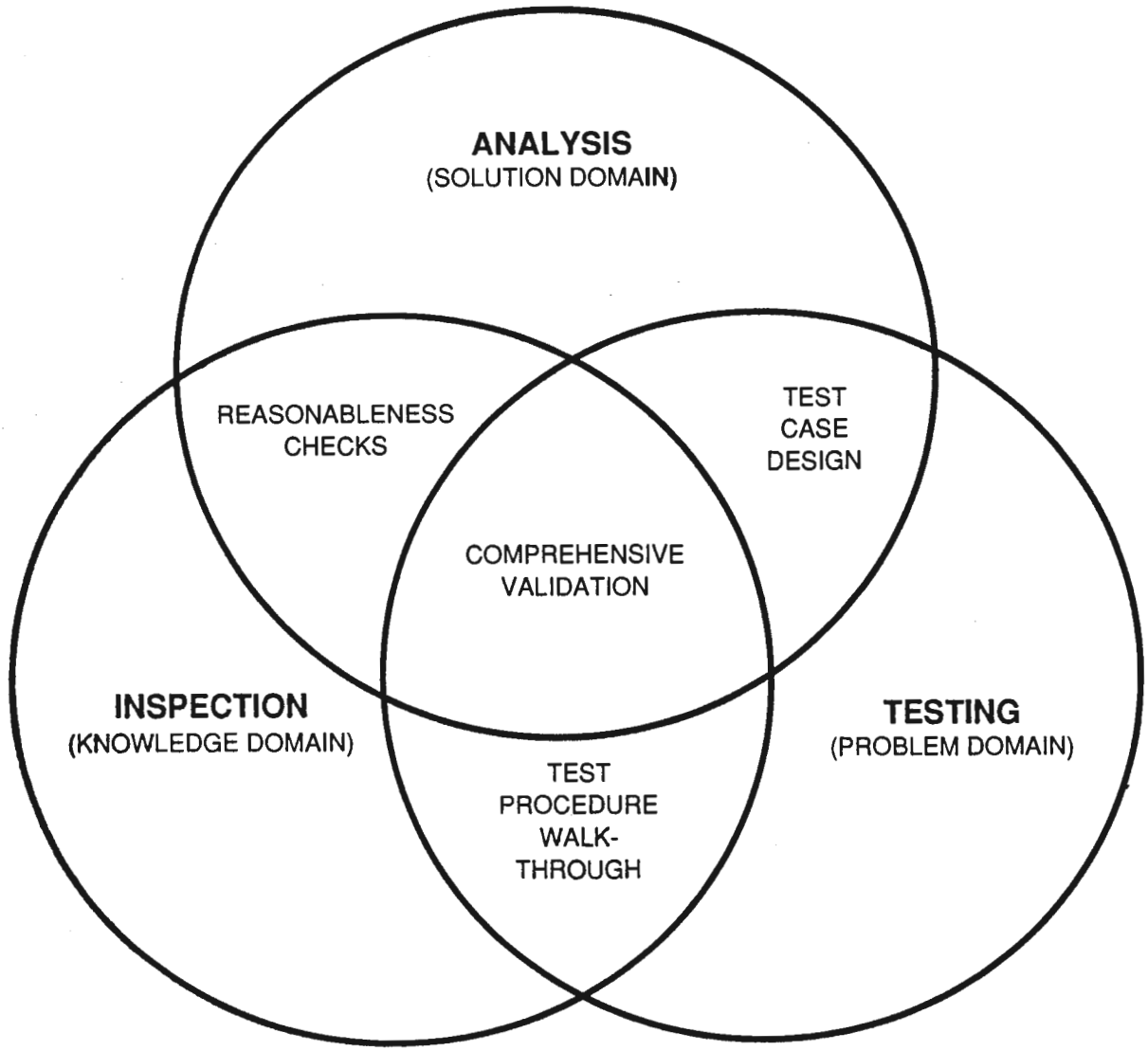



FIGURE 2-6. INTERDEPENDENCE OF ASSURANCE METHODS

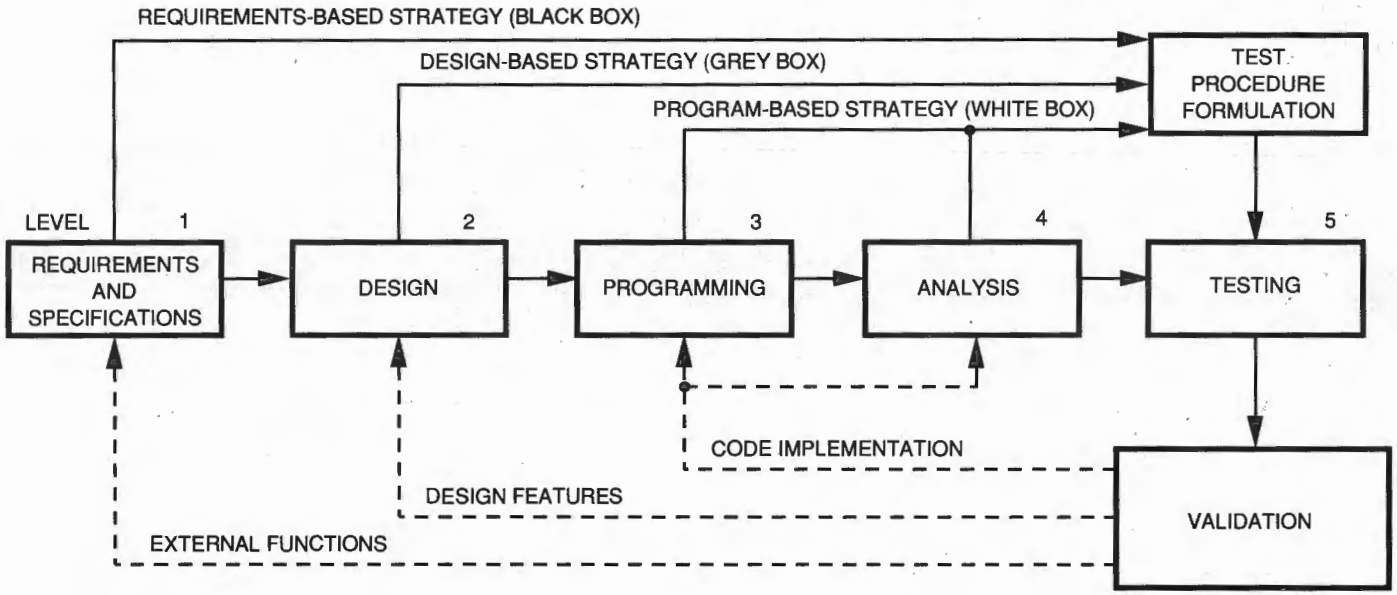



FIGURE 2-7. MULTILEVEL TESTING

There is a considerable difference in the airworthiness demands for a full-time flight-critical system or function and a flight-phase critical function. The increased demands for airworthiness assurances are primarily a function of system exposure time (the time from last system validation to completion of the present use). When a flight-critical function is implemented digitally, there is concern about the capability of ensuring its airworthiness. Therefore, these examples seek to demonstrate the requisite technology and its application in a representative DFCS development program for a pitch-axis augmented fly-by-wire/light system.

## 3. THE ASSURANCE ASSESSMENT PLAN

The FAA AC 25.1309-1 recommends procedures for demonstrating compliance with the requirements of Part 25 of the Federal Aviation Regulations (FAR) for flight-essential and flight-critical avionics systems. This publication outlines the use of a quantitative safety analysis that may include probability analysis, fault tree analysis, failure-modes and effects analysis, and other comparable techniques such as systems simulation. Such an integrated approach is discussed in detail in Ness, 1983 and chapter 3 of this handbook. This chapter describes a system-level failure-effects simulation that focuses on flying-qualities degradation and real-time multilevel assessment of software behavior stressing architectural fault tolerance.

The flying-qualities degradation testing is a conventional technique and incorporates an all-up simulation including sensors, servos, research cockpit pilot interface, aircraft model, and test electronics. The real-time multilevel testing is a considerably more sophisticated technique and a valuable new testing methodology for flight-critical systems. This approach to multilevel demonstration testing is a means of establishing high levels of confidence in the airworthiness of a given flight-critical DFCS.

Figures 2-3 and 3-1 present the developmental activities and expected results for a verification and validation process as it relates to a digital flight system design. Table 3-1 indicates five typical levels of testing that might be undertaken. The top level, most generally seen, is conventional system functional testing. Later on in this chapter the flying qualities for a Relaxed Static Stability (RSS) airplane test case are examined. Multilevel control structures that were also related to simulation developmental activities are examined coincidentally, as shown in figure 3-2. To conclusively examine sophisticated flight-critical systems requires that significant emphasis be placed on automated testing and wideband instrumentation. This level of testing is illustrated in table 3-1.

One major aspect of flight-critical DFCS verification and validation is the verification that logic for the top-level systems using fault-tolerant architecture has been implemented. Other testing might check the design and the redundancy of the design verification channel, because fault-tolerant architectures demand intricate hardware and software logic to maintain cross-channel coordination.

This kind of verification testing occurs in the latter stages of system development and is addressed using the fidelity of an all-up FCS simulator. If the test simulator lacks an automated test capability that would permit other than system-level checks, then it is necessary to develop real-time test programs running in parallel (commonly known as real-time test monitors). Real-time test monitors are extremely valuable in achieving the necessary level of assurance. Figure 3-3 presents the components of an all-up simulation facility (the

Reconfigurable Digital Flight Control System (RDFCS) Facility at Ames Research Center), and table 3-2 highlights some of the more significant capabilities of that facility.

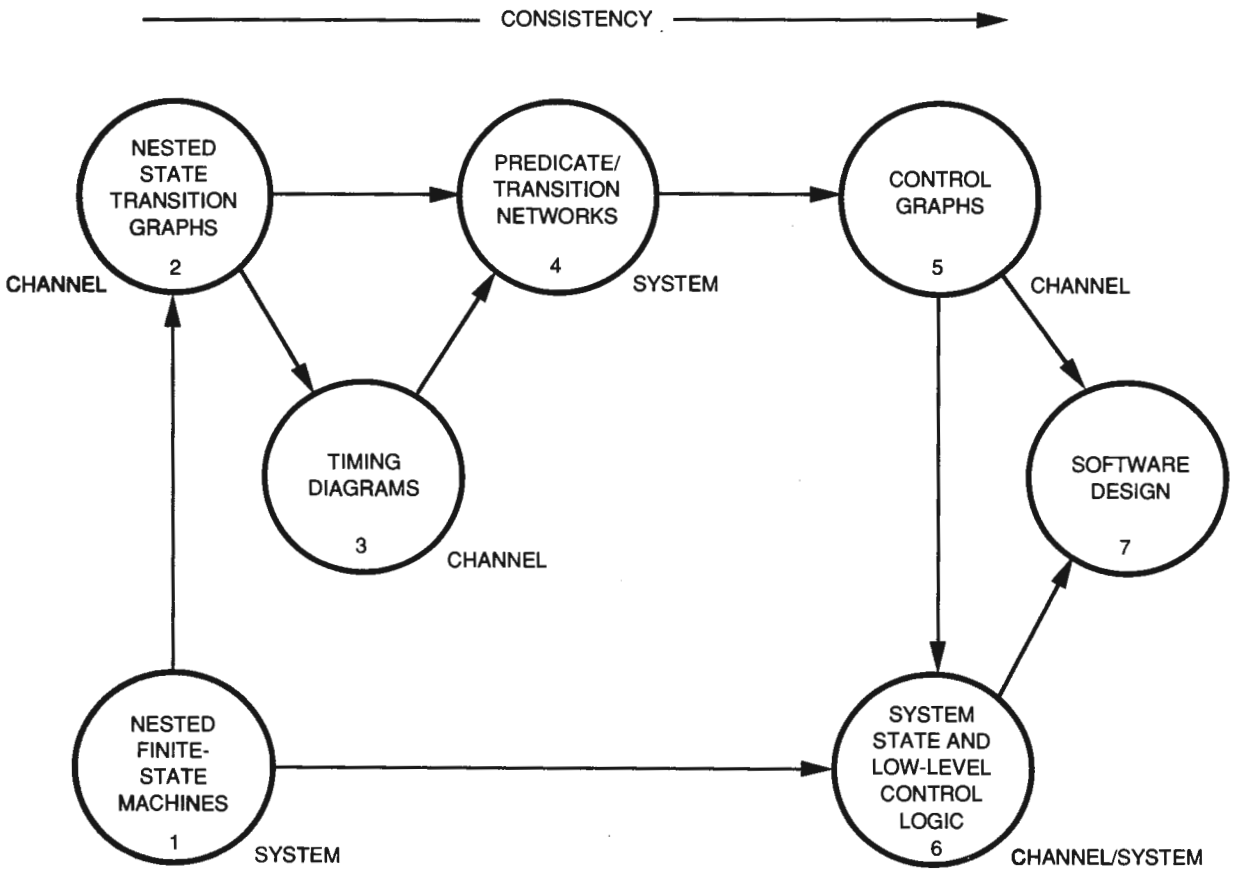



FIGURE 3-1. ARCHITECTURAL DESIGN TASKS

TABLE 3-1. DIGITAL FLIGHT CONTROL SYSTEM TESTING SCENARIO

| Type of Testing | Focus | Execution Monitor | Monitor Location | Primary Concern |
|---|---|---|---|---|
| System validation | System performance | None | N/A | Fault Flying qualities |
| System verification | System state | Nested finite-state machines | Simulation/ test computer | Faulted states |
| | Channel synchronization | Predicate/ transition network | Single channel | Single-point failures |
| | Mode/fault logic | Boolean expressions | Single channel | Logic correctness |
| | Control flow | Control graphs with assertions | Single channel | Path traversals |

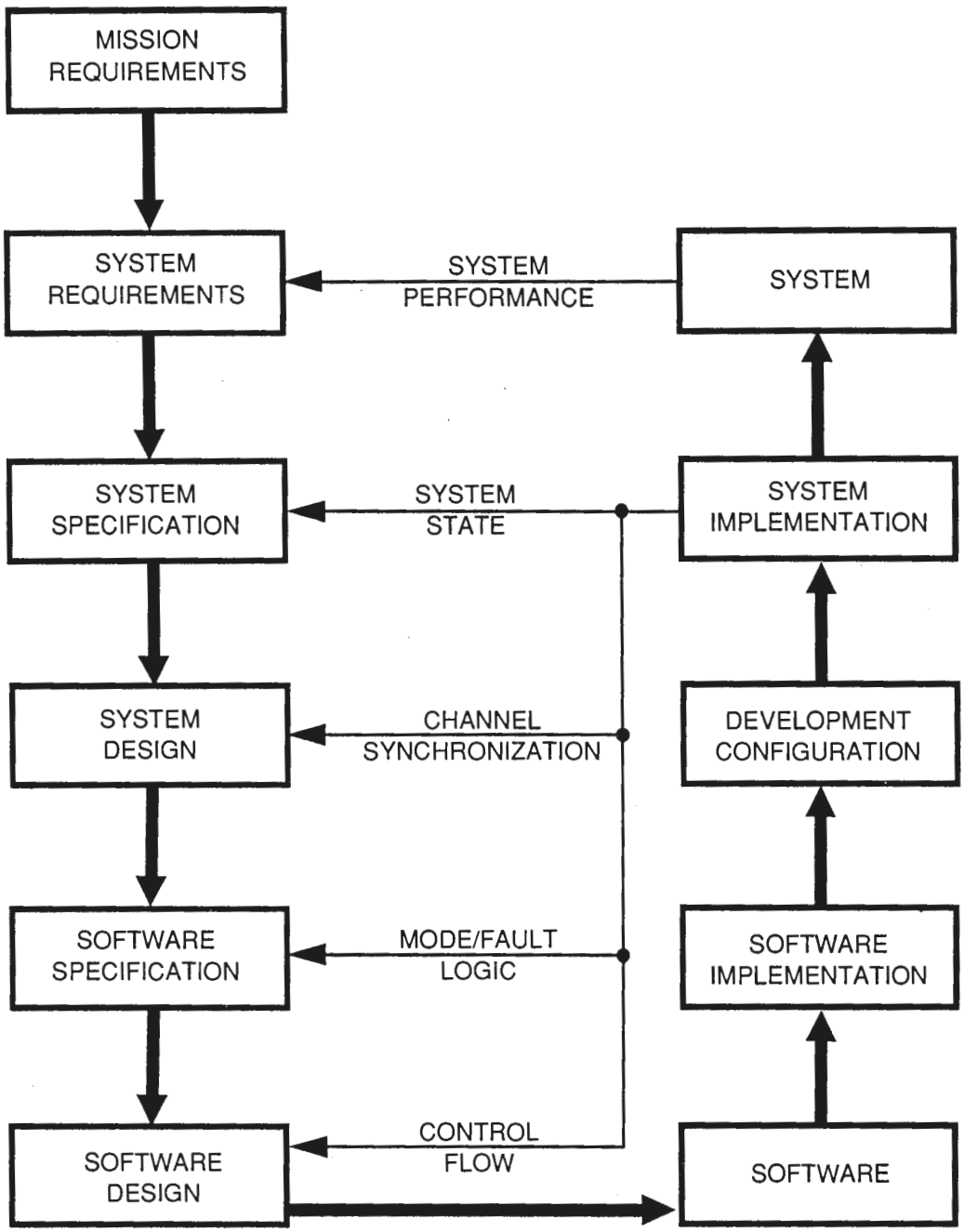



FIGURE 3-2. MULTILEVEL-TESTING CLOSURE

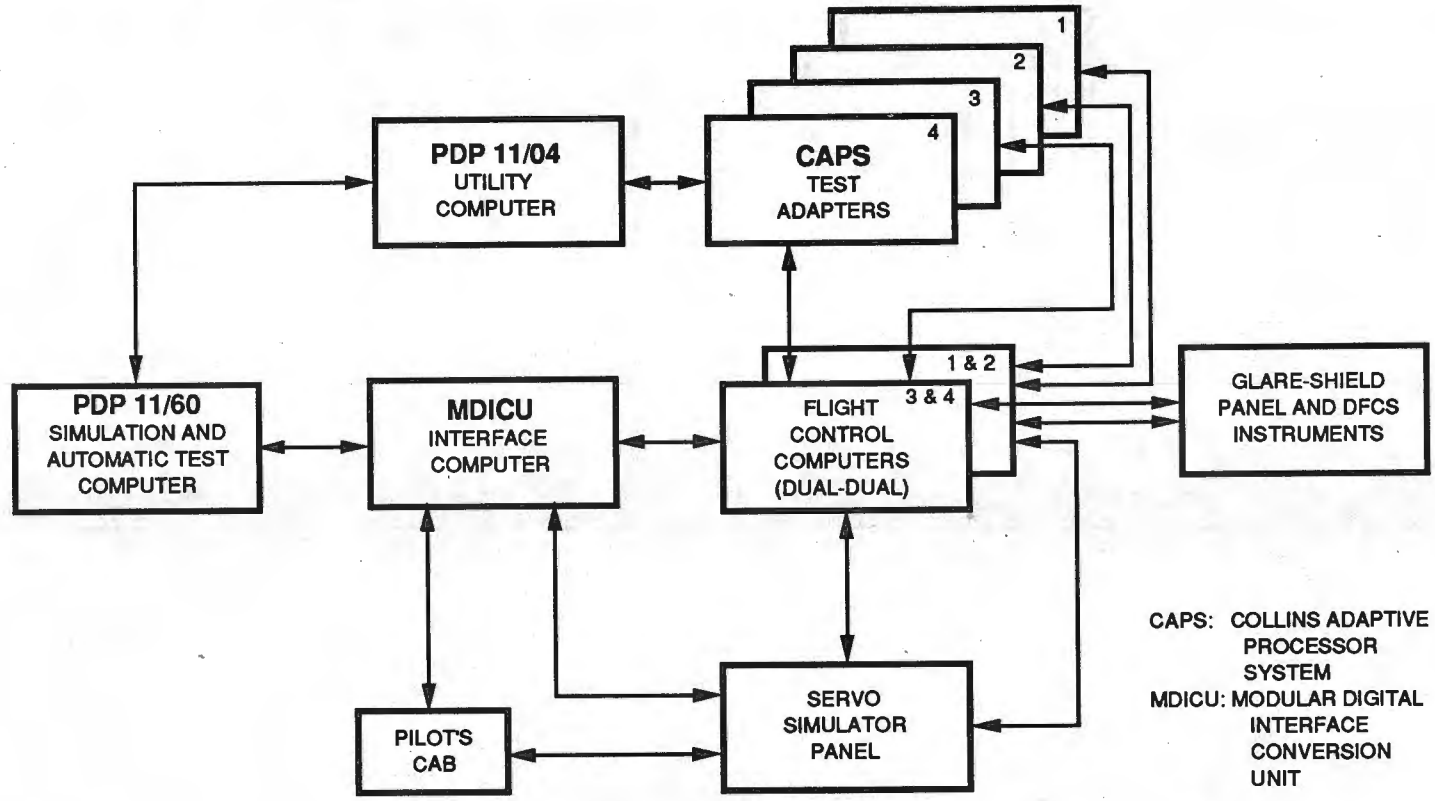



FIGURE 3-3. RECONFIGURABLE DIGITAL FLIGHT CONTROL SYSTEM ALL-UP SIMULATION COMPONENTS

TABLE 3-2. RECONFIGURABLE DIGITAL FLIGHT CONTROL SYSTEM FACILITY CAPABILITIES

| Feature | Significance/Elaboration |
|---|---|
| Language for FCCs and MDICU<br><br>Augmented higher order | Comprehensible, well composed software for DFCS or test purposes<br>Instruction set augmented for machine-related use<br>Mature support software |
| Efficient airframe simulation | 75 percent quiescent frame time at 50 Hz<br>Ample time for test software execution |
| Automated fault insertion | Simulated sensor faults in MDICU<br>Simulated FCC memory faults via PDP-11/04 laboratories<br>FCC hardware fault insertion via Draper |
| Software instrumentation | PDP-11/60 programmed data retrieval via the PDP-11/04 and Collins Adaptive Processor System (CAPS) test retrieval<br>Background mode FCC data retrieval<br>Data object access to all FCCs via link loader options |

# 4. THE USE OF SIMULATION AS AN ASSURANCE TOOL

Engineering-level systems components and a piloted simulation constitute a central and critical part of the DFCS development process. They are used during the verification and validation process of systems development. The work described here is intended to further demonstrate a more effective use of system simulation during the assurance process. For both piloted flight , and system simulators the importance of the use of actual system elements in providing the required fidelity cannot be overemphasized, particularly for the m, re complex digital systems. The final assessment of the performance of the control-system handling qualities, degraded systems operation, electronic displays, and side-arm controllers can only be made with a piloted simulator. In digital systems, performance can be seriously compromised as a result of the inherent phase lags that are induced into the system.

## 4.1. Assurance Assessment

Throughout the development of a product, assurance features must be designed to accommodate the verification and validation processes (Goldberg, 1979 and Wensley, 1969). As the complexity of the system increases, particularly in fault-tolerant designs, early planning for accountability is of utmost importance to guarantee the conclusiveness of the assurance process.

## 4.2. Digital Flight Control System Assurance Methodology

The quadruplex DFCS described by Mulcare, Downing, and Smith, 1987, was developed primarily by making software changes to a dual-dual DFCS flight-test pallet. Although constraints were imposed in the changeover from a dual-dual to a quadruplex setup, the resulting configuration was adequately representative of a conventional quadruplex system. Since the modification was originally developed to demonstrate rigorous system and software design methodologies (Mulcare, Ness, and Davis, 1984) the facility is well-suited for use as an assurance assessment demonstrator.

## 4.3. Simulation Development

Within the constraints of the existing dual-dual FCS, a representative fail-operational-squared (fail-operational/fail-operational) architecture has been defined. The redundancy management coordination of this design can be verified through predicate/transition network simulation, and the high-level software design is then represented in nested control graphs. Cross-channel coordination is effected through respective channel control states broadcast over serial digital buses. The system is then tested under various simulated faulted and abnormal conditions using real-time software execution monitors to resolve low-level system management events.

In the development of the simulator, RSS flight cases are used and validated, using airplane time-histories. New sensor outputs are routed through the Modular Digital Interface Conversion Unit (MDICU) to the flight computers. New mode and fault logic signals are assigned on the logic discrete switch panel on the FCS simulator. Lastly, the pitch-axis inputs are introduced into the flight computer by means of a manual control stick. It is therefore possible to assess flight-qualities degradation through real-time closed-loop systems simulation with or without pilot input.

FCS definition evolved in accordance with the development products shown in figure 2-3, which highlights the stages of mechanization along the downward path on the left of the figure and the assurance steps along the upward path on the right of the figure. Further details of the development activities are presented in table 4.3-1. In general, the objective is to highlight key steps corresponding to progressive system definitions that should lead to a certifiable DFCS. Design definition usually addresses either system functions or architecture. System-function definition centers on control laws, whereas system-architecture definition centers on system and software structure. It is particularly vital that the organization of the flight software be explicitly described prior to digital implementation. These aspects of system definition, particularly system/software architecture, are central to reliable/fault-tolerant DFCSs and will be further discussed before the presentation of application examples (see paragraph 4.6). These design tasks, noted in figure 3-1 are illustrated through a sequence of examples.

To assist in the design definition of a flight-critical FCS, an RSS airplane with a Fly-By-Wire (FBW) FCS is defined. The control law shown in figure 4.3-1 has been used in each of the four redundant FCS channels. Pilot command is through a control stick. Short-period damping is provided from the simulated airplane, and angle of attack (AOA) is used to control the pitch divergence that is associated with the relaxed static stability. Control-surface scheduling is used, because control-surface effectiveness increases with airspeed. Six point-mass simulation cases were used, as shown in table 4.3-2, with their sensor feedback gains.

## 4.4. Basic System Architecture

Channel redundancy for fault tolerance in DFCSs demands intricate hardware/software logic to maintain cross-channel coordination. Such wide bandwidth processes must take place dependably despite component tolerances, inaccuracies in timing, occasional software errors, and hardware faults. At risk are the integrity and continued proper operation of the total system. It is vital to confirm the safety and acceptability of the cross-channel logic under a broad range of deviant or discrepant conditions.

In the latter stages of system development, these concerns can be addressed using the fidelity of an all-up, real-time FCS simulator. However, such simulators do not usually incorporate the automated test capability that is necessary to permit checks at other than the system level. Real-time test programs are desirable; they run in parallel with redundant DFCSs to ascertain the correct cross-channel logic operation, thereby guaranteeing a high degree of resolution and certainty.

TABLE 4.3-1. DEVELOPMENT PHASE ACTIVITIES AND PRODUCTS

| | |
|---|---|
| **1. Conceptual Phase** | |
| Purpose | Provide for user needs |
| Input | Mission requirements |
| Mech.* | Formulate system requirements; explore design solutions |
| Assur.* | Validate requirements; analyze design solutions |
| Product | System requirements and concepts |
| Status | Functional architecture; feasible design solution |
| **2. Definition Phase** | |
| Purpose | Design for user needs |
| Input | Airworthiness and system requirements; system concepts |
| Mech.* | Formulate system specs; refine system concepts |
| Assur.* | Verify specs; validate system concepts |
| Product | System specs; conceptual system design |
| Status | Validated conceptual design; acceptable design solution |
| **3. Analysis Phase** | |
| Purpose | Design system to specs |
| Input | System specs |
| Mech.* | Design system structure; design control laws |
| Assur.* | Verify software specs, system structure, and control laws |
| Product | System design; software and hardware specs |
| Status | Verified system structure; superior design solution |
| **4. Design Phase** | |
| Purpose | Design software to specs |
| Input | Software specs |
| Mech.* | Design software structure; define software components |
| Assur.* | Verify software design and unit specs |
| Product | Software design; program unit specs |
| Status | Verified baseline design; comprehensive design definition |
| **5. Coding and Checkout Phase** | |
| Purpose | Implement software to specs |
| Input | Unit specs |
| Mech.* | Implement program units |
| Assur.* | Check and debug units; verify program units |
| Product | Software implementation |
| Status | Baseline system config.; comprehensive system definition |

*NOTES: Mechanization process; Assurance process.

TABLE 4.3-1. DEVELOPMENT PHASE ACTIVITIES AND PRODUCTS (Continued)

| 6. Integration Phase | |
|---|---|
| Purpose | Construct system with hardware/software components |
| Input | Verified structure and components |
| Mech.* | Assemble/develop system |
| Assur.* | Identify and rectify inconsistencies |
| Product | System implementation |
| Status | Developmental system |
| **7. Development Test Phase** | |
| Purpose | Test to spec requirements |
| Input | System specs |
| Mech.* | Develop system; optimize performance |
| Assur.* | Identify and rectify deficiencies; verify performance |
| Product | Verified implementation |
| Status | Verified system |
| **8. Validation Phase** | |
| Purpose | Test for compliance with requirements |
| Input | System requirements |
| Mech.* | Modify system if necessary |
| Assur.* | Confirm acceptability |
| Product | Provisional configuration |
| Status | Validated system |
| **9. Certification** | |
| Purpose | Demonstrate airworthiness compliance |
| Input | Airworthiness requirements and certification plan |
| Mech.* | Modify system if necessary |
| Assur.* | Confirm airworthiness |
| Product | Production configuration |
| Status | Certificated system |

*NOTES: Mechanization process; Assurance process.

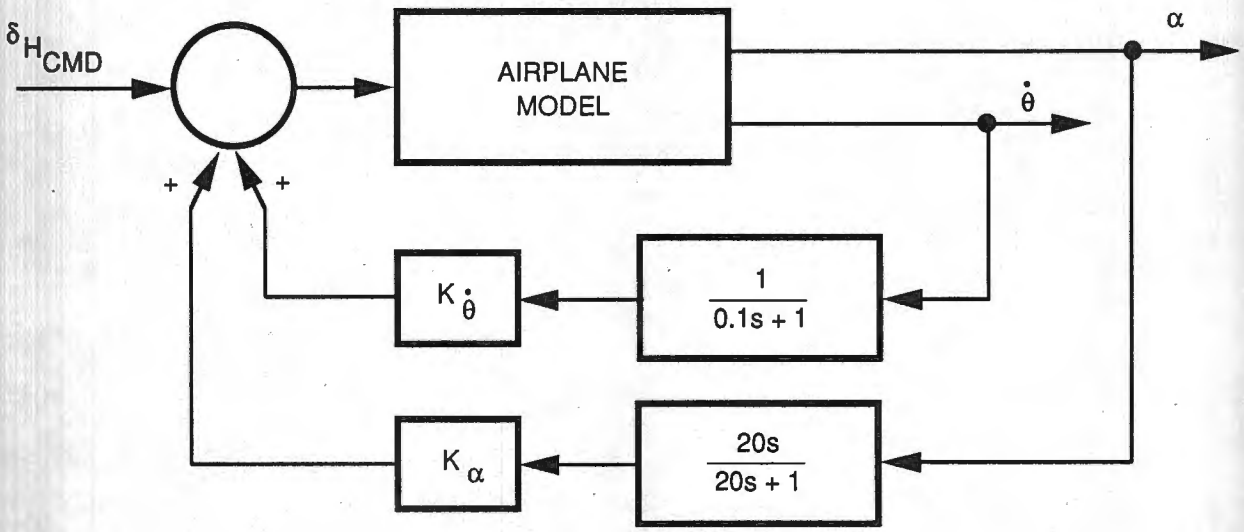



FIGURE 4.3-1. CONTROL LAW USED IN REDUNDANT FLIGHT CONTROL SYSTEM CHANNELS

TABLE 4.3-2. STABILITY AUGMENTATION GAIN SCHEDULING

| Flight Case | True Airspeed | $K_\alpha$, deg/deg | $K_{\dot{\theta}}$, deg/deg/sec |
|---|---|---|---|
| A1RSS | 283.7 | 1.00 | 0.70 |
| C13RSS | 910.7 | 0.73 | 0.10 |
| C15RSS | 442.2 | 1.00 | 0.40 |
| D2RSS | 487.1 | 1.00 | 0.33 |
| E3RSS | 265.7 | 1.00 | 0.80 |
| F6RSS | 224.1 | 1.00 | 0.50 |

## 4.5. Developing the Flight Control System

The top-level primary FCS requirements are assumed to be MIL-F-9490D operational states (Specification MIL-F-9490D). These requirements encompass both airplane flying qualities and system safety in terms of redundancy margins. Operation State 1 denotes normal system status and Level 1 flying qualities (good); State 2 allows some deterioration in safety margins and Level 2 flying qualities (somewhat degraded); State 3 indicates marginal safety and Level 3 flying qualities (marginal); and State 4 designates unsafe components or flying qualities (worse). Knowledge of flying-qualities degradation under conditions of successive component failures makes it possible to view the operational state logic from an architectural point of view. The system design task then focuses on defining the channel-level logic to effectuate the system-level logic.

Figure 4.5-1 represents the top-level DFCS channel logic design; the labels in the circles correspond to the particular states that can be assumed by the logic variables. These logic variables denote states or substates of the channel. The names appearing on the arcs of the state transition graphs designate independent logic events such as pilot-mode selection, timer interrupt, or component failure. Such events in turn may effect changes in the channel states which must reflect the state of the system. Note that the transition graphs are nested to reduce complexity; the lower-level graphs capture substate information.

In the Gate substate of the Cycling state, for example, each channel develops its view of system status based on control state information for each of the four channels. This kind of consensus is one of the major determinants in calculating the operational state of the system. In all, the implementation of this design necessitates a substantial expansion of the logic in the flight code. First, however, it is necessary to examine the system-level coordination logic in terms of meeting the timing constraints given in table 4.5-1.

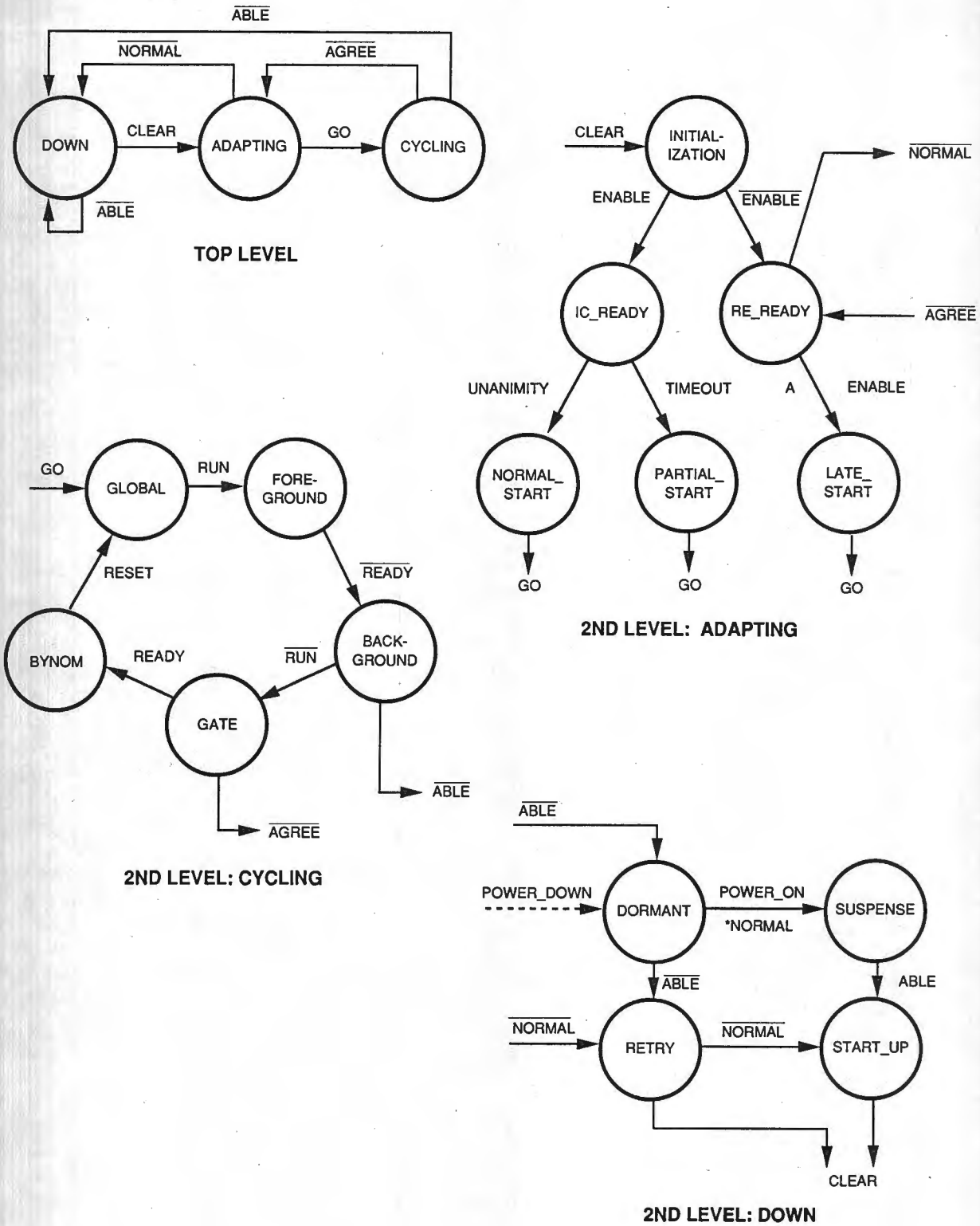



FIGURE 4.5-1. TOP-LEVEL DIGITAL FLIGHT CONTROL SYSTEM CHANNEL LOGIC DESIGN

TABLE 4.5-1. CYCLING STATE LOOP TIMING

| EVENT | Substate and Duration | | | | |
|---|---|---|---|---|---|
| | Global (0-1 msec) | Foreground (1-tfg) | Background (tfg-48) | Gate (48-49) | Synch (49-50) |
| Run | False | True | True | False | False |
| Ready | True | True | False | False | True |
| Reset | True | False | False | False | False |

NOTES: Computational frame time = 50 msec; tfg <48 msec (where tfg is foreground time); timer interrupt at t = 48 msec ≥run; reset ≥t = 0.

System synchronization logic with a discrete time-base imposed is shown in figure 4.5-2, which represents a predicate/transition network (Goldberg, 1979). This view represents only one channel, but all four channels are the same for the subject architecture. Here the same logic nomenclature is retained where applicable, and some new logic variables are added. The logic names appear in the rounded boxes, which are called places in the network. At any given time, the collective values of all the places constitute the state of the system modeled. To examine and verify the correctness of all possible state sequences, the network's operation must be simulated using a computer program. Properly accomplished, such a simulation, under both faulted and fault-free conditions, verifies the system logic design.

The simulation is based on the firing of transitions (denoted by the rectangles in figure 4.5-2), which yield new values for the logic variables stored in the places. The top half of each transition box designates a predicate whose satisfaction by system logic values enables it to be fired. Whenever a transition fires, the new logic values described in the bottom half of the rectangle are assigned. These values are then reflected in the appropriate places, and a new network state produces a new set of transitions that can be fired.

This mechanism can be seen more clearly by noting the partial network in figure 4.5-3. Here the network captures a design where the hardware initialization within a channel sets its POWER ON to TRUE and its CLEAR to FALSE. This arms the top transition, whose firing corresponds to software initialization that sets CLEAR to TRUE among other logic-variable assignments shown. Referring to figure 4.5-3, note that the event of CLEAR being set to TRUE effects the top-level channel state transition from DOWN to ADAPTING. At this point, the channel tries to synchronize with the other channels, to enter the CYCLING state.

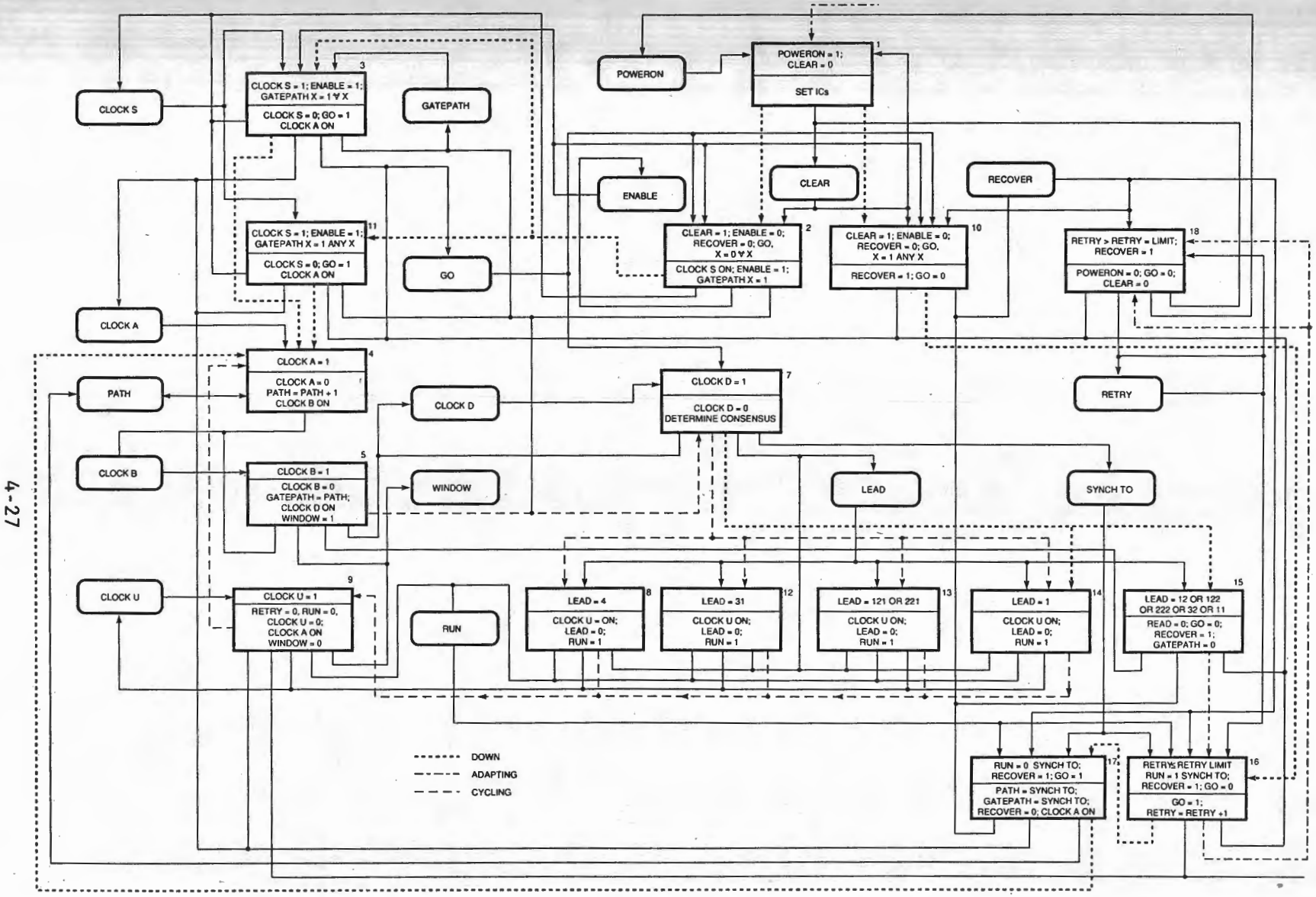



FIGURE 4.5-2. SYNCHRONIZATION PREDICATE/TRANSITION NETWORK

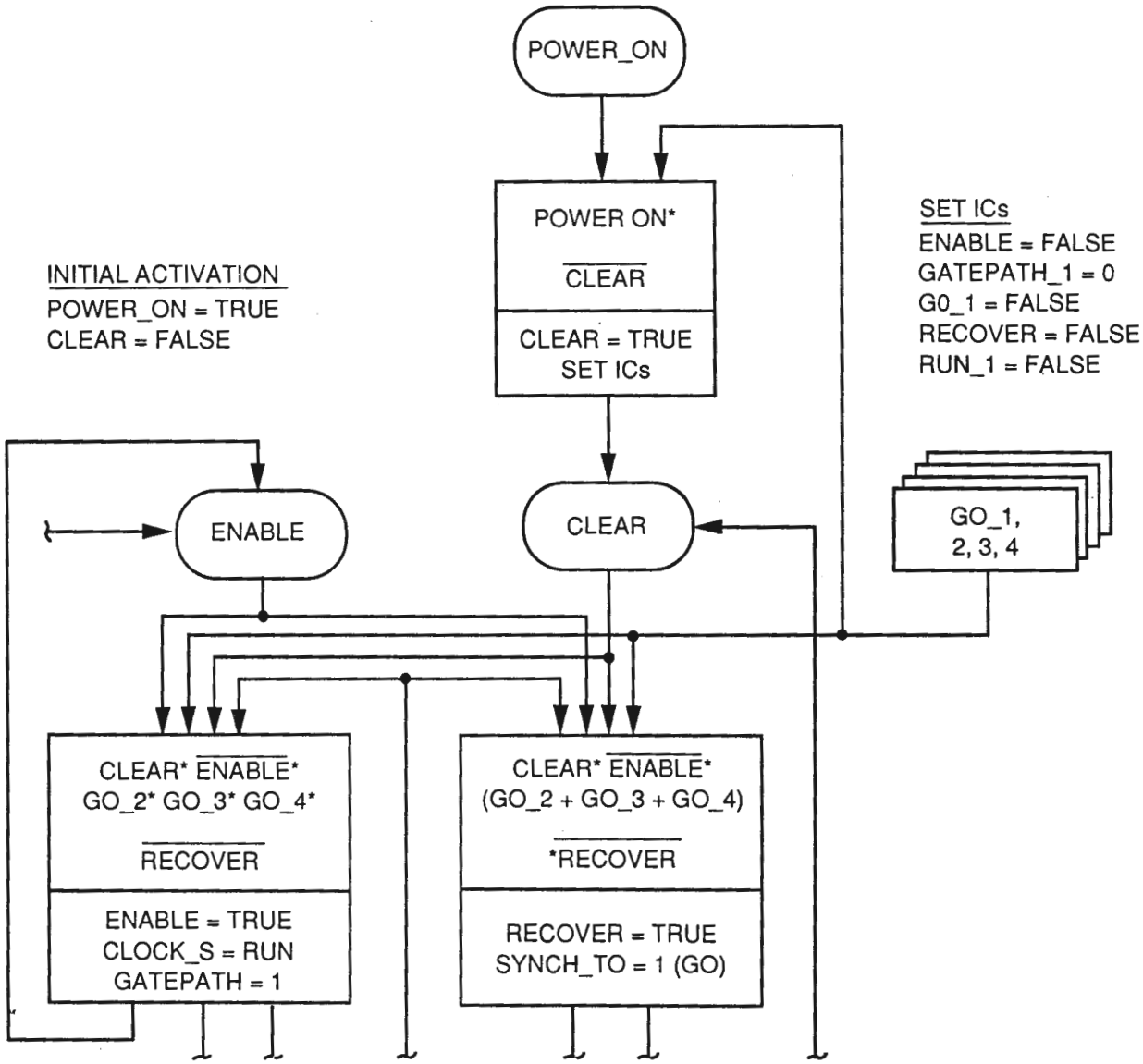



FIGURE 4.5-3. PARTIAL PREDICATE/TRANSITION NETWORK

Network simulation should always yield acceptable system states, and should never terminate unless all channels are failed. Determining that this is the case is a matter of defining and obtaining correct logic operation. In the demonstration, this was accomplished (with one exception noted later). Figure 4.5-4 illustrates some typical discrete-event simulation results for four-way channel synchronization. The pulses at the top indicate instances where individual channels were forced out of synchronization; that is, the corresponding logic-variable RECOVER went TRUE. The resynchronization logic in the network model then satisfactorily restored synchronization, and the RECOVER was set to FALSE, as indicated by the end of the pulse in figure 4.5-4.

Following synchronization design verification, which captured time-based hardware/software interaction, the design emphasis shifted to the top-level software design with the constraints imposed by the existing RDFCS hardware.

In the basic system, the flight software is hardware-interrupt-driven at the 60-Hz rate. Since the AFBW control laws were designed for 20-Hz operation, the executive software invokes the applications control functions every third interrupt. The executive program itself is somewhat austere, as appropriate for a dedicated system. Here the computational channels mutually coordinate themselves in a frame-synchronous, double fail-operational manner.

To achieve this, the channel design must be mapped into a software design with the objectives of maintaining consistency in the system-to-software design transition and minimizing the complexity of the software. Accordingly, the control graph shown in figure 4.5-5 represents a mapping of the top-level transition graph in figure 4.5-1 to a software control structure that preserves the design logic in a form exhibiting only moderate decision-logic complexity as measured by the cyclomatic complexity number. (This number is a measure of program complexity, thus a measure of program testability, maintainability, and, by implication, reliability. The number is a function of the number of control graph vertices, edges, and connected components. See McCabe, 1976 for details.) As byproducts, the control graph yielded test-case input vectors and logic assertions that were later used in verifying the implemented code.

## 4.6. Establishing System Reliability

To establish system reliability and the required architecture, a tradeoff study must be conducted. In establishing system reliability, the reliability assessments are directed toward contrasting levels and types of redundancy relative to their effect on system reliability for critical functions. Table 4.6-1 delineates 12 different system architectures for critical system function failure on both 5- and 10-hour missions. The system architecture shown in figure 2-1 actually corresponds to cases 1 and 2 in table 4.6-1, depending on whether inherent backup capability is invocable. Because of the arrangement of the mode-selection switch that was devised in the RDFCS laboratory, the backup mode was manually selectable. Figure 4.6-1 represents the reliability model for architecture case 1, the one actually implemented. Note that the dependencies and logic embedded in this reliability model do not, however, apply to case 2 in table 4.6-1.

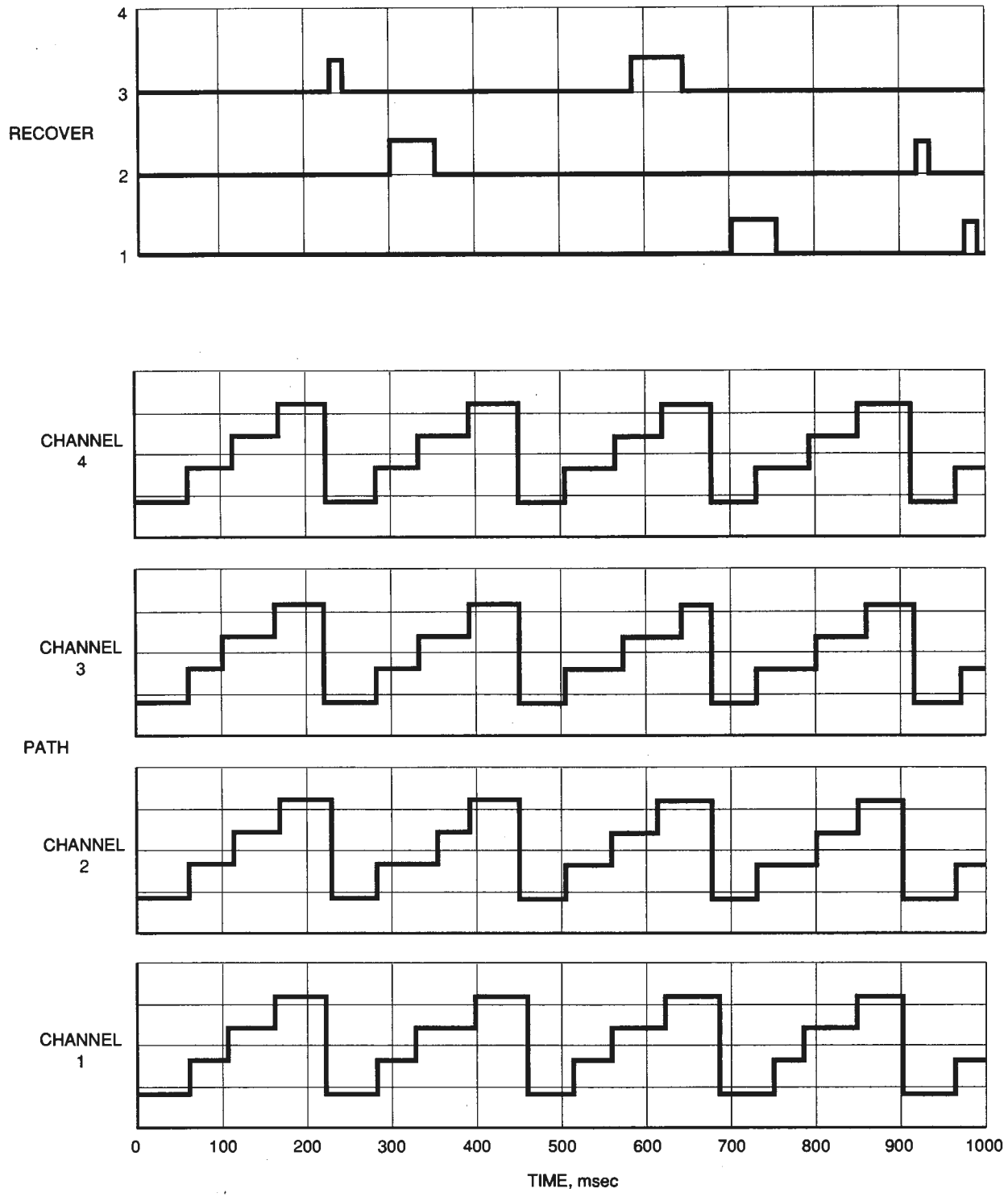



FIGURE 4.5-4. TYPICAL DISCRETE-EVENT SIMULATION RESULTS

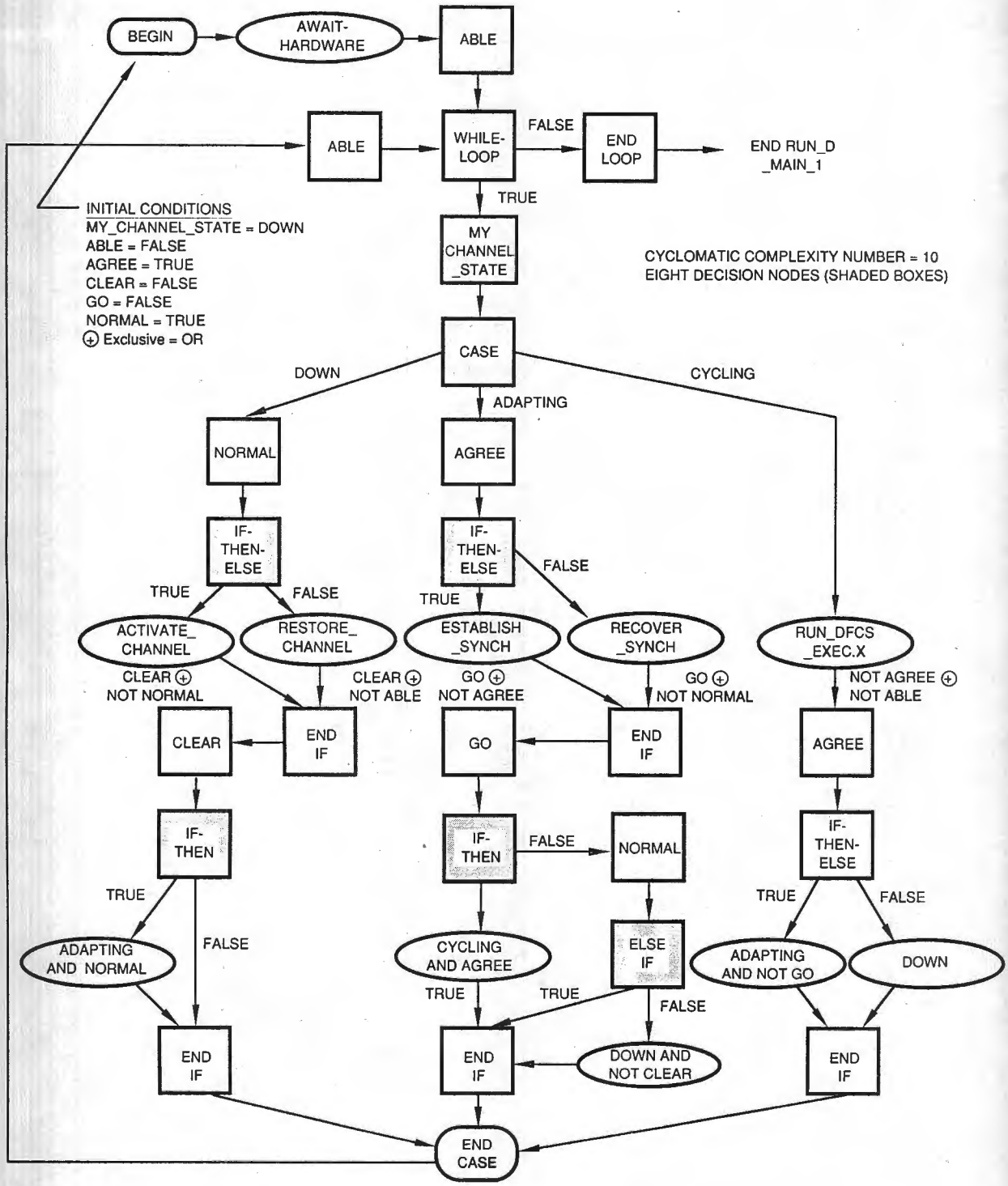
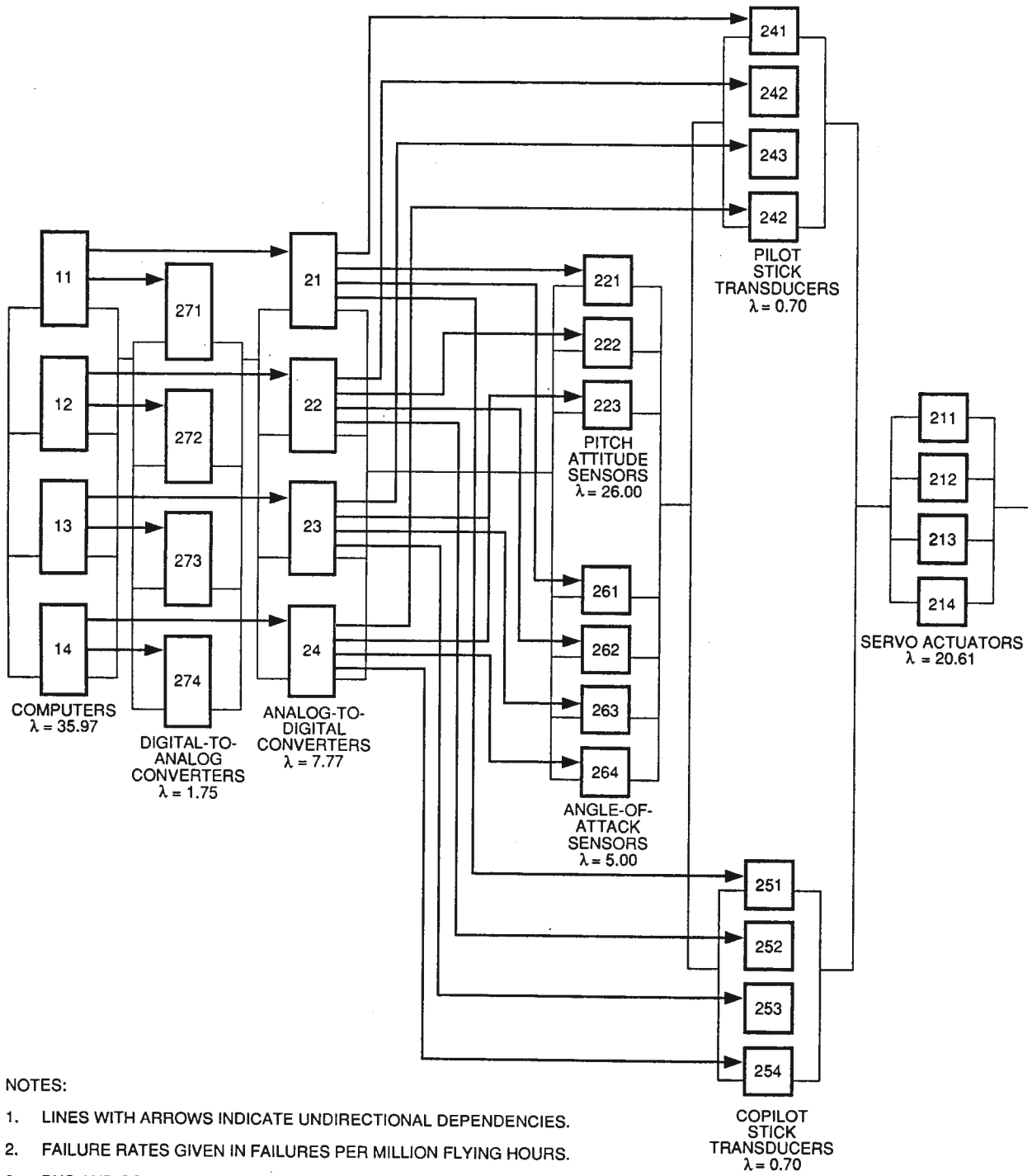



FIGURE 4.5-5. CONTROL GRAPH OF TOP-LEVEL SOFTWARE




NOTES:

1. LINES WITH ARROWS INDICATE UNDIRECTIONAL DEPENDENCIES.
2. FAILURE RATES GIVEN IN FAILURES PER MILLION FLYING HOURS.
3. BUS AND CONNECTOR FAILURES INCLUDED.

FIGURE 4.6-1. RELIABILITY MODEL FOR IMPLEMENTED ARCHITECTURE

TABLE 4.6-1. DIGITAL FLIGHT-CONTROL SYSTEM RELIABILITY ASSESSMENTS

| Case | System | SAS Backup | Servo Setup | Failure Probability | |
|---|---|---|---|---|---|
| | | | | 5 hour | 10 hour |
| 1 | Quad AFBW | Pitch hold | Quad | $0.478X10^{-10}$ | $0.382X10^{-9}$ |
| 2 | Quad AFBW | None | Quad | $0.615X10^{-10}$ | $0.492X10^{-9}$ |
| 3 | Quad AFBW | None | Triplex | $0.626X10^{-10}$ | $0.500X10^{-9}$ |
| 4 | Quad AFBW | None | Dual-dual | $0.107X10^{-7}$ | $0.430X10^{-7}$ |
| 5 | Dual-dual AFBW | None | Dual-dual | $0.200X10^{-6}$ | $0.798X10^{-6}$ |
| 6 | Quad FBW | N/A | Quad | $0.457X10^{-10}$ | $0.365X10^{-9}$ |
| 7 | Quad FBW | N/A | Triplex | $0.468X10^{-10}$ | $0.374X10^{-9}$ |
| 8 | Triplex FBW | N/A | Triplex | $0.153X10^{-6}$ | $0.612X10^{-6}$ |
| 9 | Quad SAS | Pitch hold | Quad | $0.478X10^{-10}$ | $0.382X10^{-9}$ |
| 10 | Quad SAS | None | Quad | $0.615X10^{-10}$ | $0.491X10^{-9}$ |
| 11 | Quad SAS | None | Triplex | $0.626X10^{-10}$ | $0.500X10^{-9}$ |
| 12 | Triplex SAS | None | Triplex | $0.188X10^{-6}$ | $0.751X10^{-6}$ |

The cases of primary interest here (1 through 5 in table 4.6-1) are all-up AFBW architectures. Cases 1 through 3 meet the critical-function reliability requirements defined in Defeo, Doane, and Saito, 1982, but the analyses only take into account the hardware fault contributions to unreliability. Still, the data are instructive in several ways. Basically, the backup pitch-hold mode adds surprisingly little to survivability, partly because it would be needed so infrequently, and then only at a time when it too might well be unavailable owing to the loss of common components with the basic pitch Stability Augmentation System (SAS).

Cases 4 and 5 (table 4.6-1) are inadequate because of reduced redundancy levels. Cases 5 through 7 isolate the reliability properties of straight FBW architectures. A comparison of cases 2 and 6 shows, for example, that the critical pitch SAS function increases the probability of failure by about only a third for a straight FBW system. The straight SAS function is isolated in cases 9 through 12, and it is noteworthy that triplex AOA sensors and computers are inadequate for case 12. However, triplex servos with quadruplex sensors are satisfactory.

Tables 4.6-2 and 4.6-3 reveal flying-qualities degradation, as well as system failure, the extent and likelihood of the degradation being important measures of system acceptability. Cases 6 through 8 are omitted because the straight FBW function does not degrade in stages; instead, it usually fails completely when an appropriate combination of failures has occurred. Since stability-augmentation degradation is basically sensor-related, this set of servo-oriented architecture variations is not fully useful in delineating flying-qualities tradeoffs.

TABLE 4.6-2. DEGRADATION OF FLYING-QUALITIES: 5-HOUR MISSION

| Case* | Flying-Qualities Level (probability) | | | |
|---|---|---|---|---|
| | 1 | 2 | 3 | <3 |
| 1 | 0.999+ | $0.212X10^{-6}$ | $0.017X10^{-11}$ | $0.478X10^{-10}$ |
| 2 | 0.999+ | $0.212X10^{-6}$ | 0 | $0.615X10^{-10}$ |
| 3 | 0.999+ | $0.212X10^{-6}$ | 0 | $0.626X10^{-10}$ |
| 4 | 0.999+ | $0.212X10^{-6}$ | 0 | $0.607X10^{-10}$ |
| 5 | 0.999+ | $0.120X10^{-3}$ | 0 | $0.200X10^{-6}$ |
| 9 | 0.999+ | $0.212X10^{-6}$ | $0.137X10^{-10}$ | $0.478X10^{-10}$ |
| 10 | 0.999+ | $0.212X10^{-3}$ | 0 | $0.615X10^{-10}$ |
| 11 | 0.999+ | $0.212X10^{-6}$ | 0 | $0.626X10^{-10}$ |
| 12 | 0.999+ | $0.221X10^{-6}$ | 0 | $0.188X10^{-6}$ |

*Cases 6 through 8 omitted because the FBW function does not degrade in stages (it fails completely given appropriate combinations of failures).

TABLE 4.6-3. DEGRADATION OF FLYING-QUALITIES: 10-HOUR MISSION

| Case* | Flying-Qualities Level (probability) | | | |
|---|---|---|---|---|
| | 1 | 2 | 3 | <3 |
| 1 | 0.999+ | $0.848X10^{-6}$ | $0.250X10^{-14}$ | $0.382X10^{-9}$ |
| 2 | 0.999+ | $0.848X10^{-6}$ | 0 | $0.492X10^{-9}$ |
| 3 | 0.999+ | $0.848X10^{-6}$ | 0 | $0.500X10^{-9}$ |
| 4 | 0.999+ | $0.848X10^{-6}$ | 0 | $0.430X10^{-7}$ |
| 5 | 0.999+ | $0.240X10^{-3}$ | 0 | $0.798X10^{-6}$ |
| 9 | 0.999+ | $0.819X10^{-6}$ | $0.108X10^{-9}$ | $0.382X10^{-9}$ |
| 10 | 0.999+ | $0.848X10^{-6}$ | 0 | $0.492X10^{-9}$ |
| 11 | 0.999+ | $0.848X10^{-6}$ | 0 | $0.500X10^{-9}$ |
| 12 | 0.999+ | $0.883X10^{-6}$ | 0 | $0.751X10^{-6}$ |

*Cases 6 through 8 omitted because the FBW function does not degrade in stages (it fails completely given appropriate combinations of failures).

However, certain factors are of interest. Cases 11 and 12, although they show only a slight increase in flying-qualities degradation for the fully triplex architecture, indicate a notable disposition toward system failure. This suggests that the triplex computers are a weaker point in configuration 12 than are the three AOA pairs. Case 8 (table 4.6-1) tends to reinforce this inference.

The architectures used here are based on earlier-design DFCSs that have been used in retrofitting aircraft that had previously used analog systems. In retrofitted systems the interconnecting wiring is usually dedicated point-to-point signal paths, with little use of Multiplexer (MUX) buses. Newer systems commonly use parallel MUX buses, and considerably more complex bus topologies will be used when true digital computational capability comes into common use. Newer systems will use broader bandwidths, damage tolerance, and soft-fault processing. As a result, extended Mean Time Between Failures (MTBF) of up to 90,000 hours and system availabilities of 0.95 at a service life of 40,000 hours will be achieved. Considerably more system integration will also lead to additional diversity in MUX bus organization.

## 4.7. The Airplane Simulation

A negative static stability margin was postulated as a requirement for this demonstration, and six RSS flight cases were developed. These flight cases are then analyzed to determine the pitch-axis dynamic behavior, and a non-real-time simulation is developed and checked against the predicted pitch-axis dynamics for the free or, unaugmented, airplane. The same flight-case data are then used in the RDFCS facility simulation. Next, the non-real-time simulation time-history responses are used to check the RDFCS simulation.

Both the non-real-time and the RDFCS simulations discussed in this chapter are implemented using a state-variable approach (figure 4.7-1). Here the pitch-axis dynamics of the free RSS airplane, as captured in the matrix A (figure 4.7-1), are quite divergent or unstable, so certain states must be fed back to enable a stability augmentation function. The state variables here are pitch, pitch rate, vertical-axis velocity, and horizontal-axis velocity. The first two states are inertially oriented and are directly measured by airplane sensors. The second two are referenced to the airstream incident on the airplane, and are combined to form the directly measured signals: AOA and true airspeed. These two air-data signals are produced using matrix C.

The above sensor feedback signals then appear in vector u (figure 4.7-1). Pilot stick inputs are applied through vector d.

Table 4.7-1 summarizes conversions of six basic wide-body transport-type flight cases to six corresponding RSS flight cases with -5 percent static stability margins. For flight case A1RSS, for example, calculations revealed the neutral point to be 53 percent of the Mean Aerodynamic Chord (MAC), where neutrality denotes that there is no pitch-moment change with change in AOA. More specifically, the stability derivative describing the airframe behavior goes to zero. The -5 percent static margin then corresponds to shifting the center of gravity aft to 58 percent MAC.

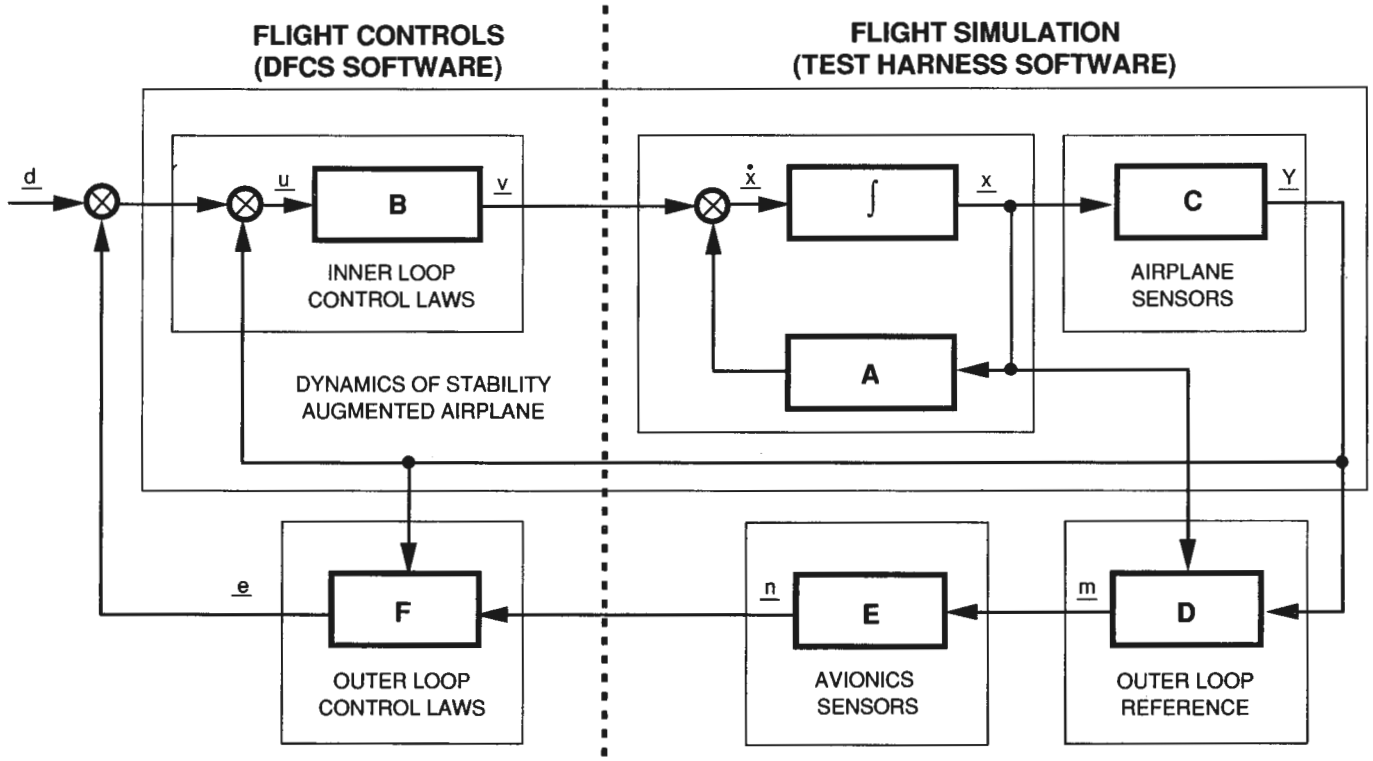



FIGURE 4.7-1. DIAGRAM OF STATE-VARIABLE AIRPLANE SIMULATION

TABLE 4.7-1. RELAXED STATIC STABILITY FLIGHT CASES

| Parameter | A1RSS | C13RSS | C15RSS | D2RSS | F3RSS | F6RSS |
|---|---|---|---|---|---|---|
| c.g., $\%\bar{c}$ | 58.0 | 51.0 | 54.2 | 52.8 | 55.5 | 64.4 |
| Neutral point, $\%\bar{c}$ | 53.0 | 46.0 | 49.2 | 47.8 | 47.8 | 49.4 |
| $V_T$, ft/sec | 283.7 | 910.7 | 442.2 | 487.1 | 265.7 | 224.1 |
| $\tau_1$, sec | 1.099 | 0.160 | 0.122 | 0.660 | 0.938 | 0.920 |
| $\tau_2$, sec | -5.86 | -2.40 | -1.64 | -7.55 | -7.09 | -3.17 |
| $\omega_{PH}$, rad/sec | 0.169 | 0.008 | 0.134 | 0.125 | 0.167 | 0.209 |
| $\zeta_{PH}$ | 0.276 | 0.325 | 0.526 | 0.453 | 0.488 | 0.192 |
| $\Delta C_M/\Delta\delta_H$, 1/rad | -3.247 | -2.830 | -2.415 | -2.533 | -3.138 | -3.016 |
| $\Delta C_M/\Delta_q$, 1/rad | -11.48 | -14.99 | -12.15 | -12.75 | -11.76 | -11.23 |
| $\Delta C_M/\Delta$ , sec/rad | -3.14 | -7.83 | -3.44 | -3.18 | -3.14 | -3.03 |
| $\Delta C_M/\Delta C_L$ | 0.050 | 0.044 | 0.053 | 0.050 | 0.053 | 0.170 |
| $\Delta C_M/\Delta$ , 1/rad | 0.295 | 0.353 | 0.288 | 0.292 | 0.306 | 0.960 |

This kind of alteration to the airframe dynamics yields a pair of real or non-oscillatory roots, $\tau_1$ and $\tau_2$ in table 4.7-1. Note that the negative time-constants for $\tau_2$ correspond to positive roots, which plot on the positive axis of the s-plane and produce an absolute instability or exponential divergence that dominates the dynamic response of the airplane's pitch axis. With a -5 percent margin, moreover, the rate of divergence is quite rapid and clearly unacceptable. For flight case F6RSS, for example, the 3.17 second time-constant yields a doubling of pitch attitude in 2.2 seconds, which is so rapid as to be unflyable.

A non-real-time simulation was used to generate airplane time-history check cases before work was begun at the RDFCS simulator. The organization of the

previously been installed in the simulator test computer. For the RSS flight cases, no changes were made to the software organization itself, other than to add interfaces for the SAS control laws. The organization of the simulation provides for the conversion of flight-case data into a discrete-time model, trimming for specified initial conditions, and generation of dynamic time-history outputs. Ground effects, random gust options, or point simulation flight-case transitions may be selected.

Two stages of analysis were performed. First, the RSS airplane behavior was approximated by shifting the center of gravity aft to -5 percent MAC. It should be noted that six of the old flight cases have been converted into six derivatives sensitive to the reduced lever-arm of the empennage about the new center of gravity. As indicated in table 4.7-1, these all relate to the generation of a pitching moment. The analyses of the RSS flight cases then involved the examination of their respective dynamics. Specifically, the RSS stability derivatives were used to calculate the free-airplane response by finding the root solutions to the characteristic equation for each flight case. These results are also given in table 4.7-1.

Note that the sign on the pitching moment, owing to wing-lift, changes from negative for 25 percent to positive for 58 percent MAC. In the RSS case then, increasing lift produces more nose-up moment, which in turn further increases lift. This positive feedback causes an instability in the free, or unaugmented, airplane response. The rate of divergence is quite rapid. For flight case F6RSS, for example, the time-constant of -3.17 seconds yields a doubling of pitch attitude every 2.2 seconds. The given time-constants, radial frequencies, and damping ratios were obtained from the root solutions of the characteristic equations for the respective flight cases, as determined by the stability derivatives.

Simulation check cases were first run using the non-real-time dynamic simulation. Time-histories were generated for each case and checked against the root solutions. For the RSS cases, the divergent real root dominated the time-histories. The pitch-angle sequence exhibits an exponential time-constant roughly the same as the analytical calculated time-constant value of -3.17 seconds for flight case F6RSS once the initial transient has subsided. The initial upset of 3.0 degrees per second of pitch rate employed here is not really needed to exhibit the instability. It is used as a standard upset as needed for the corresponding augmented airplane.

Whereas these free-airplane responses are used as a cross-check on the real-time airplane simulation, the augmented airplane response is used to check both the simulation and the stability-augmentation control laws. Analytical-root-solution (eigenvalue) checks can also be made for the augmented airplane response, where the order of the characteristic equations is increased as a result of the presence of the sensor feedbacks. All these checks should be in general agreement. They then serve to corroborate the control laws used in different stages of development.

In general, the requirement for stability augmentation is to effect desirable flying qualities for the augmented airplane. Basically, the eigenvalues of the closed-loop airplane should exhibit suitable damping and frequency characteris-

closed-loop airplane should exhibit suitable damping and frequency characteristics. In effect, the augmentation control law should override the RSS airplane's positive pitching moment caused by increasing wing lift. Restoration of the negative pitching moment is possible through a nose-down stabilizer input for increasing AOA.

For the subject demonstration, the design criterion was simply to provide approximately the same, or better, pitch-axis handling and response for the stability-augmented RSS airplane than was available on the basic 25 percent MAC center-of-gravity free airplane.

## 5. VERIFYING THE SIMULATION

There are two major concerns with the verification of DFCSs: cross-channel synchronization and cross-channel consensus. Cross-channel synchronization poses the greater challenge because coincident, wideband access to the required logic signals, either hardware or software, usually demands some special instrumentation. In addition, means must be provided for rapidly appraising the observed logic states so that anomalies can be noted and captured before new events mask the attendant circumstances. Such appraisals must correlate events over the different channels, as well as reference external events such as fault insertions. For closure on assurance objectives, the online assessment must be based on precise representations of the verified design. These then constitute real-time test criteria when incorporated into execution monitors.

Observation and assessment of system availability and function/mode states are more tractable because of relaxed bandwidth requirements. Moreover, monitoring hardware test points is not generally necessary. The overall magnitude and complexity of the associated logic considerations means that a substantial amount of logic must be monitored. It is therefore desirable to minimize the number of logic variables that must be monitored. Perhaps the most formidable problem for the system monitor is that of judiciously defining test cases. The intent here is to maximize system-level coverage while acknowledging the unit-level test cases and coverage of previous programs. This is, of course, an issue largely independent of the construction and operation of the execution monitor itself. Note that in the case of the cross-channel synchronization monitor, previous unit-level testing is not very significant. This is because definitive testing is possible only after the hardware/software is available and operating.

As in most exploratory investigations, the scope of the DFCS test article was purposely limited, even though the quadruplex system architecture logic was fully implemented. Appropriate aspects of this logic are examined by two real-time execution monitors. The channel synchronization monitor for this implementation resides in one of the flight computers, and the system state monitor has been installed in the main simulation/test computer. The first monitor examined certain low-level events. On a larger scale it noted the readiness of each channel to operate with the flight-computer consensus.

Here the term consensus refers to the computer subsystem operational state as manifested by the majority of operable computers in terms of their synchronized, cooperative behavior. Most of the flight-computer faults that can occur compromise the affected computer's capacity to function and its ability to participate in this consensus. Such a loss would be noted by the system state monitor directly, as well as in terms of the DFCS system state consensus. Also, the synchronization monitor will capture some of the constituent logic sequences in the respective DFCS channels that culminate in the system state noted.

An automated testing scenario as generalized in figure 5-1 and elaborated on in table 5-1 has been implemented to simulate, observe, and record DFCS coordination phenomena. The test-case design has resulted from analysis of the DFCS specification and design, within the constraints of the capabilities of the test facility. Those constraints discouraged the use of automated test drivers, such as those used in Benson, Mulcare, and Larsen, 1987. However, this is not a significant limitation in this example, in which the emphasis has been on the real-time monitoring and results processing, instead of on testing productivity.

Multilevel testing is used to obtain conclusive evidence of the consistency of the DFCS implementation. Test-case design focuses on the architectural system/software logic indicated in figure 3-2. Here system validation is not at issue, so verification test scripts for the all-up system are devised only for the multiple levels indicated. Actually, these various level test requirements should be by-products of the corresponding stages of system development. The key point is that low-level design and implementation details should be examined simultaneously during system testing.

The DFCS simulator facility was implemented as shown in figure 5-2. The dual-dual architecture was replaced by a double fail-operational (quadruplex) architecture where each channel is relatively independent of all others. As a consequence, only one computer channel would fail at a time. The utility computer is used to retrieve selected software state data, through the test adapters, for the simulation/test computers.

Simulator testing was mechanized as shown in figure 5-3. The system consensus execution monitor, which was in the simulation/test computer, observed each channel's view of the consensus along with certain lower-level logic states. These parallel sets of data were used to check the consistency of the logic states of the operable channels. These states were referenced to the specified system states, according to component availability, and to lower-level logic design descriptions.

Not surprisingly, the bandwidth of the instrumentation computer was inadequate for monitoring and evaluating cross-channel synchronization logic. The fourth DFCS flight computer was used to monitor the synchronization logic of the other three, a function made possible because of the built-in cross-channel serial data transmission broadcast from each channel. This was also the medium used for synchronization control under the software-only modification rules for the test facility. Consequently, the monitor was able to view the synchronization process without having to fetch the wide bandwidth signals of interest.

Table 5-2 provides a breakdown of the multilevel monitoring functions and their focus. All but the fourth category, control flow, have been implemented and run. Still, there was ample quiescent time in the testing part of the system to conduct concurrent additional tests. The purpose of all of the monitors was to observe logic variable changes and their effects on related states. Figure 4.5-1 illustrates state-transition graphs that define the synchronization design within each channel. The monitor can then be interpreted as a data structure for each channel coupled with the specified outcome taken from the system level.

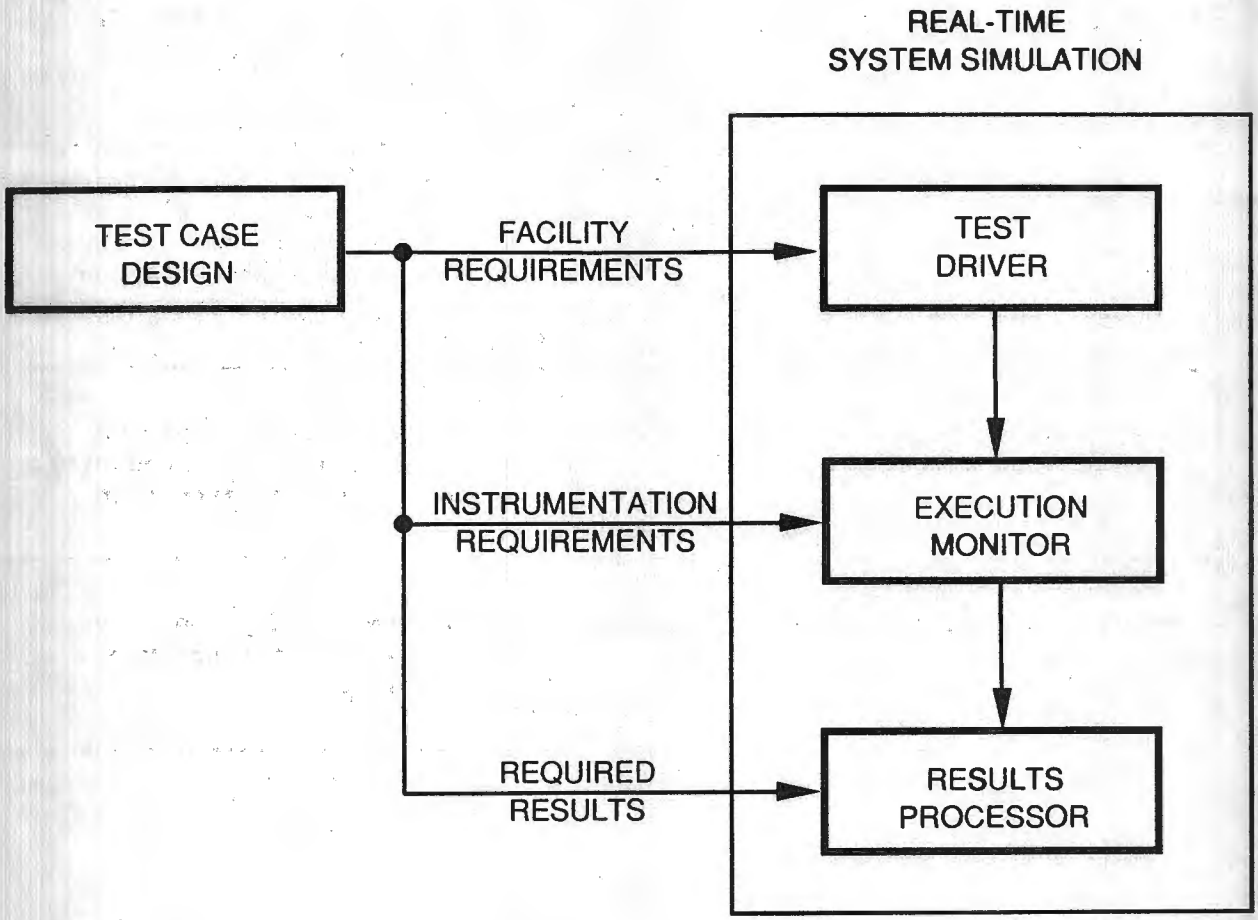



FIGURE 5-1. AUTOMATED TEST SCENARIO

TABLE 5-1. AUTOMATED TESTING TOOLS

| Function | Type of Tool | Purpose |
|---|---|---|
| Test case design | Static analysis | Assessment of properties without actual program execution |
| Test driver | Automated testing | Automated control of test stimuli, environment, and data acquisition in a non-interfering manner |
| Execution monitor | Instrumentation | Implementation of probes for monitoring or recording during program execution |
| Results processor | Dynamic analysis | Assessment of properties during or after program execution |

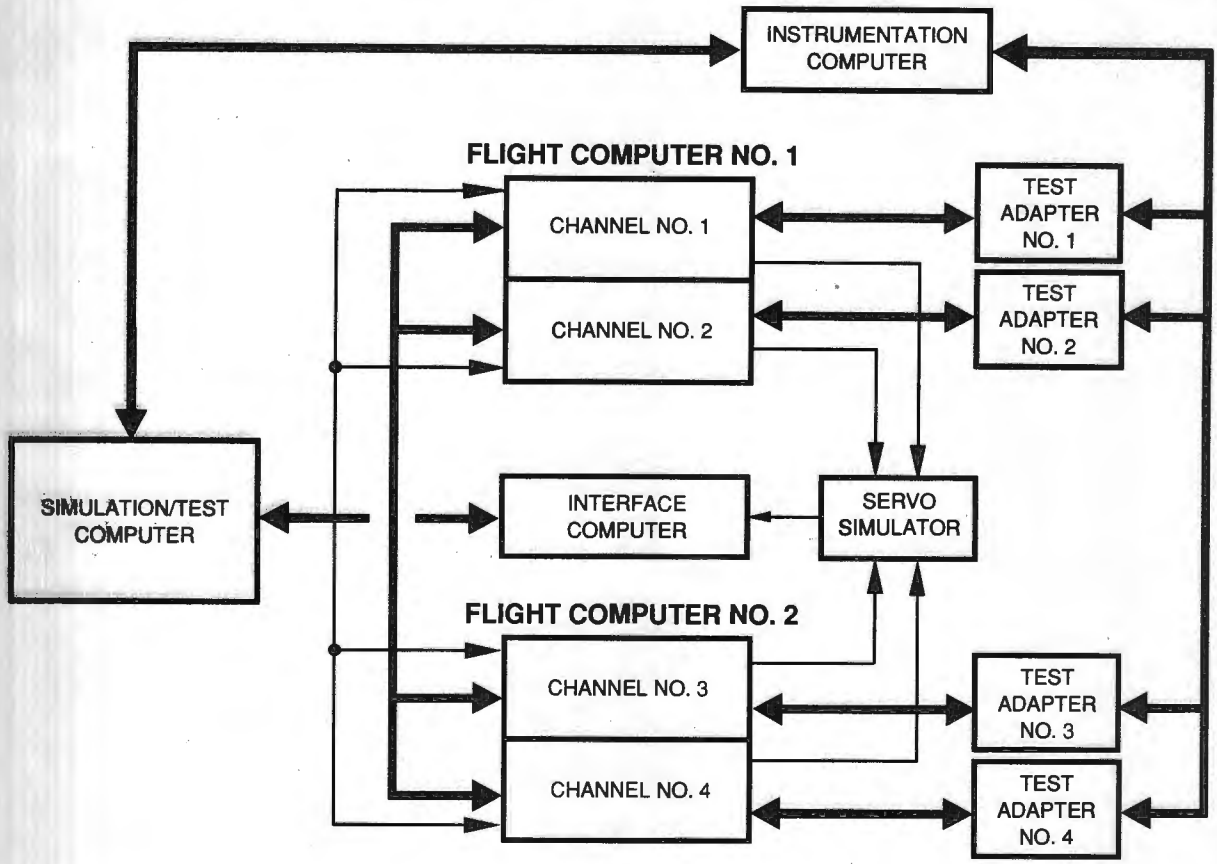



FIGURE 5-2. IMPLEMENTATION OF DIGITAL FLIGHT CONTROL SYSTEM SIMULATOR FACILITY

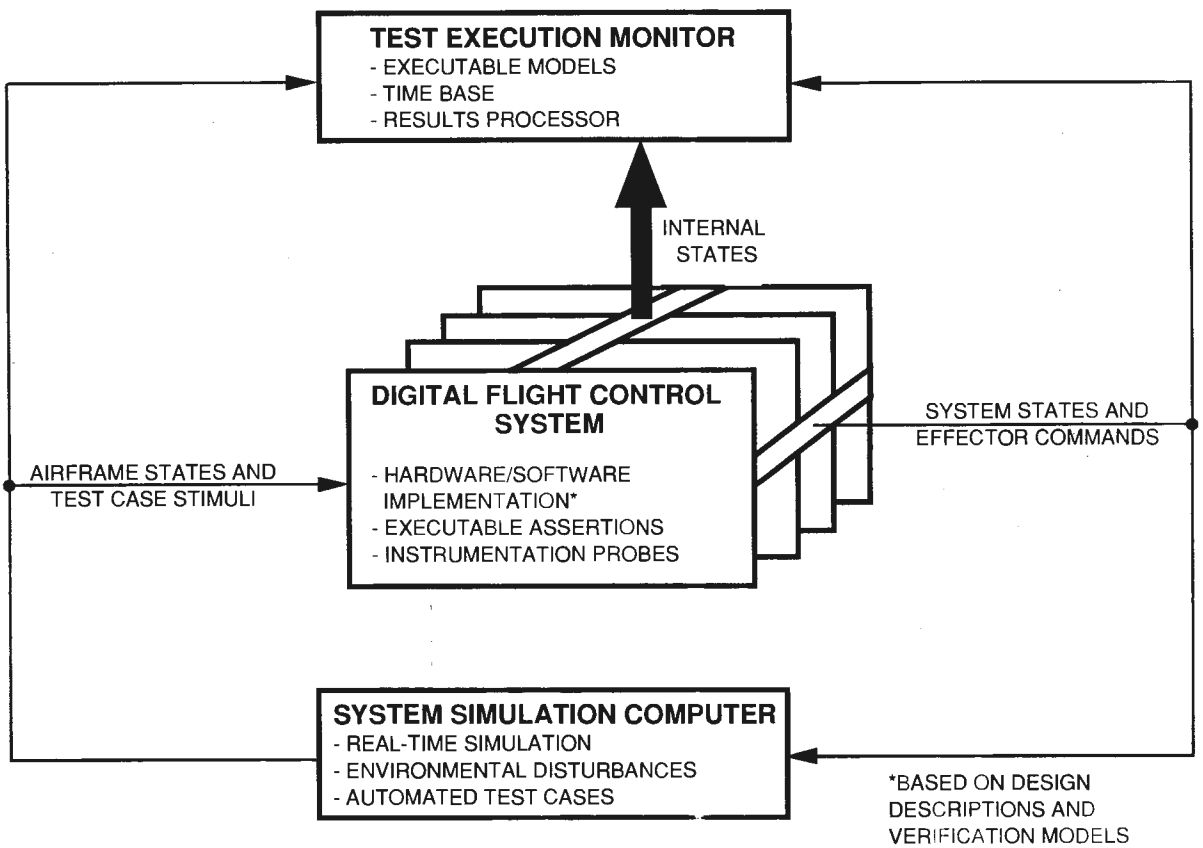



FIGURE 5-3. MECHANIZATION OF SIMULATOR TESTING

TABLE 5-2. MULTILEVEL TEST CASES AND MODES

| Verification Focus | Execution Monitor | Monitor Location | Primary Concern |
|---|---|---|---|
| System State | Nested-state transition graphs | Simulation/test computers | System state consensus |
| Channel synchronization | Nested-state transition graphs | One flight computer channel | Coincidental faults |
| Mode/fault logic | Boolean expressions | Simulator/test computer | Logic correctness |
| Control flow | Control graphs with assertions | One flight computer | Software path traversal |

## 6. SUMMARY

In addition to a variety of simulated hardware fault insertions, the quadruplex DFCS used for this example actually sustained an out-of-tolerance Flight Control Computer (FCC) clock problem during testing. This was at first thought to be a new DFCS design problem, but examination showed that the system was responding properly to discrepant hardware. It is not certain if the frequent, rapid resynchronizations would have otherwise been noticed since the defective channel remained online most of the time. This is one example of the ease with which a discrepancy, which might not be readily apparent at a higher level, can be noted at a low level. It is a case supportive of the subject execution monitor.

A distinction must be made between a physical error and a design error as the source of a discrepancy. As an instance of actually locating a design fault during simulator testing, one of the synchronization test cases that was applied disclosed a flaw in the verified design that was implemented in the DFCS. Specifically, the case involved two channel pairs, each synchronized for clock resetting and path traversal. But each pair continued to run without attempting to synchronize to the other pair. The resultant skew tended to yield some otherwise unwarranted voter/comparator logic trips. This occurred in flight-qualified hardware.

This problem was identified by iterating part of the design process, namely, that of reconstructing the discrepant operation in the predicate/transition network analysis simulation. Once this was done, the logic flaw and the required fix were identified. This illustrates not only the power of the diagnostic information provided by the synchronization execution monitor, but the need for better design verification test-case definition.

All of the other simulator test results merely affirmed (as intended) the correct operation of the DFCS logic as noted, for example, in figure 6-1. Affirmation was sufficient because verification is a process of confirming specified operation under the range of admissible conditions. Such confirmation was thought to have been achieved with a high degree of certainty through automated, multilevel testing, which depended crucially on the real-time execution monitors.

Despite certain inconveniences inherent in the RDFCS facility, the effectiveness of real-time execution monitors was considered excellent. There seems to be no substitute for the fidelity, precision, and conclusiveness provided in verifying fault-tolerance implementation.

As shown here, the execution-monitor testing approach should be supported by rigorous, early design verification as described in Barton, Mulcare, and LeBlanc, October 1985. Essentially, the same monitor criteria and logic model can be used both for design verification in pure simulation and for implementation verification in a system simulator. The interplay of these two assurance

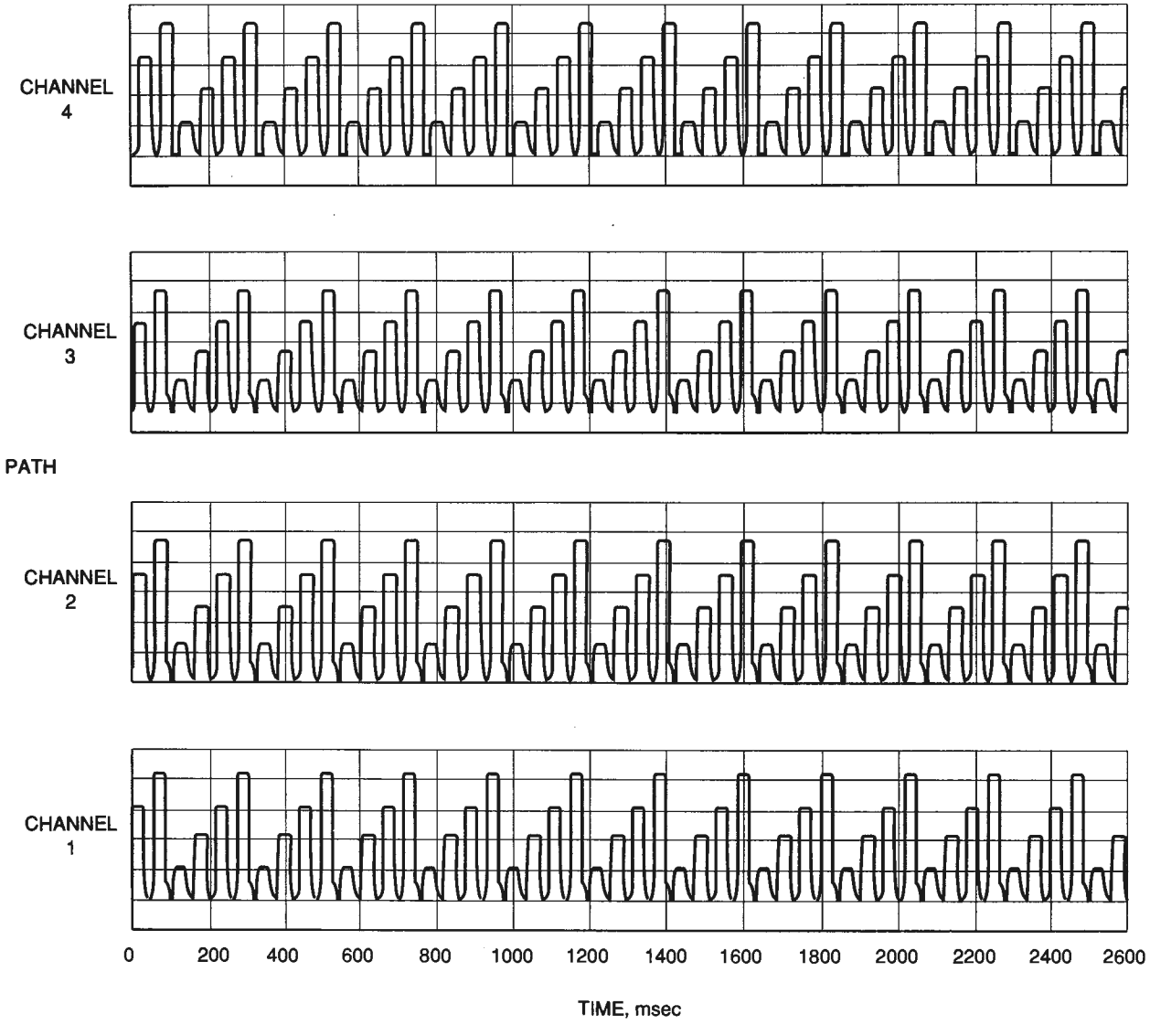



FIGURE 6-1. CROSS-CHANNEL SYNCHRONIZATION TIME-HISTORIES

activities, explicitly connected through the same precise criteria, tends to maximize confidence in the integrity and feasibility of system fault tolerance. This is fortunate, for generally the greatest source of fault-tolerant system unreliability is the fault-tolerance mechanisms themselves (Driscol, 1985).

The system state monitor aids in the interpretation of the synchronization monitor results. It can also be related to the status of DFCS components other than the flight computers. Moreover, the observed logic states can be readily correlated with flight software path traversals in an integrated testing strategy. In the interest of test productivity, comparable low-level monitoring of functions associated with different path traversals could be accomplished simultaneously. Multilevel testing would then be complemented by multifunction testing.

In summary, the demonstration of the execution monitors supports the addition of certain features to system simulators to facilitate the fault-tolerant architecture testing described. Moreover, such changes in the composition and use of system simulators appear to be a DFCS trend. As another example, similar or complementary low-level testing features have also been shown to be most valuable in the task of statistically calibrating low-level computer hardware fault-detection capability through intensive automated testing (Bensen, Mulcare, and Larsen, 1987).

## BIBLIOGRAPHY


Barton, L. A., D. B. Mulcare, and R. J. LeBlanc, "Predicate/Transition Network Analysis of Redundant Channel Synchronization," Computers in Aerospace V Conference, October 1985.

_____, "Predicate/Transition Analysis of Redundant Channel Synchronization," AIAA Paper 85-6020P, 1985.

Bensen, J. W., Development and Implementation of the Real-Time Six Degree of Freedom Airplane Simulation for the Reconfigurable Digital Flight Control System, Report Contract NAS2-10270, December 1981

Bensen, J. W., D. B. Mulcare, and W. E. Larsen, Hardware Fault Insertion and Instrumentation System: Experimentation and Results, DOT/FAA/CT-86/34, March 1987.

Boehm, B. W., "Verifying and Validating Software Requirements and Specifications," IEEE Software, January 1984.

Defeo, P., D. Doane, and J. Saito, An Integrated User-Oriented Laboratory for Verification of Digital Flight Control Systems: Features and Capabilities, NASA TM-84276, 1982.

Driscol, K. R., "Reversion, A Response to Generic Faults," AIAA Paper 85-1981CP, 1985.

Goldberg, J., General Concepts of Validation. Methods for Fault-Tolerant Avionics and Control Systems, March 1979.

Larsen, W. E. and A. Carro, "Digital Avionics Systems: Overview of FAA/NASA/Industry-Wide Briefing," 7th AIAA/IEEE Digital Avionics Systems Conference, December 1986.

McCabe, T. J., "A Complexity Metric," IEEE Transactions on Software Engineering, December 1976.

MIL-F-9490D, General Specification for Design, Installation, and Test of Piloted Aircraft Flight Control Systems, U.S. Air Force, 6 June 1975.

Mulcare, D. B., L. E. Downing, and M. K. Smith, Quadruplex Digital Flight Control System Assessment, DOT/FAA/CT-86/30, November 1987.

Mulcare, D. B., W. G. Ness, and R. M. Davis, "Analytical Design and Assurance of Digital Flight Control System Structure," AIAA Journal of Guidance, Dynamics, and Control, May-June 1984.

Ness, W. G., et al., Integrated Assurance Assessment of a Reconfigurable Digital Flight Control System, DOT/-FAA/CT-82/154, April 1983.

Reed, J. E. and E. M. Boothe, "Digital Avionics, Active Controls, and the FAA: Advanced Integrated Flight Systems (AIFS)," 2nd AIAA/IEEE Digital Avionics Systems Conference, November 1977.

AC 25.1309-1, System Design Analysis, September 7, 1982.

Wensley, J. H., Design for Validation. Validation Methods for Fault-Tolerant Avionics and Control Systems, March 1969.

## GLOSSARY

ANALYTICAL ROOT SOLUTION. Information obtained from the roots of the characteristic equations of the airplane model such as short-period frequency response.

ANGLE OF ATTACK. Angle between the longitudinal axis of an aircraft and the direction of movement.

ASSURANCE ASSESSMENT. Procedures whose purpose is to ensure that a proposed system functions according to design specifications.

BROADCAST. Transmission of messages to all terminals without reference to the identification of the receiving station or terminal.

CHORD. The straight line segment intersecting or touching an airfoil profile at two points.

DECOUPLED MANEUVERS. Changes in an aircraft's direction and attitude in one axis without affecting direction or attitude in other axes.

DESIGN ERROR. A functional flaw resulting from a misinterpretation of the specifications of the system.

DOUBLE FAIL-OPERATIONAL SYSTEM. A quadruplex (or higher) redundant flight control system which is designed to incur failures in two redundant lanes (or channels) before it fails.

DUAL-DUAL ARCHITECTURE. Two parallel dual computers with a voting plane at the output of each dual computing lane.

ENVELOPE LIMITING. General or additional limits imposed on the structural, "g" limits, speed, attitude, etc. of the aircraft. In some cases, envelope limiting imposes additional constraints on the envelope that cannot be exceeded regardless of pilot inputs.

ERROR. A mistake in specification, design, production, maintenance, or operation of a system causing undesirable performance.

EVENT, EXTREMELY IMPROBABLE. An event with a probability of occurrence on the order of $10^{-9}$ or less.

EVENT, IMPROBABLE. An event with a probability of occurrence on the order of $10^{-5}$ or less.

EVENT, PROBABLE. An event with a probability of occurrence greater than $10^{-5}$.

FAILURE. The inability of a system, subsystem, unit, or part to perform within specified limits.

FAILURE, HIDDEN. A failure that is not manifested at the time of its occurrence.

FAULT. An error in the operation of a system.

FAULT, HARD. A defect in the hardware or software of a digital control system that permanently affects some functional performance of the system.

FAULT, SOFT. A transient defect in the software of a digital flight control system that can be overcome by error-correctable code or by recycling of power to the computer system.

FAULT INSERTION. A testing technique used to obtain information about data latency and built-in test coverage of a digital flight control system.

FAULT TOLERANT. Software which continues to operate satisfactorily in the presence of faults.

FAULT TREE ANALYSIS. A top-down deductive analysis that identifies the conditions and functional failures necessary to cause a defined failure condition. The fault tree can be used to establish the probability of the ultimate failure condition occurring as a function of the estimated probabilities of contributory events.

FLIGHT CODE. The application software of the digital flight control system.

FLIGHT-CRITICAL. A description of functions whose failure would contribute to or cause a failure condition preventing the continued safe flight and landing of the aircraft.

FLIGHT-ESSENTIAL. A description of functions whose failure would contribute to or cause a failure condition which would significantly affect the safety of the airplane or the ability of its crew to cope with adverse operating conditions.

FLIGHT-PHASE CRITICAL. A description of functions which are critical only during certain phases of flight.

FLY-BY-LIGHT. Flight control system where fiber optics carry the signal.

FLY-BY-WIRE. Flight control system with electric signaling.

GROUND EFFECT. Increase in aircraft lift when operating near the ground.

MEAN AERODYNAMIC CHORD (also mean chord). The chord of an airfoil whose length is equal to the area of the airfoil section divided by the span.

NEGATIVELY STABILIZED. Aircraft design in which the point of effective lift is aft of the center of gravity.

POINT-MASS SIMULATION. Same as state variables airplane simulation (q.v.)

POSITIVELY STABILIZED AIRCRAFT. Aircraft design in which the effective point of lift is forward of the center of gravity.

PREDICATE/TRANSITION NETWORK. A bipartite graph (a type of linear graph) to model concurrency between redundant concurrent events. Basically a modified generalized Petri net.

QUADRUPLEX ARCHITECTURE. The use of four separate lanes (or channels) of computer redundancy. Each lane can fail separately providing a fail-operational squared capability for the digital flight control system.

RELAXED STATIC STABILITY AIRCRAFT. An aircraft whose center of gravity is behind the wing's point of effective lift.

RELIABILITY ANALYSIS. A means of determining the probability of failure in a system. Military flight-critical systems typically are required to have reliability levels of $10^{-5}$ to $10^{-7}$, whereas civil flight-critical systems have reliability levels of $10^{-9}$ or less.

STATE-VARIABLE AIRPLANE SIMULATION (also point-mass simulation). Fixed aerodynamic variables are used in the solution of the equations of motion of the model instead of using look-up tables in which each aerodynamic derivative varies with airspeed, altitude, etc. The model performance is only accurate at or near the point in the flight envelope for which the variables are chosen.

STATIC MARGIN. The degree of instability in a relaxed statically stable airplane.

SYSTEM EXPOSURE TIME. The period during which a system may fail. This period extends from the last verified proper functioning to the completion of the next required performance.

TIME CONSTANT. Time required to double amplitude of the divergent real root in the pitch axis of the aircraft model.

TRANSPARENT RECOVERY. Correcting a soft fault without interrupting the system's intended performance.

VALIDATION. Demonstration and authentication that a final product operates in all modes and performs consistently and successfully under all actual operational and environmental conditions founded upon conformance to the applicable specifications.

VERIFICATION. Demonstration by similarity, previous in-service experience, analysis, measurement, or operation that the performance, characteristics, or parameters of equipment and parts demonstrate accuracy, show the quality of being repeatable, and meet or are acceptable under applicable specifications.

## ACRONYMS AND ABBREVIATIONS

| | |
|---|---|
| AC | Advisory Circular |
| AFBW | Augmented Fly-By-Wire |
| AOA | Angle of Attack |
| CAPS | Collins Adaptive Processor System |
| DFCS | Digital Flight Control System |
| FAA | Federal Aviation Administration |
| FAR | Federal Aviation Regulation |
| FBW | Fly-By-Wire |
| FCC | Flight Control Computer |
| FCS | Flight Control System |
| MAC | Mean Aerodynamic Chord |
| MDICU | Modular Digital Interface Conversion Unit |
| MTBF | Mean Time Between Failures |
| MUX | Multiplexer |
| RDFCS | Reconfigurable Digital Flight Control System (facility) |
| RSS | Relaxed Static Stability |
| SAS | Stability Augmentation System |

U.S. Department of Transportation
**Federal Aviation Administration**

DOT/FAA/CT-88/10

# HANDBOOK-VOLUME II
# DIGITAL SYSTEMS VALIDATION

## CHAPTER 5
## ADVANCED FAULT INSERTION AND SIMULATION METHODS

**PREPARED BY:**

**COMPUTER RESOURCE MANAGEMENT, INC.**
**950 HERNDON PARKWAY, SUITE 360**
**HERNDON, VIRGINIA 22070**

**PREPARED FOR:**

**FEDERAL AVIATION ADMINISTRATION**
**TECHNICAL CENTER**
**ATLANTIC CITY INTERNATIONAL AIRPORT, NEW JERSEY 08405**

NOTICE

This document is disseminated under the sponsorship of the U.S. Department of Transportation in the interest of information exchange. The United States Government assumes no liability for the contents or use thereof.

The United States Government does not endorse products or manufacturers. Trade or manufacturers' names appear herein solely because they are considered essential to the objective of this report.

TABLE OF CONTENTS


| Section | | Page |
|---|---|---|
| 1. | FAULT TOLERANT SYSTEMS RELIABILITY PREDICTION BACKGROUND | 5-1 |
| 1.1 | Introduction and History | 5-1 |
| 1.2 | Reliability Requirements | 5-2 |
| 1.3 | Fault Tolerant Systems Performance Issues | 5-2 |
| 1.3.1 | Processor Redundancy | 5-2 |
| 1.3.2 | Byzantine Resilience | 5-3 |
| 1.3.3 | Synchronization | 5-3 |
| 1.3.4 | Scheduling | 5-4 |
| 1.3.5 | Deadlock Protection | 5-4 |
| 1.4 | Reliability Prediction Issues | 5-4 |
| 1.5 | Fault Modeling | 5-7 |
| 1.5.1 | Origin of Faults | 5-7 |
| 1.5.2 | Fault Models | 5-9 |
| 1.6 | Fault Tolerant Techniques | 5-11 |
| 2. | FAULT INJECTION SIMULATION AND TEST METHODS | 5-19 |
| 2.1 | NASA FT System Research Summary | 5-20 |
| 2.1.1 | Architectural Design Methods | 5-21 |
| 2.1.2 | Detail Design Methods | 5-21 |
| 2.1.3 | Developmental Phase Methods | 5-21 |
| 2.1.4 | Systems Implementation and Evaluation Methods | 5-22 |
| 2.2 | Fault Insertion Methods | 5-22 |
| 2.3 | Software Simulation Fault Insertion Methodology | 5-26 |
| 2.3.1 | Method | 5-26 |
| 2.3.2 | Application | 5-26 |
| 2.3.3 | Example Results | 5-26 |
| 2.3.4 | Limitations | 5-29 |

TABLE OF CONTENTS


| Section | | Page |
|---|---|---|
| 2.4.1 | Method | 5-30 |
| 2.4.2 | Application | 5-31 |
| 2.4.3 | Example Results | 5-31 |
| 2.4.4 | Limitations | 5-32 |
| 2.5 | Fault Emulation Fault Insertion Methodology | 5-33 |
| 2.5.1 | Method | 5-33 |
| 2.5.2 | Application | 5-33 |
| 2.5.3 | Example Results | 5-34 |
| 2.5.4 | Limitations | 5-39 |
| 2.6 | Physical Fault Insertion Methodology | 5-39 |
| 2.6.1 | Method | 5-39 |
| 2.6.2 | Application | 5-39 |
| 2.6.3 | Example Results | 5-40 |
| 2.6.4 | Limitations | 5-45 |
| 3. | ADDITIONAL FAULT INSERTION AND TEST EXAMPLES | 5-47 |
| 3.1 | Draper Labs FTMP Tests Using Fault Emulation | 5-47 |
| 3.1.1 | Gate-Level Model For Redundancy Communicator Interstage | 5-50 |
| 3.1.2 | Fault Detection Effectiveness of Diagnostic Tests | 5-52 |
| 3.1.3 | Experimental Results on Hardware/Software | 5-54 |
| 3.2 | Electromagnetic Induced Transient Injection Test Examples | 5-57 |
| 3.2.1 | Electromagnetic Transients and Fault Injection | 5-57 |
| 3.2.2 | Upset and Failure Detection | 5-60 |
| 3.2.3 | Digital Engine Controller Test Examples | 5-62 |
| BIBLIOGRAPHY | | 5-71 |
| GLOSSARY OF TERMS | | 5-76 |
| ACRONYMS AND ABBREVIATIONS | | 5-78 |

LIST OF ILLUSTRATIONS


| Figure | | Page |
|---|---|---|
| 1.4-1 | FAULT TREE RELIABILITY MODEL | 5-5 |
| 1.4-2 | MARKOV MODEL QUADRUPLEX WITH LATENT FAULTS | 5-6 |
| 1.5-1 | SIMPLIFIED BREAKDOWN OF FAULT TYPES | 5-8 |
| 1.6-1 | FAULT CONTAINMENT REGION REQUIREMENTS | 5-12 |
| 1.6-2 | ARCHITECTURE FOR CONGRUENT TRANSFER FROM SIMPLEX SOURCE | 5-14 |
| 1.6-3 | FAULT TOLERANT PROCESSOR - FUNCTIONS | 5-16 |
| 1.6-4 | FAULT TOLERANT PROCESSOR - PHYSICAL VIEW | 5-17 |
| 2.0-1 | FAULTED PROCESSOR PERFORMANCE MEASUREMENT | 5-20 |
| 2.2-1 | FAULT LATENCY DATA | 5-25 |
| 2.3-1 | FAULT LATENCY DISTRIBUTION - GATE LEVEL FAULTS | 5-27 |
| 2.3-2 | FAULT LATENCY DISTRIBUTION - COMPONENT PIN LEVEL FAULTS | 5-28 |
| 2.3-3 | LOGIC DIAGRAM ILLUSTRATING DETECTABLE FAULTS | 5-29 |
| 2.4-1 | FAULT EMULATION COMPUTATION TECHNIQUE | 5-30 |
| 2.5-1 | FIAT FAULT INSERTION AND EXPERIMENT MONITOR | 5-34 |
| 2.5-2 | FAULT INSERTION MONITOR SOFTWARE | 5-35 |
| 2.5-3 | FIAT EXPERIMENT INTERFACE TREE | 5-35 |
| 2.5-4 | DETAILED VIEW OF FAULT INJECTION RECEPTOR SOFTWARE | 5-36 |
| 2.5-5 | REAL-TIME CHECKPOINTING SYSTEM | 5-37 |
| 2.6-1 | FAULT INSERTION AND INSTRUMENTATION SYSTEM (FIIS) | 5-40 |
| 2.6-2 | FTMP VAX DATA ACQUISITION ENVIRONMENT | 5-41 |
| 2.6-3 | DEVICE PIN ACCESS FOR FAULT INJECTION AND MONITORING | 5-41 |
| 2.6-4 | FAULT INJECTOR HARDWARE | 5-43 |
| 3.1-1 | EXPERIMENT DIAGRAM | 5-47 |
| 3.1-2 | C/I EXCHANGE NETWORK | 5-49 |
| 3.1-3 | ACTUAL CIRCUITRY FOR CROSS CHANNEL LINKS | 5-51 |
| 3.1-4 | SYSTEM FLOW DIAGRAM | 5-53 |
| 3.1-5 | C/I MODEL FOR DIAGNOSTIC EMULATION | 5-55 |
| 3.1-6 | DIAGNOSTIC TEST PERFORMANCE | 5-56 |
| 3.1-7 | COMBINED PERFORMANCE OF ALL NON-VOTER TEST | 5-56 |
| 3.2-1 | TIME-DOMAIN TRANSIENT WAVEFORM GENERATED BY LIGHTNING | 5-58 |
| 3.2-2 | BLOCK DIAGRAM OF THE HAMILTON STANDARD EEC | 5-63 |
| 3.2-3 | SIMULATION OF EEC INPUTS FOR EM TRANSIENT EXPERIMENT | 5-64 |
| 3.2-4 | BLOCK DIAGRAM OF EM TRANSIENT TEST INSTRUMENTATION | 5-65 |
| 3.2-5 | CLARIFICATION OF GATE DISTANCE DEFINITIONS | 5-67 |

LIST OF TABLES


| Table | | Page |
|---|---|---|
| 1.5-1 | FAULT EFFECTS: MANUFACTURING VS. OPERATIONS | 5-9 |
| 1.5-2 | FAULT MODELS AT DIFFERENT LEVELS OF ABSTRACTION | 5-10 |
| 2.1-1 | VALIDATION AREAS FOR THE DESIGN PROCESS | 5-21 |
| 2.2-1 | FAULT INSERTION METHODS | 5-22 |
| 2.2-2 | FAULT INSERTION EXAMPLES | 5-23 |
| 2.5-1 | ERROR DETECTION STATISTICS - CHECKPOINTING ONLY | 5-38 |
| 2.5-2 | ERROR DETECTION STATISTICS - CHECKPOINTING AND CHECKSUMS | 5-38 |
| 2.6-1 | EXPERIMENTAL RESULTS SUMMARY FROM FIIS | 5-44 |
| 3.1-1 | TESTS USED FOR EXPERIMENT | 5-54 |
| 3.1-2 | SUMMARY OF UNDETECTED FAULTS | 5-57 |
| 3.2-1 | ERRORS IN SIMULATED EM TRANSIENT ON EEC | 5-69 |
| 3.2-2 | ERRORS IN SIMULATED EM TRANSIENT ON MICROPROCESSOR MODULE | 5-69 |

# 1. FAULT TOLERANT SYSTEMS RELIABILITY PREDICTION BACKGROUND

## 1.1 Introduction and History

Application of microprocessor-based computers to flight critical systems is becoming more commonplace. The flight management systems on board the newest series of aircraft bring to aviation a new capability in management of the air traffic and additional capability for safe flight in marginal weather conditions. All-electronic engines have come into use on the latest generation aircraft as well. Stability augmentation puts the control of aircraft loads under computer control. The latest Airbus has brought military fly-by-wire concepts into commercial aviation for stability and control. This Airbus, operated by a new generation of on-board computers, offers greater fuel efficiency and smoother flight path control under varying aircraft configurations. Additional computer applications are under development to increase future aircraft system automation for stability and control, and for flight management. Considerable attention has been directed toward the technology and methods for validating the performance of these flight critical systems (National Aeronautics and Space Administration [NASA] Conference Publication 2114, Working Group Meetings I and II).

Failures of these advanced electronic systems would place a severe load on the pilot and crew in operating the aircraft without automation. The reliability of electronic components is not good enough to ensure a sufficiently low probability of failure in flight critical systems. Computer processors, having millions of components on a single chip of silicon, are available. Testing of these devices under all possible conditions is not feasible. Single logic circuit failures within these high density Very Large Scale Integration (VLSI) components must not cause loss of function. Such loss of function could arise from either a random failure or a defect present within the device that escaped detection in the original design and fabrication of the device. If a microprocessor is part of a system, the user must assume that the device may have some inherent fault that eluded detection during manufacturing.

The addition of duplicate, redundant, computers is a promising approach to ensure that at least one good computer will be available in the event of failure during flight. Redundancy has been useful in many prior aircraft systems; the low cost, weight, and power requirements of microelectronic devices make this attractive. Pioneers in reliability modeling demonstrated mathematically that system reliability could be enhanced by redundant hardware. These results are based upon the assumptions that the devices were initially fault free, that failures occur independently, and that errors can be detected by comparing the results from redundant components. More recently it has been shown that increased reliability can be attained, even if the components have inherent faults and not all failures can be detected (Arnold 1973).

The critical factor in achieving increased reliability through redundant components is the determination of how the system detects and processes faults. Therefore, the study of faulted behavior of digital circuits has expanded for two reasons: (1) to determine whether the output is correct for a given input and the device is sufficiently complete to manufacture and distribute, and (2) to determine the effectiveness of the device or system to detect, isolate, and recover from faults and errors. Because of the complexity of the computer software and the system (i.e., the number of component circuits), the entire study and testing is viewed as a stochastic process.

This chapter deals with system reliability issues in depth, through discussion of the concepts and methods for system testing by inserting faults and measuring the recovery process. An extensive bibliography is included for further reading on this subject (references that are given in the text by the author's last name and date refer to entries in the bibliography). A majority of the discussion material results from efforts by NASA Langley over the last ten years. When NASA began these investigations, reliability modeling was almost entirely based upon the failure modeling of single components. Data for these models were obtained for military programs from accelerated life cycle testing and 100 percent acceptance tests on receipt. These methods are too costly and too restrictive for commercial programs. Furthermore, the concept of 100 percent testing of microcomputers has proved impractical to implement, even for MIL-SPEC parts, due to the size and complexity of the test effort. Some of the work on fault insertion and testing of fault tolerant systems is still in progress and all the answers are not available for this challenging topic.

## 1.2 Reliability Requirements

The reliability of a system is the probability of performing a given function from some initial time, t=0, to time t. Typically, it is more convenient to deal with the probability of failure, which is one minus the reliability. For mission critical applications, a typical requirement is that the probability of failure be no more than $10^{-5}$ per hour. For flight critical safety related applications, the acceptable probability of failure is $10^{-9}$ per hour.

## 1.3 Fault Tolerant Systems Performance Issues

### 1.3.1 Processor Redundancy

Redundancy uses the execution of the same code on two processors to detect and correct the effects of faults before they propagate errors into other tasks in a system. Reliability predictions are based upon independence of the faults in the individual processors; i.e., a fault affecting one processor must not be able to cause an error in another processor. Fault independence can be enforced by separating the computations into fault containment regions (FCRs). An FCR is a collection of components that operates correctly regardless of any arbitrary logical or electrical fault outside of the region. If the entire system is partitioned into FCRs, an arbitrary fault inside such a region of containment cannot cause any component outside that region to fail in any manner. Therefore, fault tolerance includes containment of faults and correct handling of errors.

In order to use hardware redundancy effectively against random component failures, it is necessary that faults in different FCRs be independent and uncorrelated. To avoid common mode failures, each FCR must have a power supply and clock independent and separate from those of any other region. Furthermore, it is necessary to provide electrical isolation at all inter-FCR communication points.

Duplex redundancy is sufficient to detect errors, however, triplex redundancy is needed to mask errors. Redundancy is very effective if all the computations are performed in lock step with identical inputs and intermediate steps. Exact match voting can then be used to detect and mask errors. Thus any value not bit-for-bit identical to the other two is considered to be in error and is discarded. The input values must be interchanged between the FCRs to ensure that the same values are used by all three processors. If the processors are not in lock step synchronization, exact match voting cannot be used, and fault detection and masking is compromised.

### 1.3.2 Byzantine Resilience

A fault tolerant process is termed "Byzantine Resilient" if it is tolerant of intermittent faults that can send good information part of the time. The term alludes to the problem faced by Byzantine generals trying to authenticate messages sent by traitorous messengers. Unfortunately computer components do not often fail in benign ways. Components involved in communication between processors have been known to fail in ways that send conflicting information to different parts of the system (Johnson 1989). If the redundancy is not sufficient to isolate these failures, then they are very difficult to isolate and correct. The difficulty arises because a processor correctly processing incorrect input will appear faulty to the rest of the system. In order for a system of parallel processors to be Byzantine resilient the network topology must be divided into four separate FCRs. Each computation must be performed in two separate regions; whenever data values pass from one containment region to another the communications must be via separate paths. If a containment region no longer has three separate paths for communications, then the system will no longer be Byzantine Resilient. The reconfiguration of such a system is very complex.

### 1.3.3 Synchronization

Synchronizing individual processors in a redundant computation makes it possible to schedule completion of individual process steps. This simplifies the determination of whether a processor has failed or simply is slow to complete a task. Failures of synchronized processors must not propagate beyond a containment region. Global clocking methods are not recommended because they permit too high a chance of a single point of failure. Much research has been done on synchronization methods. Various methods range from loosely coupled methods to lock step synchronization. The computation overhead for correctly dealing with processor synchronization is usually very large.

### 1.3.4 Scheduling

Task scheduling in the redundant processors requires a real-time executive that can operate in the presence of faults. This executive must be responsible for reconfiguration and fault masking on data between processors. The executive must also maintain data integrity in the presence of faults. This means that multiple copies of the executive must exist in the various FCRs.

### 1.3.5 Deadlock Protection

It is necessary to guard against the deadlock condition where every processor is waiting for a task to be completed as a subtask on another processor. Parallel processors require specific software measures to prevent deadlock. If the operating system provides no deadlock control during execution, then it must be shown, for each process, that no deadlock can occur in the worst-case combination of events. Implementation of a program to ensure that deadlock cannot occur is extremely difficult. The information necessary to determine deadlock is usually not available until operational testing is performed. Deadlock recovery schemes often use rollback to a prior processing step where the state was recorded. A combination of avoidance and recovery is usually sufficient to handle deadlock. If avoidance is used, it must be shown that the system will avoid deadlock under all possible circumstances or that deadlock is extremely unlikely (i.e., the probability acceptably low). If roll-back recovery is used, the probability that the system will miss a real-time deadline due to roll-back must be shown to be acceptably low.

## 1.4 Reliability Prediction Issues

Modern reliability prediction of fault tolerant systems is a mathematically intensive problem usually requiring computer modeling and analysis tools such as those developed at or in conjunction with NASA Langley (Hybrid Automated Reliability Predictor [HARP], Fault Tree Compiler [FTC], Computer Aided Reliability Evaluator [CARE III], Pade Approximation With Scaling [PAWS], Scaled Taylor Expansion Matrix [STEM], Semi-Markov Unreliability Range Evaluator [SURE], etc.). These computer programs differ from one another in their modeling complexity and solution techniques, but have been shown to be largely in agreement for predicting reliabilities of systems within each of their stated limitations.

The reliability models utilized are limited by their techniques for mathematically describing hardware-type events and solution algorithms. They are also limited by the availability of accurate data for input to the model.

Typically, the failure and recovery of a system of components is described by some type of fault tree diagram, such as those illustrated in figures 1.4-1 and 1.4-2. These modeling examples provide just a small glimpse of the complexity of modern reliability prediction problems.

One of the most direct reliability models is a fault tree. The fault tree shown in figure 1.5-1 is used to calculate the failure probability of the system for raising and lowering aircraft landing gear. This type of model was developed in 1961 by Bell Telephone Laboratories. The technique starts with a single

event and delineates all possible "basic event" paths to that single event. Each basic event must be independent, the characteristic distinction of a random/stochastic process. Each fundamental event must also have a known probability (perhaps calculated from another fault tree). This fault tree, in itself, will not allow for the reconfiguration in a modern fault recovery process, but it is a valuable tool for ascertaining the probabilities and failure rates of other parts of a system.

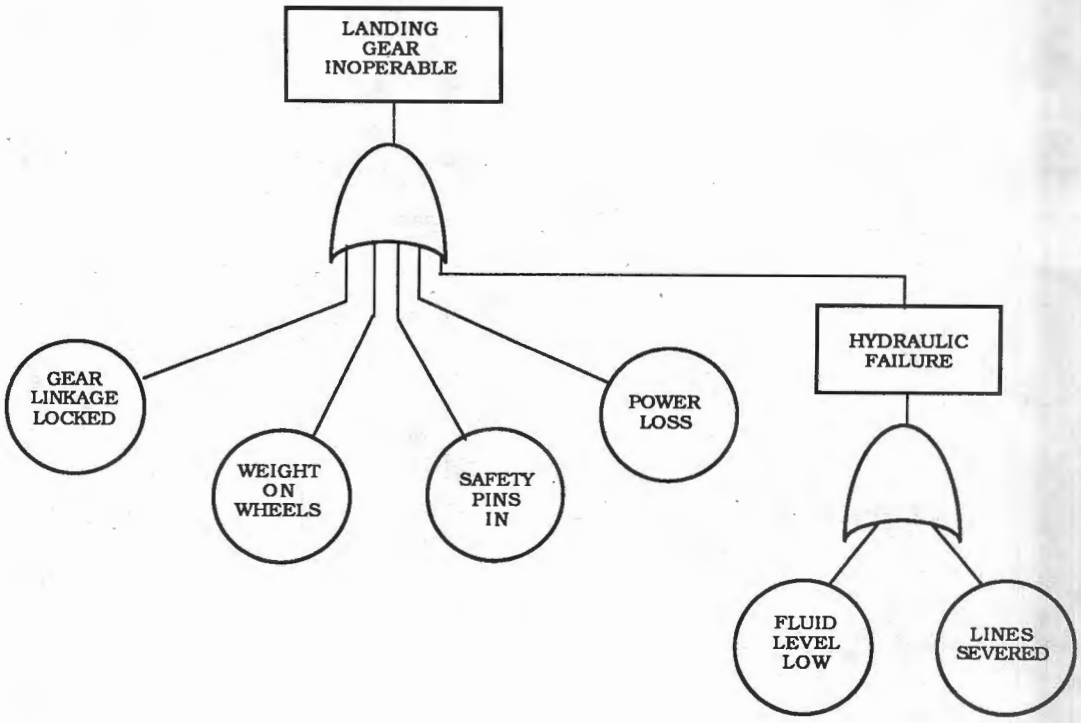
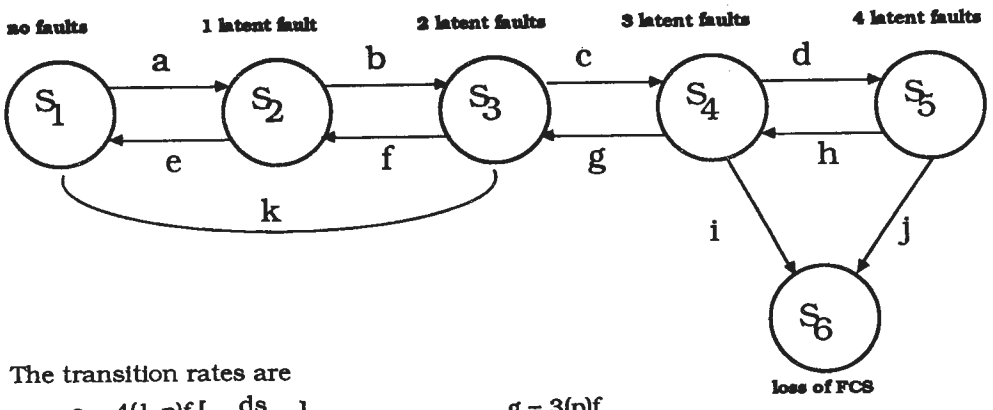



FIGURE 1.4-1. FAULT TREE RELIABILITY MODEL

A Markov model is required to incorporate the fault recovery reconfiguration process. Figure 1.4-2 illustrates a Markov model for a quadruplex Flight Control System (FCS) with latent faults (McGough 1988). The key difference from the fault tree of figure 1.4-1 is the incorporation of multiple end states, thus allowing for reconfiguration paths through activation of redundant components or degradation to simplex states (node 7 in figure 1.4-2). Transitions to each state are characterized by a time rate for failure and recovery. The Markov model becomes semi-Markov if the recovery rates, F(t), are not simply approximated by the common exponential distribution rates used for component failures. However, this adds significant complication to the mathematics required for solution of the model.

Computation of the reliability of a fault tolerant system is more complex than the examples shown in figures 1.4-1 and 1.4-2. The reliability of the overall system is a function of the reliability of the individual components of the system. It is also a function of the fault recovery process used in the fault tolerant system. Individual part reliabilities may be obtained effectively from life cycle testing and materials analysis. This data is fundamental to the failure prediction methods used by the most common component reliability prediction (MIL-HDBK-217E 1986). Including behavior such as fault recovery in the system during the presence of a fault is even more challenging. This data must be obtained from tests run on both hardware and software under controlled fault injection into real and simulated systems. Data obtained in this manner is ordinarily specific to the system under test.




The transition rates are

$a = 4(1-p)f\left[\frac{ds}{as+ds}\right]$

$b = 3(1-p)f$

$c = 2(1-p)f$

$d = (1-p)f$

$e = as + (p)f$

$f = 2(p)f$

$g = 3(p)f$

$h = 4(p)f$

$i = as$

$j = as$

$k = as$

**Note that the backwards transitions (e, f, g, h, and k) are the result of repairs due to detection of faults.**

FIGURE 1.4-2. MARKOV MODEL QUADRUPLEX WITH LATENT FAULTS

As has been indicated, the failure rate of components is an essential part of the data needed for reliability prediction. Studies have centered around the testing of components in manufacturer's plants and during incoming quality assurance inspections and testing at systems manufacturing facilities. The most comprehensive data is available for the large volume components and originates from the larger manufacturers. Testing is almost nonexistent at small and low volume commercial facilities. Device testing is much more stringent for MIL-SPEC parts.

Logic devices may be tested by generation of test pattern inputs to a device. Their outputs can then be compared to a known correct response. This approach

works well for combinatorial devices, but not so well for microprocessors. Testing of microprocessors has become a significant problem due to the incredible complexity of VLSI designs. Millions of logic gates, sequential and parallel data propagation, and very fast (to the tens of nanoseconds) timing relationships make it virtually impossible to expose the microprocessor to all possible combinations of inputs and outputs. For most applications the level of testing is a direct function of economics.

The inclusion of redundant fault tolerant hardware helps circumvent overall system failure due to the failure of a single component. Studies (Arnold 1973 and Johnson 1989) have shown that system reliability is enhanced through redundancy even with less than 100 percent fault detection; however, the fundamental operation of the fault tolerant system is fault detection and recovery. Therefore, fault recovery data is crucial to the accuracy of the reliability prediction model. One of the significant purposes of controlled fault injection into fault tolerant systems is the procurement of fault recovery data which may be used in the reliability prediction model. Complex and innovative system architectures make it difficult to predict fault recovery behavior based upon history and experiments on other types of systems. Experimental fault injection is usually required to ascertain a specific system's reliability.

## 1.5 Fault Modeling

Validation of fault toleration systems requires an understanding of faults. This section discusses the origin of various faults and models that have been defined for them.

### 1.5.1 Origin of Faults

The wide range of faults and where they occur in digital systems performance is illustrated in figure 1.5-1. These faults may be classified in three general areas: design errors, manufacturing errors leading to faults, and faults occurring in-service.

Design errors are the result of improperly translating an idea or concept into an operating system. Gathering information on design errors is difficult because each error occurs once per system development and the errors are usually complex. Design errors have been extensively studied in the software realm due to the increased emphasis on reliable software and the fact that software faults are entirely due to design and coding errors. (In general, coding errors will be identified and eliminated during testing.) Such errors often enter into the design of algorithms where partitioning into separate regions of validity is necessary; for example, with square roots, arc tangent, arc sine, and so on. If sufficient testing is not done between regions, the value at the region boundary can become undefined or very large or small. The effects of these errors are difficult to generalize due to their infrequency and diversity, hence few generalized models are available. These faults may produce little or no effect until just the exact combination of input conditions and time history present themselves.

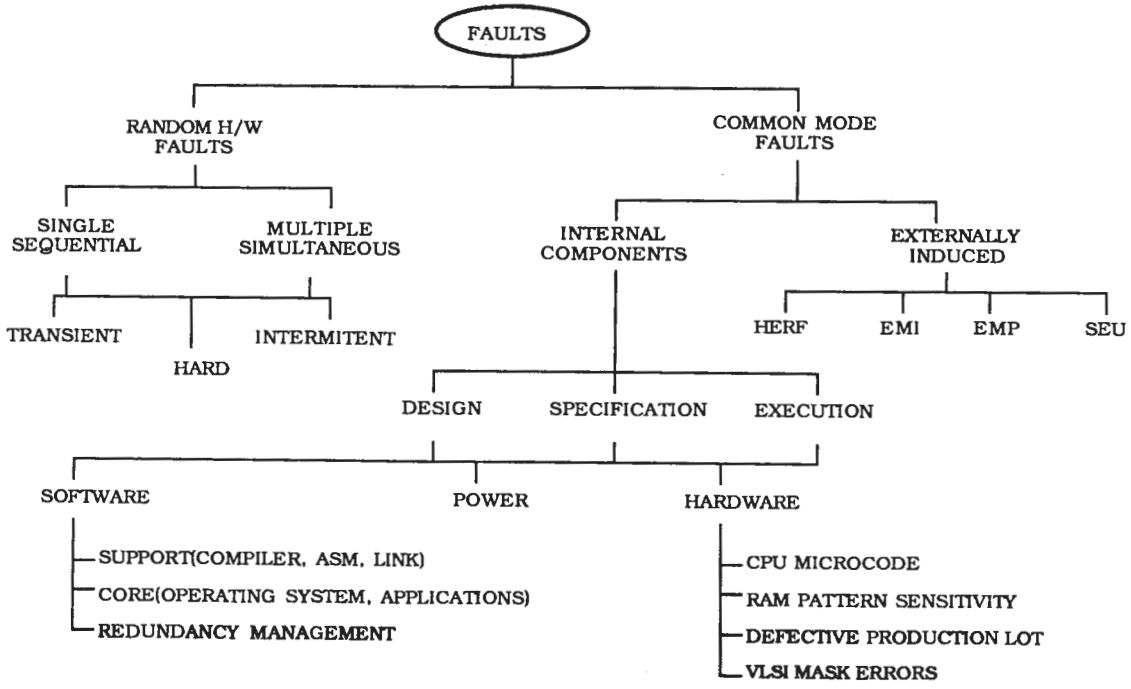



FIGURE 1.5-1. SIMPLIFIED BREAKDOWN OF FAULT TYPES

Manufacturing defects may be introduced by improper processing or by the use of defective materials in the fabrication of the system. Examples of manufacturing defects in semiconductor devices include cracked dies, over or under doping of the semiconductor material, and flaws in the mask. Effects of manufacturing defects on Complementary Metal-Oxide Semiconductor (CMOS) circuit behavior have been characterized. In a manufacturing research program, defective masks were utilized to fabricate several circuits and defective parts examined for faults (Ferguson 1987, Marchall 1985). Results of this study determined that 99 percent of all faults can be characterized by bridging or breaks, while only 50 percent of the faults can be modeled by stuck line faults.

In-service failures are caused by physical processes occurring through normal and abnormal use over the life of the system. Following are examples of CMOS device failures:

- Oxide breakdown due to high electric fields in the insulator, usually in the gate oxide.
- Electromigration, drifting of metal atoms toward the cathode, influenced by high current density in the semiconductor.
- "Hot electron" trapping in the gate oxide in the channel of a Metal-Oxide Semiconductor (MOS) transistor due to high temperature and high electric fields.

- Electron hole pairs or alterations to stored charge due to impacts by cosmic rays and alpha particles.

- Physical failures due to electrical overstress caused by overvoltages, electromagnetic (EM) environment (interference (EMI), radiation (EMR), lightning), and electrostatic discharge.

Other life cycle failures are caused by many other sources including design deficiencies, production techniques, mechanical stress, thermal stress, moisture, and corrosion.

Very little has been published on the distribution of in-service faults due to manufacturing processes and the operating environment. Attempts to map physical failures into logical models are not satisfactory. Several failures resulted in stuck-at behavior, while others resulted in parametric faults. An interesting contrast is apparent between in-service and manufacturing operations. The results of one study (Ferguson 1987, reported in Czeck, Siewiorek, and Segall 1989), are shown in table 1.5-1.

TABLE 1.5-1. FAULT EFFECTS: MANUFACTURING VS. OPERATIONS

| FAULT EFFECT: | MANUFACTURING STAGE | FIELD |
|---|---|---|
| Oxide | Less than 10% | 25% - 75% |
| Metal | 30% to 40% | 4% - 17% |

During in-service operations, the highest failure rate is for gate oxide (25 to 75 percent) and the next highest is for opens and shorts in metal runs (four to 17 percent). Oxide failures result in transistors being stuck off. Metal failures result in opened and shorted lines, and in breaks and bridges. During manufacturing, the most common failure (30 to 40 percent) is due to metalization, with a minor component (less than 10 percent) of faults causing transistors to have stuck-off faults.

### 1.5.2 Fault Models

Fault models are abstractions of failure mechanisms. They range from the physical device level to the gate, functional, and architectural level. A wide range of fault models have been developed to generate test programs. Fault models at various levels of abstraction, illustrated in table 1.5-2, cover the levels of system detail from network to switch.

TABLE 1.5-2. FAULT MODELS AT DIFFERENT LEVELS OF ABSTRACTION

| LEVEL | FAULT MODELS | BASIS | LIMITATIONS |
|---|---|---|---|
| Network | Communication lost, delayed or unordered. Lost nodes. | Abstraction of behavior. | Failure modes are unknown and complex. |
| Physical Message Switch | Data Change. Message or process lost. Data inconsistent. Time outs. | Abstraction of behavior. | Failure modes are unknown and complex. |
| Remote Terminal | Data change. Wrong assertion, source, or destination. | Abstraction of behavior. | Based on RT models not implementation. |
| Functional | Complement or dual function Truth table modification. | Observation? Ad hoc. | Many realizations and fault modes. |
| Gate | Gate output stuck at 0 or 1, Single stuck line. | TTL and PC board behavior. | Technology outdated. |
| Switch | New and missing devices, Shorts and breaks (opens), Transistors stuck on/off. | Processing defects. | Simulation overhead, difficult to observe and fault insert. |

Switch-level fault models are used for MOS devices where unidirectional logic gate models do not provide the bidirectional behavior of bridging, stuck open, and stuck closed transistors. Switch level models contain nodes connected by bidirectional transistor switches. Faults are modeled as nodes stuck high or low, transistors stuck open or closed, and extra or missing transistors. Low level simulations and models are required because the failure modes of the devices, especially CMOS, cannot be modeled at the gate or higher levels. Switch level simulations are a great aid in the design and generation of test programs, and assessment of test coverage, but the simulation is difficult to use for large systems due to the excessive run time for the simulations.

Gate-level fault models assume that inputs and outputs of gates are stuck at high or low logic levels, but the gates function correctly. These faults are based upon printed circuit boards, Transistor-Transistor Logic (TTL), and pre-TTL logic of the 1960s. The gate-level model is not applicable to MOS implementations since failure modes in CMOS can transform a combinational circuit into a sequential circuit. Furthermore, complex MOS circuit implementations do not map gate lines to circuit nodes.

Studies were conducted to determine the effects of processing mask defects on faults by mapping the defects to circuit fault models. These studies showed that only 50 percent of the faults are representable by gate-level models having a single line stuck-at fault. Further studies showed that the development of

fault models should be independent of the technology and updated as technology advances.

There is continuing research underway on higher levels of abstraction in fault models. Czeck, Siewiorek, and Segall 1989 provides a guide to this literature. This level of abstraction is driven by the VLSI in the electronic circuitry, and the complexity of dealing with testing based upon implementation faults. Some work has been completed laying the ground work for functional testing based upon the possible scenarios of failures occurring within the control section, data section, or data storage section of a generalized microprocessor. These models were independent of the implementation, hence they did not contain a fault library. The models were useful, however, in presenting a methodology for developing a set of test procedures for microprocessors.

## 1.6 Fault Tolerant Techniques

The degree to which a system can recover from faults depends critically upon the mechanisms for detecting and isolating faults. A careful distinction must be drawn between those techniques which are approximate and those which achieve near perfect fault detection and recovery. The following discussion is based upon a reference document from Charles Stark Draper Laboratory (CSDL) (1986).

Consider, for example, a dual channel system that relies upon bit-for-bit comparisons of the two outputs for fault coverage. This approach provides nearly perfect fault detection coverage. If a fault in one of the channels affects the correct operation of the channel output, then the output of that channel must differ from that of the other correctly operating channel. The latency between the occurrence of the fault and its appearance as an output error depends upon how frequently the malfunctioning hardware is exercised by the operating software (control routines and diagnostics). A fault will be readily detected when miscompare occurs, but it will not be obvious which channel is correct.

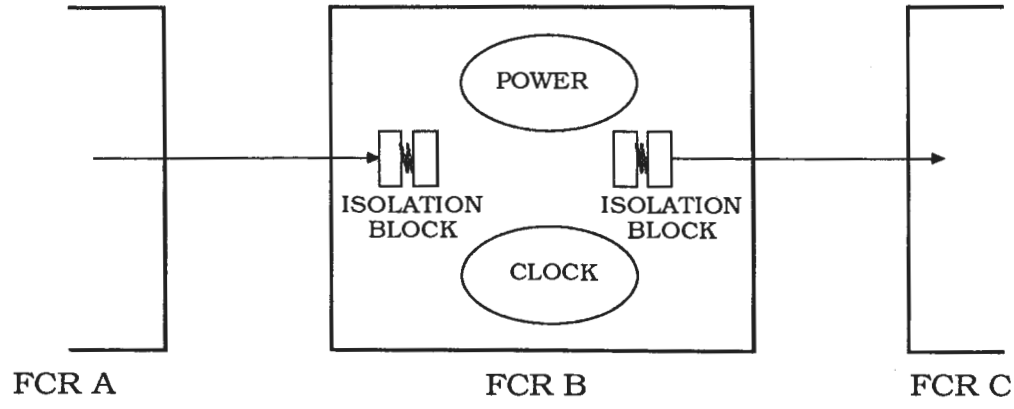
Diagnostic software routines may be run to isolate the channel which is faulty in a dual-channel system, but the coverage of the self-test software is often quite limited in its ability to detect faults. An example of a troublesome fault that is difficult to isolate is a pattern memory error in one bit position of the system memory. This fault may appear as a "dying one" (i.e., a one that is surrounded by all zeros occasionally turns into a zero). To catch such a fault requires a large number of runs with a great number of patterns on the memory. The likelihood that a "dying one" fault will be detected in a diagnostic test is not very large, and certainly not close to one hundred percent.

The addition of ad hoc solutions such as parity checks can help to isolate the memory pattern faults. Special circuitry to disassociate faults in the rest of a system adds considerable overhead in hardware and increased complexity. Furthermore, there is no technique for adding self-tests to logic circuits.

Other common techniques, which result in less than perfect coverage, include temporal checks, integral checks, and diagnostic checks. Watch dog timers have been used to catch certain hardware faults as well as programs stuck in infinite

loops. Integral checks are built into the basic number representation. Parity, Hamming codes, and Bose-Chaudadhary codes are all examples of programs written specifically to exercise the hardware. Examples of this type of check include Read-Only Memory (ROM) checksums, voter tests, and processor opcode tests. These methods add to the coverage of faults, but each one addresses only a limited number of possible faults, and unless all fault types are considered, the system will not achieve high fault coverage.

Current thinking on fault tolerant computer systems has identified four requirements that must be met to achieve a high degree of fault coverage. Any process which is critical to the operation of the system must be replicated. Critical processes must be correctly mapped into FCRs, such as defined in figure 1.6-1. There must be a sufficient number of FCRs and communication channels between them. And finally, there must be an "exact consensus" (bit-for-bit and synchronous, to be discussed later) among replicated processes. These requirements, described more fully below, are applicable for all processes, whether application program or operating system code.




FAULT CONTAINMENT REGIONS (FCR)
REQUIREMENTS:

1. INDEPENDENT POWER
2. INDEPENDENT CLOCKING
3. ELECTRICAL ISOLATION
4. PHYSICAL SEPARATION

FIGURE 1.6-1. FAULT CONTAINMENT REGION REQUIREMENTS

Most critical real-time control applications demand uninterrupted operation. Effects of faults must be masked by detection, isolation, and correction. These three processes must be mapped into three independent FCRs. In general, to mask the effects of f simultaneous faults, the nominal requirements call for (2f+1)

FCRs. However, this nominal requirement is not sufficient when the requirements for fault isolation are considered.

Fault isolation imposes additional requirements. The isolation of a faulty channel may be accurately determined for any output if the channels in the voting scheme are synchronized and have bit-for-bit agreement under normal, no fault, operation. Two processes which are initiated identically, receive identical inputs, have no failures, and operate on the inputs in the same way, are said to be congruent processes.

Congruence allows for a precise definition of the process of fault detection and isolation:

- Fault Detection - Two congruent processes not in bit-for-bit agreement indicate the presence of a fault.

- Fault Isolation - A process not in bit-for-bit agreement with the majority vote is the faulty process.

The concept of congruence has greatly simplified our thinking about fault detection and isolation. Since the processors are synchronized, the majority vote for a congruent process requires a simple truth table to determine the correct output and to isolate the faulty processor.

Incongruent processes need more complex fault detection and isolation algorithms and circuitry. These unsynchronized processes, having only approximately equal outputs, must rely on having "windows of agreement" or thresholds. In this case, fault detection coverage is a function of how precisely one can define these thresholds. For most dynamic processes these thresholds are a function of inputs and outputs. These parameters change with time and the operating mode of the systems. The use of application process driven thresholds for fault detection and isolation put a serious and uncalled for burden on the applications programs to ensure fault tolerance in the system.

Incongruent processes have been used successfully for tasks such as flight control, where computers were used as a one-for-one replacement of the analog counterparts. These implementations suffer from all the ills discussed earlier, including low fault coverage and excessive involvement of the applications programmer in redundancy management. Moreover, they also exhibit a high false alarm rate. These inexact consensus techniques will not work at all in more complex applications that require real-time fault isolation and reconfiguration of distributed system elements. Such systems communicate in abstract messages that have no semantic meaning (like actuator commands) if their redundant counterparts are only approximately equal.

Thus in the final analysis, fault detection, isolation, and masking need congruent processes. Achieving exact consensus in congruent processes having f faults, requires (3f+1) FCRs. This is more severe than stated earlier. In addition, processes must go through an algorithm for consensus that requires (f+1) communication rounds over links providing at least (2f+1) disjoint paths. For example, to tolerate a single fault there must be a least four FCRs, two communication rounds, and three disjoint paths.

This more precise mathematical basis provides a proof of fault tolerance from any arbitrary failure mode. This is driven from a need to account for more than Stuck-At-One (S-A-1) or Stuck-At-Zero (S-A-0) faults. A particularly difficult mode is one in which one receiver detects a one while the other receiver detects a zero. Such failures are termed Byzantine failures and can occur at sensor signal level, logic level, and even at program levels. For example, a redundancy management program in computer A can send the message to computer B that computer C is bad, while it tells computer C that computer B has failed. Although such modes may appear to be remote, they are in fact not unlikely. At least one in-flight failure of a triplex system has been traced to such a Byzantine fault and the lack of safeguards against such faults.

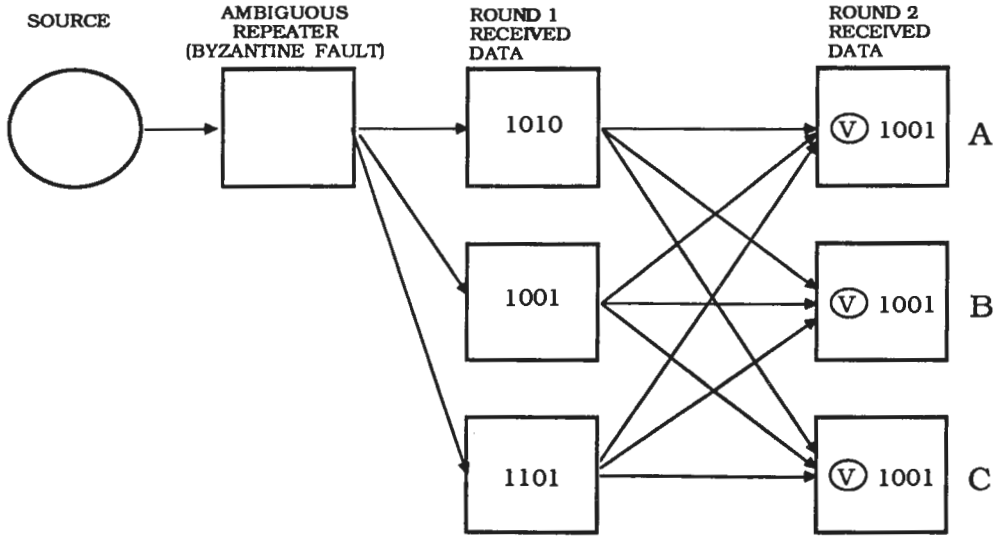



FIGURE 1.6-2. ARCHITECTURE FOR CONGRUENT TRANSFER FROM SIMPLEX SOURCE

An architecture that meets all the requirements for exact consensus in the presence of Byzantine faults is shown in figure 1.6-2. This diagram shows six FCRs that are interconnected by three disjoint paths and go through two-round communication protocol. Data is distributed, in each channel, to voters that produce majority voted outputs. The voters perform a two-out-of-three vote on a bit-for-bit basis. The least significant bits of the three values are voted to produce the least significant bit of the majority output. The next order bit in the three words is then voted to produce the next bit of the output. This process continues until the most significant bit is voted to produce an output word that is the bit-for-bit majority of the three inputs. In the diagram, a Byzantine fault in the form of an ambiguous repeater is postulated in the channel that is distributing the value of the sensor attached to that channel. After the two-round exchange and the bit-for-bit vote, it is evident that all correctly operating channels have identical or congruent values.

The CSDL Fault Tolerant Processor (FTP) architecture for implementing the requirements for fault containment and consensus voting illustrates these considerations. A picture of that fault tolerant implementation is shown in two views: a functional view shown in figure 1.6-3, and a physical layout shown in figure 1.6-4.

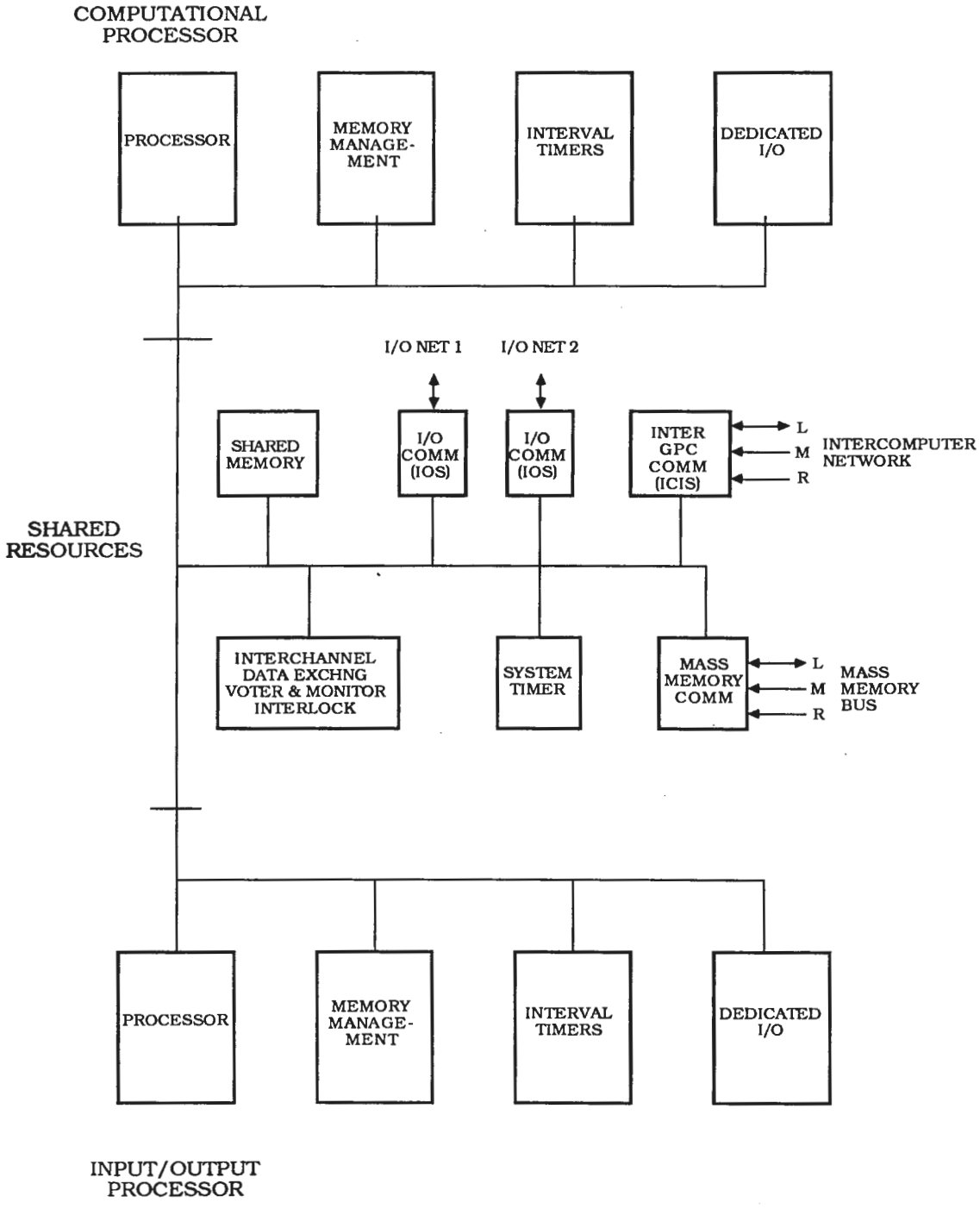



FIGURE 1.6-3. FAULT TOLERANT PROCESSOR - FUNCTIONS

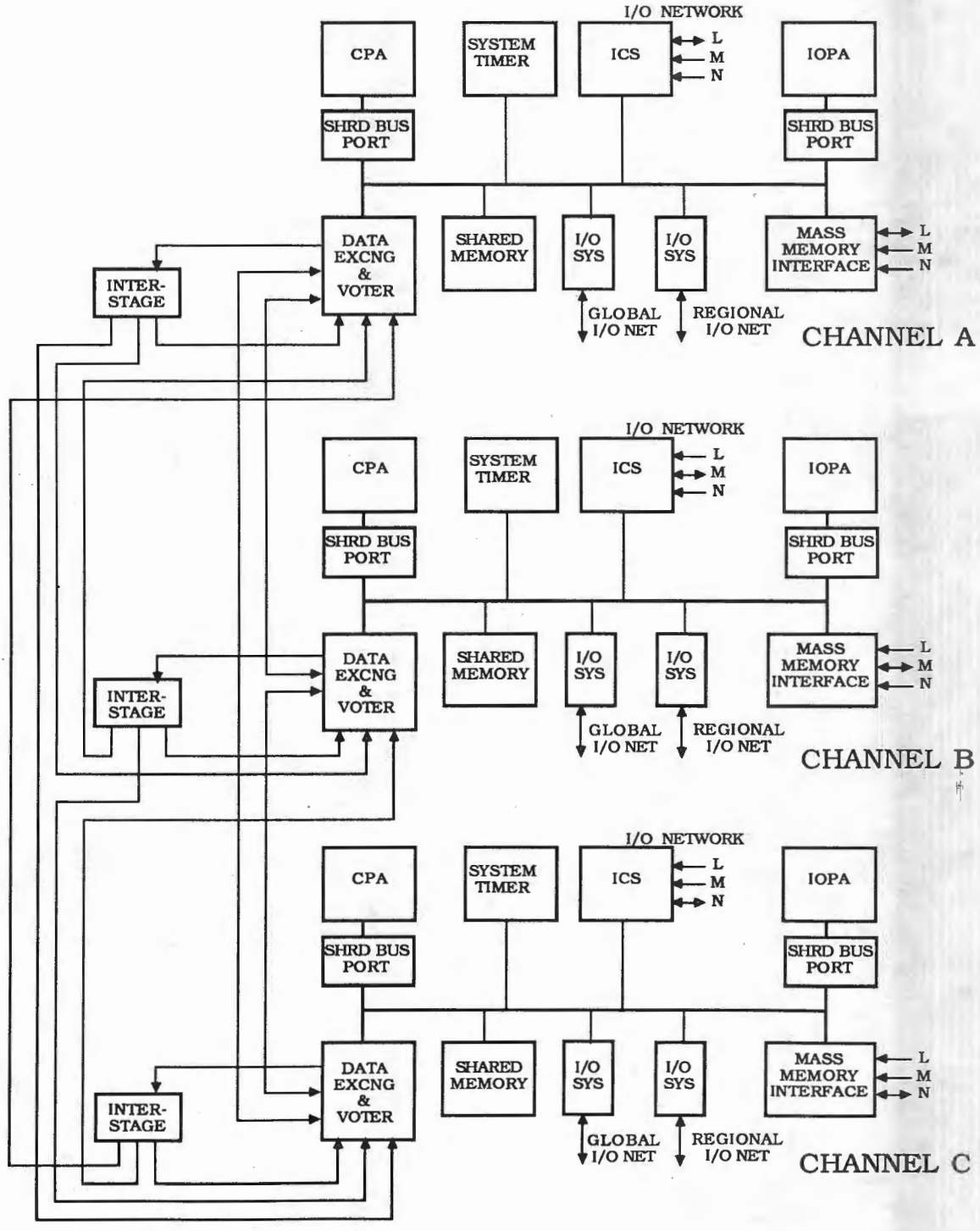



FIGURE 1.6-4. FAULT TOLERANT PROCESSOR - PHYSICAL VIEW

## 2. FAULT INJECTION SIMULATION AND TEST METHODS

Simulated faults are needed to demonstrate that an FT system response to faults is appropriate. Measurements are needed to demonstrate faulted behavior of the systems. These measurements also give added confidence that the predicted reliability, determined from coverage and component failure modeling, is correct. Furthermore, these tests validate that a system's performance under faulted conditions meets the need for safety and reliability of aircraft systems.

A typical reliability requirement for life critical operations is $10^{-9}$ per hour of operation. This translates into approximately 10 failures per million years per unit. This is about five orders of magnitude greater reliability than today's non-redundant systems. More importantly, for the purpose of demonstrating the reliability, life cycle testing will not be possible without simulation of the effects of faults.

Life cycle testing offers realism, but the rate of component failures is about one per 1,000 hours of operation per system. Fault insertion speeds up the rate at which synthetic faults occur. The use of inserted faults is needed to allow testing to cover all possible fault types and locations. For example, a small board consisting of 50 packages, each having 20 pins, has a possibility of 1,000 different pin-level (hardware) faults. This is without considering time dependence or sequence faults. Software faults must also be considered as the software becomes more complex. The number of potential software faults for a small operating system is also in the tens of thousands and time dependent as well.

Much work has been done on simulation and test methodology for insertion of faults in digital systems. (For details refer to McGough and Swern 1981; Benson, Mulcare, and Larsen 1987; Padilla 1988; Baker, Mangum, and Scheper 1988; and Migneault 1988.) A typical fault injection test setup involves the elements shown in figure 2.0-1. The faulted behavior is measured comparing faulted processor responses against those of an unfaulted processor. Measurement parameters include faults that are inserted and found, faults that are inserted and not found, and no-fault situations that are identified as faults.

A great deal of work is still needed in this area. The variation in test approaches ranges from entirely simulated software faults to injecting faults on real hardware operating under simulated flight conditions. The modeling accuracy, assumptions required, and interpretation of results also varies among the different methods. NASA, more than any agency, has developed the tools and techniques for fault insertion and testing to validate FT systems reliability.

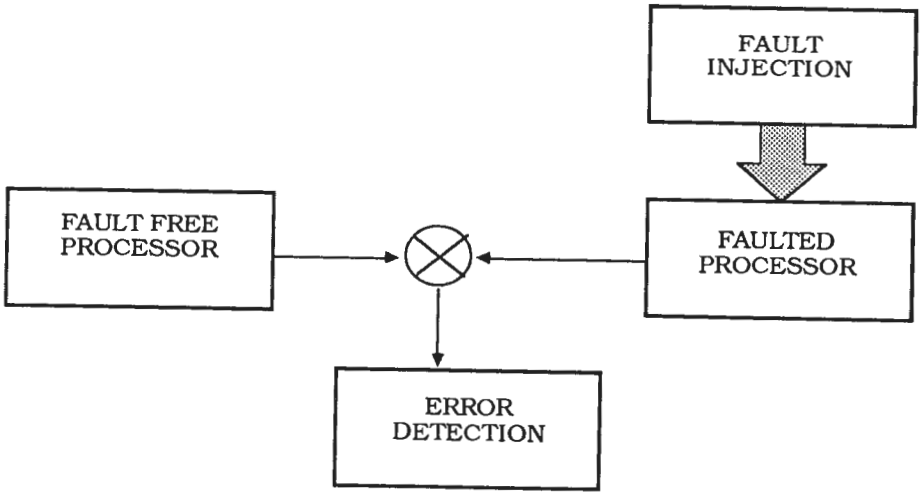



FIGURE 2.0-1. FAULTED PROCESSOR PERFORMANCE MEASUREMENT

## 2.1 NASA FT System Research Summary

Validation is the process of demonstrating that a system meets its specifications. For high-reliability systems, the specifications contain reliability requirements that necessitate operation under fault conditions. To demonstrate these extreme conditions, prediction methods must be used to determine nominal behavior of fault detection, isolation, and recovery schemes. Complementary to these methods are experimental methods, such as fault insertion, which are well suited to support modeling and analysis. Experimental methods are extremely useful when too many assumptions are needed in the analytical models or when analytical models have too much uncertainty.

Recent NASA research has emphasized fault injection methods for testing fault tolerant systems. The establishment of the Avionics Integration Research Laboratory (AIRLAB) at the NASA Langley Research Center provides a capability for government, industry, and research oriented institutions to investigate the faulted behavior of digital systems. NASA Ames is also involved in research in this area through efforts at the Reconfigurable Digital Flight Control System (RDFCS) Facility. Efforts recently completed at NASA Ames include development of the Fault Insertion and Instrumentation System (FIIS) to measure behavior of FCSs. (This work was sponsored by the Federal Aviation Administration (FAA) Technical Center.)

Several approaches have been developed for application throughout the system life cycle from design to deployment (Carter et al. 1989). Much work remains to be accomplished. Some of the methods are at different stages of development ranging from initial investigations to nearly final form (well understood and

ready for engineering application). A summary of methods and application areas is given in table 2.1-1.

TABLE 2.1-1. VALIDATION AREAS FOR THE DESIGN PROCESS

| DEVELOPMENT LEVEL | ABSTRACT | | CONCRETE |
|---|---|---|---|
| | DESIGN PROOFS | ANALYSES | TESTS |
| Architecture | Prove Architecture Against Requirements | Reliability and Error Rate Markov Models | Design Review High Level Simulations |
| Implementation | Prove Implementation Against Architecture | Fault Tree Analysis | Simulation and Emulation of Hardware |
| Realization | Prove Realization Against Implementation | Failure Modes and Effects Analysis | Support Assumptions from Analysis, Fault Insertion |

### 2.1.1 Architectural Design Methods

In the requirements and specifications phase, the tools needed include simulation and modeling for the hardware and software. These tools provide data for studies and analysis including conceptual design, hardware/software trade studies, architectural design options, and "what if" parameter bounds.

### 2.1.2 Detail Design Methods

In the design phase, little hardware is available for testing. During this time the architectures of redundant processors, operating system software, and some applications are partially designed. Many portions are in process. The tools needed for this phase include emulation and simulation of the software and processor architecture.

### 2.1.3 Developmental Phase Methods

In the development phase, the software is better defined and available for testing. Hardware is becoming available in prototypes. Everything may change as this phase continues. Tools needed for this phase include in-depth emulation and simulation for software, and prototype evaluation tools for hardware items such as error detection and communications.

### 2.1.4 Systems Implementation and Evaluation Methods

In the implementation and evaluation phase the first items are completed. The tools needed include methods for inserting faults directly into the system hardware, and methods to exercise the special capabilities built into the hardware.

## 2.2 Fault Insertion Methods

Four methods for insertion of faults are in use. The advantages and disadvantages of each are shown in table 2.2-1. These four methods (Czeck, Siewiorek and Segall 1989) are based upon:

- Software simulation using fault insertion by code modifications or special functions of the simulation engine.
- Hardware emulation using hardware representative of the system under test, such as engineering prototypes, for study.
- Fault emulation imitating fault behavior through software control of the hardware or special capabilities built into the hardware.
- Physical fault insertion inducing faults through special hardware directly into the actual system under test.

TABLE 2.2-1. FAULT INSERTION METHODS

| | FAULT INSERTION METHODS | | | |
|---|---|---|---|---|
| | SOFTWARE SIMULATION | HARDWARE EMULATION (BREADBOARD) | FAULT EMULATION | PHYSICAL FAULT INSERTION |
| Advantages | Access to system at any level of detail.<br><br>Fault types and control are unlimited. | Representative hardware with favorable access and monitoring. | True hardware and software in use. | True hardware and software in use. |
| Disadvantages | Simulation time explosion.<br><br>Lack of tools limit ability. | Implementation and other parameters will change with deployed system. | Fault types are limited. | VLSI limits access and monitoring points.<br><br>Task is difficult. |

These methods have been developed and used extensively for at least ten years. They are used for measuring fault coverage, error detection coverage, and other fault tolerance performance parameters. Measurement of these parameters requires special methods such as fault insertion.

Fault insertion has been used extensively for a number of purposes including test coverage, generation of fault dictionaries, study of error detection, and system evaluation. Table 2.2-2 summarizes these studies.

TABLE 2.2-2. FAULT INSERTION EXAMPLES

| SOURCE | TARGET | METHOD/LEVEL | GOAL |
|---|---|---|---|
| Avizienis and Rennels '72 | JPL Star breadboard | Permanent and Transient/Pin (Gate) Level | Estimate of Detection, Recovery Procedures, and Coverage |
| Goetz '72 | ESS Microstore | Simulation/Gate Level | Detection and Coverage Measure |
| Courtois '79 | MC 6800 | Simulation/Opcode (RT) Level | Detection Time |
| Decouty Michel and Wagner '80 | GORDINI: Fault Tolerant micro | Physical/Pin (Gate and RT) Level | Tool and Methodology Development |
| Kurlack and Chobot '81 | GE MCP-701 CPU | Physical Faults with FMECA analysis | Evaluation of Watchdog Timer |
| McGough and Swern '81 | Bendix Simplex BDX-930 | Simulation/Gate and Pin Level | Coverage Measurement |
| Lala '83 | FTMP Engineering Model | Physical/Pin (Gate) Level | System and Coverage Evaluation |
| Yang, York, & Siewiorek '85 | iAPX 432 | Fault Emulation/ Memory Words (RT) | Coverage Management |
| Schuette et al. '86 | MC 68000 | Transient, Physical/ Bus (RT) Level | Evaluation of Error Detection Techniques |
| Czeck et al. '87 | FTMP Engineering Model | Fault Emulation/ RT Level | Methodology Study |
| Finelli '87 | FTMP Engineering Model | Permanent Physical/ Gate Level | Fault Recovery Distributions |

By far the two most common methods involve computer simulation of hardware faults and physical fault insertion. Fault simulation has been used at all levels for the fault models discussed in section 1.5. Physical fault insertion has been used to determine fault coverage of test programs, fault latency, and fault detection efficiency. Experimental agreement among the different techniques provides reasonable confirmation of the validity of these test methods. However, there is considerable difference in fault detection coverage between tests with models at the gate level and pin level of detail. The selection of injection points and the number of points tested also affects the measurements.

Since system fault detection is a critical parameter, NASA initiated a pilot study to investigate the methodology for measuring the fault latency of digital computers. The methodology consisted of simulating a 1,000 gate equivalent computer in a host Cyber 173 computer. The simulated computer was a paper design and was referred to as the hypothetical machine. This hypothetical machine was simulated at the gate level. Actually two copies of the hypothetical machine executed identical copies of code in synchronism although one machine received a fault at the onset of the simulation. Detection or non-detection was determined after the non-faulted processor completed its execution. At that time, the computational results of the two machines were compared bit-for-bit; any difference constituted detection. If no detection occurred, the code input variables were randomly altered, and the programs were repeated for the same fault. This scheme was repeated for up to eight executions for the same fault. If a detection occurred in eight repetitions or fewer, or no detection occurred after eight repetitions, then a new trial began. This over-all process was repeated for up to 211 faults randomly selected throughout the circuitry of the computer. The 211 faults were selected as a function of the piece-part failure rates and distributed across the nodes of the gates. The latency time (i.e., time to detect) is expressed in terms of code executions or repetitions.

Results of the pilot study were both surprising and disturbing. Using six different programs ranging from simple fetch and store to a very complex linear convergence scheme, the pilot study indicated that only 50 percent of the induced faults were detected after eight repetitions for each of the six programs. Figure 2.2-1 provides a typical picture of these results.

Studies conducted by McGough and Swern (1983) on the Bendix 930 FTP showed quite significant differences between gate-level and pin-level fault injection. Preliminary results obtained from gate-level fault simulation, using a hypothetical simple computer, showed that only 50 percent of the induced faults propagate to an output port where detection is possible. This pessimistic and surprising result prompted the design and implementation of several follow-on studies using realistic avionic computers.

The same set of experiments were applied to the minicomputer with the addition of realistic flight control code. The results were once again surprising: the percentage of undetected faults was about the same for all the programs simulated. Only about 50 percent of the injected faults were detected when different instruction sets, executing in two different machines (one hypothetical and one real), were used. More complex code tends to alter the shape of the

histogram, so that almost all the faults are detected in the first execution of the code. The latency time decreases somewhat with increased code complexity, but not the percentage of faults detected.

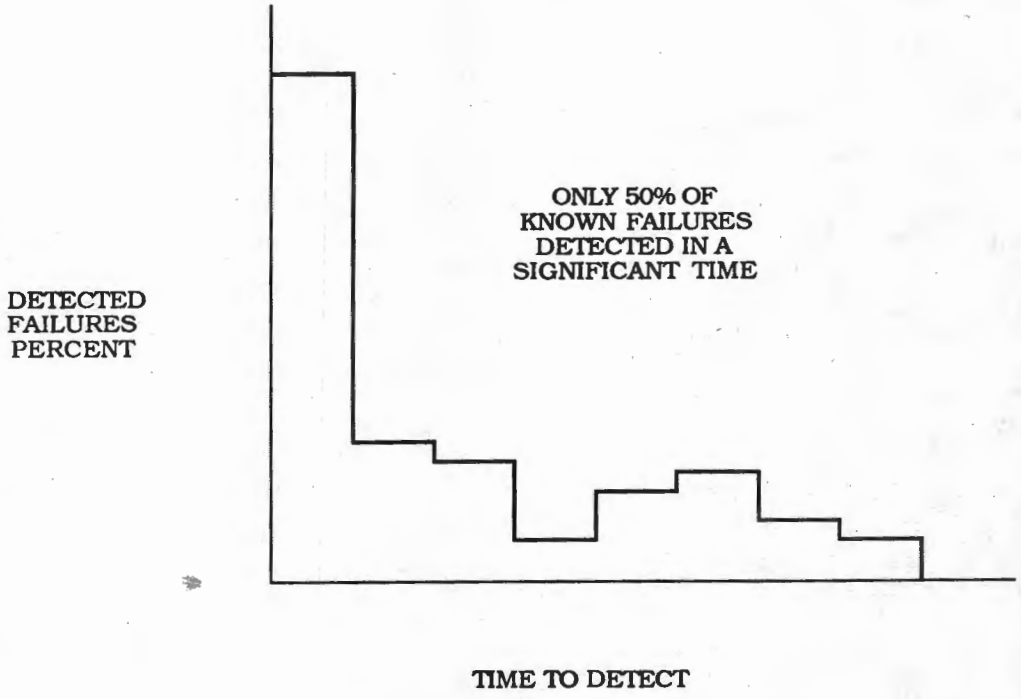



FIGURE 2.2-1. FAULT LATENCY DATA

More recently, experiments were conducted by the FAA Technical Center at NASA Ames (Benson, Mulcare, and Larsen 1987) on a Flight Control computer operating a simulated closed loop flight control. The computer configuration used was two CAP-6 processors in a dual configuration. This configuration is part of the dual-dual FCS used for flight control of the Lockheed L-1011 Tristar. An enhanced fault injection and data analysis system, based upon the CSDL Fault Tolerant Multiprocessor (FTMP) test setup, was used to control the experimental fault injection, and to collect and analyze the data. This computer was a much larger processor than that used for the Bendix study. The FCS program had 11,000 words, and the FCS computer had approximately 1,200 pins. Faults were injected on pins stuck at zero or one; approximately 2,500 faults were injected. The finding was that a much higher percentage of faults were detected; nearly 96.5 percent of all the injected faults were detected. These results were consistent with the earlier CSDL measurements on the FTMP. Additional discussion of these experiments is in section 2.6.3.

The following subsections discuss the four fault injection and test methods, and provide further examples and discussion of these issues.

## 2.3 Software Simulation Fault Insertion Methodology

### 2.3.1 Method

For this method of testing, the system digital logic circuits are simulated at the gate level-of-detail using high speed, specially designed, fault simulations. These simulations are like those for performance analysis, but stripped down to provide higher speed and specific outputs for fault detection and analysis. Logic levels are limited to zero or one. Little or no information is available for prediction of the system performance, such as speed and throughput.

Gate level-of-detail logic simulators were among the first tools for investigating the behavior of fault tolerant processors. Starting in 1977, NASA Langley initiated pilot studies on methodology for dealing with faults in digital systems. These efforts utilized a Cyber 173 to model a 1,000 gate equivalent processor. This was a paper design for a hypothetical machine. Following the pilot studies, a Gate Logic Software Simulator (GLOSS) was implemented to study the behavior of the Bendix 630 miniprocessor, a 5,000 gate machine. Subsequent to the Bendix study, it was realized that a more general capability could be useful for other processors. Several interim and improved versions of the GLOSS were developed: S-GLOSS by Stevens Institute of Technology; B-GLOSS by Bendix; and IGGLOSS, an improved version developed at Langley.

The current version being developed by Research Triangle Institute is called the Generalized Gate-Level Logic System Simulator (GGLOSS). In this version detailed gate-level simulations are run with faulted and unfaulted processors. The results are checked by comparison monitoring at critical outputs in the simulations.

Simulations may be run with faults injected at the gate level, pin level, or component level of detail. Since the entire processor is modeled at the gate level, faults may be injected at the opcode level. The gate faults are injected by holding the output node at zero or one. Pin-level faults are injected by holding the pin or terminal at zero or one.

### 2.3.2 Application

This method is most useful in early stages of development where no hardware is available, or for small enough computers so that a complete gate-level simulation may be processed.

### 2.3.3 Example Results

Researchers at NASA Langley began to study development of highly reliable prediction programs (B-GLOSS) for application to ultrareliable fault tolerant digital systems in the 1970s. One of the first efforts was modeling and analysis of the earlier Bendix 930 pilot study results. These new efforts were designed to test the validity of the pin-level injection result obtained from physical fault injection. It was desired to confirm that similar results could be obtained from a real processor executing practical software.

The pilot study results were tested in three phases using simulations of the Bendix 930 miniprocessor, a 5,000 gate equivalent computer. In the first phase, six application programs were coded using the primitive instruction set of a hypothetical machine: load, store, add, subtract, and branch. The next phase allowed the six programs to be coded using the rich instruction set of the Bendix 930. Finally, comparison monitoring detection was measured for an FCS code in lieu of the six programs.

The surprising outcome of these simulations is typified in figure 2.3-1. The results of all six programs follow the same trends. The percentage of undetected faults is about the same for all the programs, instruction sets, and two different machines: about 50 percent were undetected. For the more complex code, the shape of the histogram changes: most of the faults are detected in the first execution of the programs. The latency time decreases with increasing code complexity, but not the percentage detected.

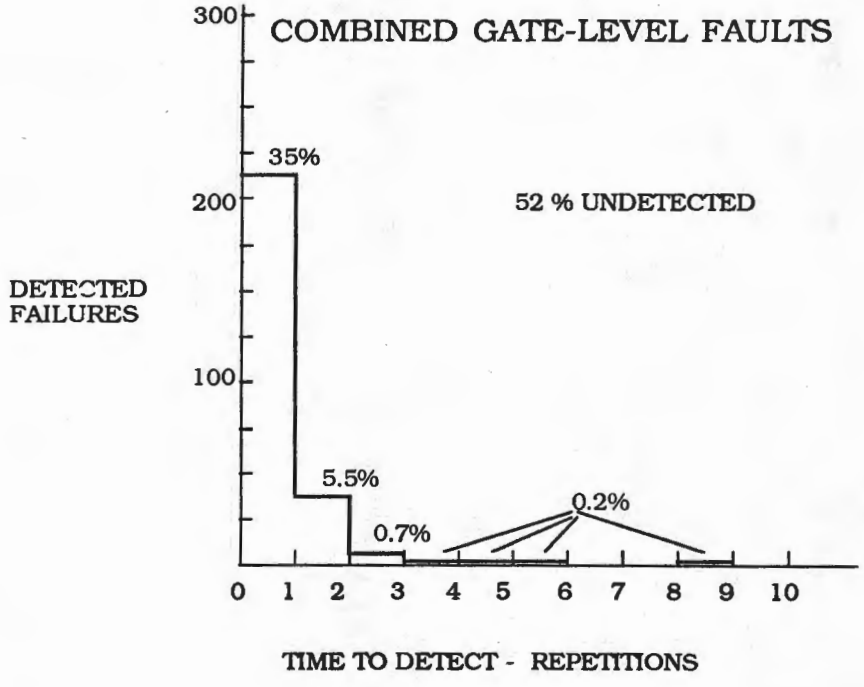



FIGURE 2.3-1. FAULT LATENCY DISTRIBUTION - GATE LEVEL FAULTS

The next set of experiments with pin-level fault injection produced more disturbing results. When the simulations were run with faults injected at pin level, but using the same programs and gate descriptions, the results shown in figure 2.3-2 were obtained. The notable difference is that the level of detection increases to about 62 percent of the injected faults.

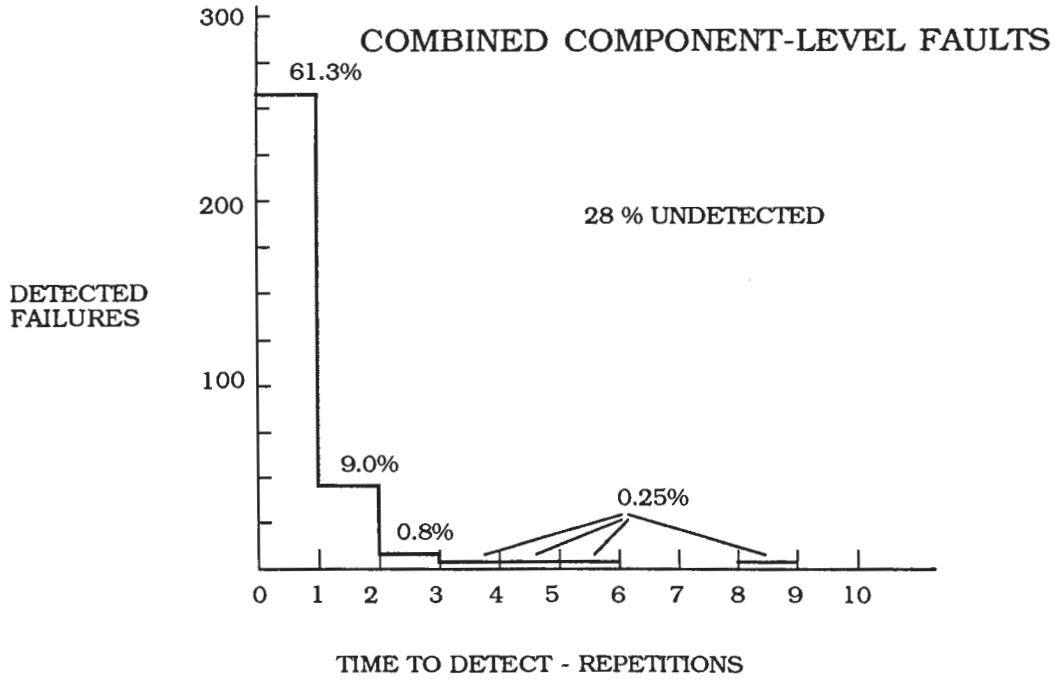
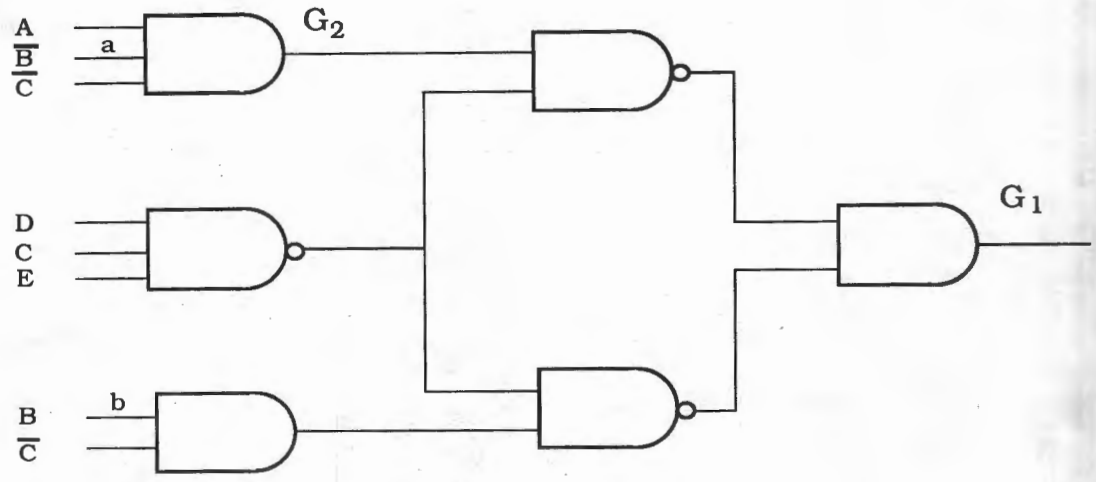



FIGURE 2.3-2. FAULT LATENCY DISTRIBUTION - COMPONENT PIN LEVEL FAULTS

Investigation of the difference between faults at the gate level and faults at the pin level determined that there is a class of faults that can never have an effect on the system and will never be detected. An example of these faults is a stuck-at fault located at a unused flip-flop output. The study found 16 percent of the faults were injected at such "don't care" locations.

An example of digital logic, shown in figure 2.3-3, may clarify the difference between pin and gate faults. The eight device terminal pins are A, $\overline{A}$, B, $\overline{B}$, C, $\overline{C}$, D, and E. Gate faults may be injected at any of the six gate output nodes, but only one is a pin. The "don't care" pin faults are line-a S-A-1. If line a is stuck at one, then line-b S-A-0 cannot be detected.

As an extension to the pilot project, the GLOSS simulation method was applied to a 2,000 instruction self-test program for the Bendix 930 processor. While the simulation exercised the program, faults were injected at the gate level and subsequently in another experiment at the pin level. The design goal of the self-test program was five percent coverage. For the experiments run with gate-level fault injection, the undetected faults were about eight percent of the detectable injected faults. With faults injected at the pin level, only two percent of the detectable injected faults were undetected. These data have been corrected for the 16 percent of "don't care" faults in this system.




THE SET OF TESTS $A\bar{B}CD$, $\overline{ABC}$, ABCDE, BCD, $AB\overline{CD}E$

CAN DETECT ANY DETECTABLE FAULT IN THE CIRCUIT $G_1$

NOTE THAT:
1. FAULTS ON PIN b ARE DETECTED BY $AB\bar{C}$
2. BECAUSE OF REDUNDANCY, THE FAULT PIN a: S-A-1 CANNOT BE DETECTED
3. IF THE FAULT PIN a: S-A-1 EXISTS, THEN FAULT PIN b: S-A-0 CANNOT BE DETECTED.

FIGURE 2.3-3. LOGIC DIAGRAM ILLUSTRATING DETECTABLE FAULTS

Work is still underway on the GGLOSS simulator and validation plans for it were recently developed (Bavuso and Miner 1989). GGLOSS has the capability to simulate S-A-1 or S-A-0 at any point in the circuit. Several innovative methods are used to eliminate the overhead in the usual event driven simulation. Sequential and combinatorial circuit elements can both be modeled. The combination of features gives GGLOSS the capability to simulate approximately $10^6$ gate evaluations per MicroVAX II Central Processing Unit (CPU) second, while allowing users to monitor the effects of simulated faults.

### 2.3.4 Limitations

The types and numbers of faults are limited to logic levels in most implementations. Faults must be modeled as gate nodes or logic lines stuck at zero or stuck at one. Faults are inserted one at a time in an order determined by an experiment plan. Logic lines may be held for several cycles or inserted momentarily.

The selection of faulted lines or devices is often at random. This imposes a limitation on the measured coverage of possible faults. A great many "don't care" faults may be reported. Recent workers emphasize measurement of critical faults, such as detection and recovery times for reconfiguration processes (Padilla 1988, 1989).

## 2.4 Hardware Emulation Fault Insertion Methodology

### 2.4.1 Method

This approach uses special purpose digital hardware to emulate portions of the system under development. (Frequently the system under development is still on a breadboard.) The concept is similar to the use of emulators in the design and test of software using an emulation of a future microprocessor design. The instruction set of a new processor can be emulated in code using the instruction set of an existing microprocessor. In this way operating system software can be developed and tested while the new processor is still under development.

In the fault insertion emulation approach, algorithms for fault detection and recovery can be emulated by a special purpose computer consisting of hardware and software for emulating a wide variety of digital logical elements. The operation is based upon high-speed emulation of algorithms describing the digital logic at various levels of detail down to the gate level.

The technique and its concept, depicted in figure 2.4-1, are independent of any particular hardware or software implementation. Migneault (1988) contains a detailed discussion of this technique.

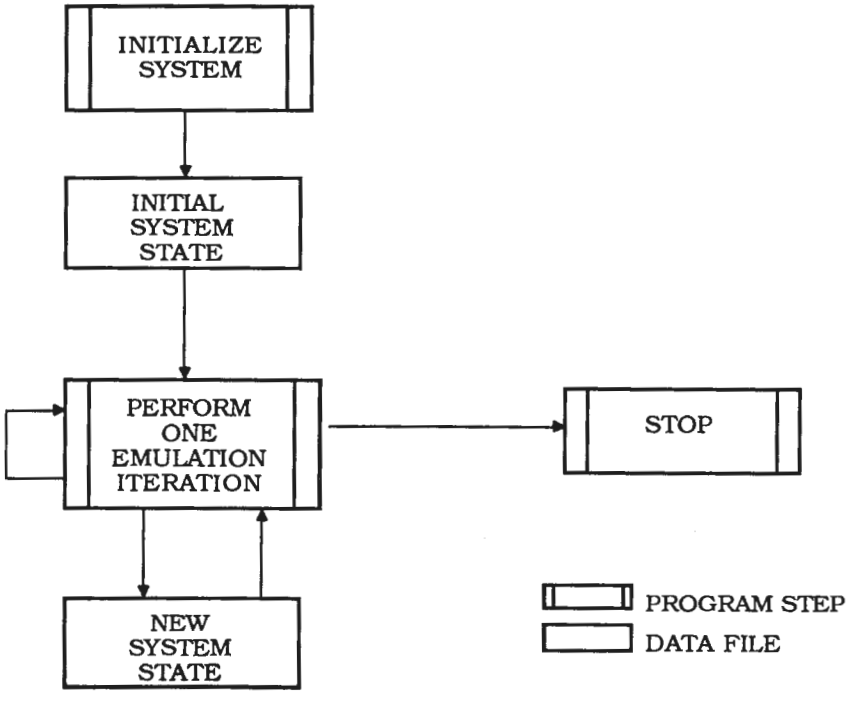



FIGURE 2.4-1. FAULT EMULATION COMPUTATION TECHNIQUE

### 2.4.2 Application

Fault tolerant digital systems are implemented by first setting reliability goals, and then designing the hardware and software systems as algorithms for fault detection and recovery. The tolerance to faults is usually in the form of redundancy components and algorithms which can detect faults and reconfigure the components in case of faults.

### 2.4.3 Example Results

This approach was developed at NASA Langley Research Center. In this approach (Bavuso and Miner 1989) an algorithm was developed to emulate any network of logic gates, flip-flops, and tristate devices. This algorithm is independent of any particular piece of hardware being emulated. A description of the target system is fed into a translator which converts the description to a form that the emulator can process. The processing of this representation of the target hardware by the software-implemented algorithm consists of the gate-level emulation of the target hardware. During emulation, faults can be injected and their effects studied.

Several unique features were incorporated in the Langley emulator system. Emulation speed is important because the target system must be run for many iterations of the fault insertion, detection, and recovery process. Memory space is necessary due to the large number of gates, flip-flops, and tristate devices in any modern digital system. The algorithm employs a general model for all types of gates (i.e., AND, OR, NAND, NOR, NOT, XOR), and a generalized model for all types of flip-flops. These general models allow for efficient use of computer memory. Time is conserved by processing, in a given cycle, those devices whose input has changed during the previous cycle.

This emulation system has a very general hybrid capability to model some parts of a system at the device level and other parts at the functional level. This capability allows the user to specify large portions of the system in a functional manner. At the same time, the user retains the capability to model other critical portions in detail, to the gate level, without the need to define every gate in the entire system.

In operation the emulator process is very straight forward. The state of the entire system at a given time consists of state descriptions of all logic devices in the network, state descriptions of all connections among devices, and a state description of the functional subsystem. The emulation must have the initial state of the entire system. Then for each time step T, the emulator calculates the output states for the next time T+1. The process continues to be driven by the series of events until the user-specified stop time is reached. The basic time step is the propagation time for input signals to propagate through a device to reach the output. For the initial emulator, the propagation time is taken as the same for all devices.

Fault insertion capabilities built into this implementation are as follows:

- The ability to insert or remove faults at user-specified times into the logic gates or ROMS.

- The ability to input to the digital logic at user-specified times from sources external to the emulation.

- The ability to output from emulated logic to external ports at times specified by the user or times driven by the system states.

As an initial experiment, a horizontally-microprogrammable computer, the Nanodata QM-1, was chosen as the host system. The emulation was coded at the microcode level to take advantage of the parallel capabilities of the host machine and to exploit the speed advantages of executing code at the most primitive level of the host computer. All processing of the hardware description and fault-injection data, as well as all post-processing of fault data, is performed on a Digital Equipment Corporation VAX 11 which is interfaced to the QM-1.

The emulation algorithm has been used to emulate a simplified model of the central processor of the Bendix BDX930, and the communications interface unit of the CSDL FTMP. Both of these system components have been the subject of other studies and comparative performance data is available to validate the emulation.

### 2.4.4 Limitations

The principal limitations of the hardware emulation technique are in speed and system modeling accuracy.

The speed of the emulation is dependent upon the size of the models and the speed of the emulation engine. The hybrid technique is important in reducing the size of the models. Speed is important both in shortening the run time and for emulation of real-time processes.

Modelling accuracy has some limitations due to the hybrid nature of the process. In this hybrid, system states are determined from the data structure and a subprogram module operating on that data structure and optionally on the gate-level network. This may not be serious, but care must be exercised in any applications to maintain consistency between the global and gate-level models.

The faults modeled are limited to pin faults, S-A-0 or S-A-1, or gate-level faults that can be defined by effects on the logic algorithms of the system under test. Conceptually this allows for a wide range of faults, but the effects of faults may be very difficult to model accurately and input to the emulation.

Propagation delay times are taken as the same for all devices. This is unrealistic and future versions of the emulator should address this parameter allowing the user to specify the delay time for each device in the gate model library.

## 2.5 Fault Emulation Fault Insertion Methodology

### 2.5.1 Method

The fault emulation technique inserts software faults, or the effects of faults, into the real system software. A special purpose processor is used to control fault insertion and monitor the system under test. Faults, or the errors manifested by faults, are inserted by triggering the error detection mechanisms or by seeding errors into memory.

Faults are injected into software controlling the system hardware or into the special capabilities built into the hardware such as communications registers, comparison-monitoring detector, configuration status registers, and sensor output registers. Possible software faults include lost or delayed messages, corrupted messages, delayed tasks, abnormal task termination, timer corruption, and system clock corruption. Injection of faults directly into system software is accomplished by transfers over the data bus of the system under test. The process is controlled by timing and monitoring of data and status information received from the data bus. Instrumentation of the effects of testing is also accomplished using the data and status information on the system data bus.

Fault emulation fault insertion offers the following features:

- Allows insertion directly into the substantial software portion of a system. These software errors can be used to examine the software and the interactions of the hardware and software.
- Is often simpler, in terms of time and effort, than hardware fault insertion.
- Is complementary in terms of functionality to hardware fault insertion and does not exclude hardware fault insertion.
- Provides a direct means for testing software implemented fault tolerant detection and recovery strategies.

Data collection and analysis is also implemented by using information on the system or processor bus. Historical data records of normal events and functions may also be collected. Then, error reports can be generated indicating exceptions and abnormal events, allowing measurement of faulted performance. Data analysis supports developing data on runs, experiments, and multiple experiments.

### 2.5.2 Application

Software fault insertion, as described in this subsection, is most useful during the system development and validation. Faults may be inserted into developmental software modules such as the communications processor. The equipment utilized is essentially a general purpose test and evaluation instrument. This concept should have application during the operation and maintenance phase as well.

2.5.3 Example Results

As an example we will discuss the implementation of the Fault Injection Automated Testing (FIAT) environment, illustrated in figure 2.5-1. FIAT was developed by Carnegie Mellon University under a NASA Langley grant. This system is described in Czeck, Siewiorek, and Segall 1989. The FIAT hardware consists of four International Business Machines (IBM) personal computer (PC) remote terminals (RTs) connected via a 10 Megabit token ring. This hardware provides the operating environment for FIAT, without limiting the generality of the system. The software structure consists of two modules: (1) the Fault Injection Manager (FIM) for experiment control and data analysis, and (2) the Fault Injection Receptor (FIRE) for data collection and processing.

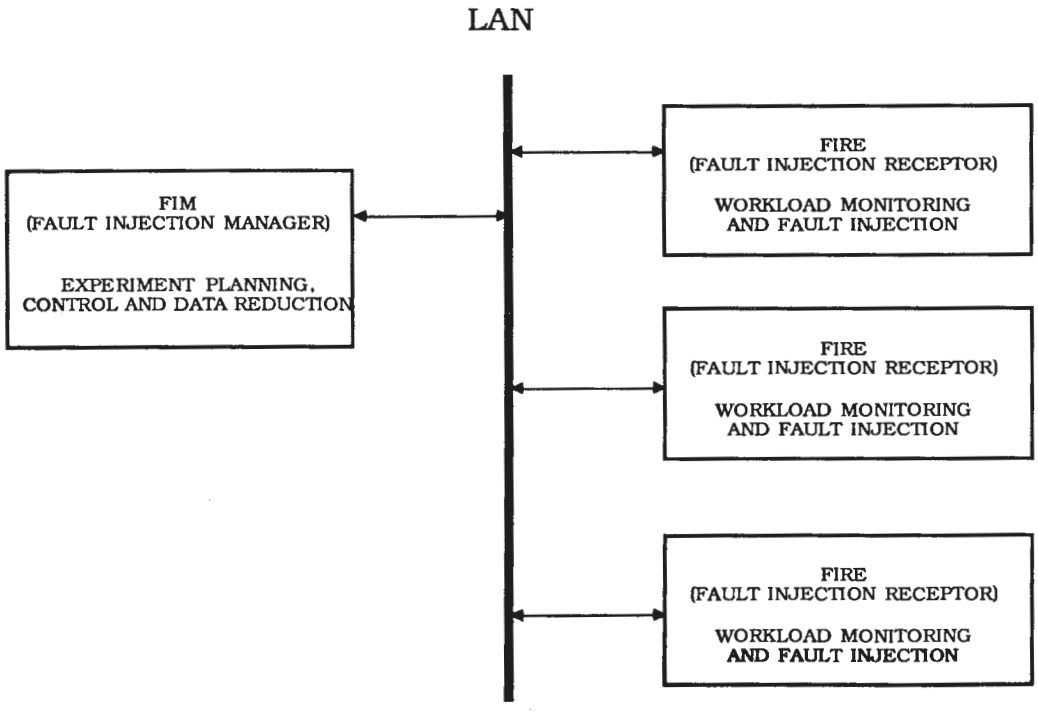



FIGURE 2.5-1. FIAT FAULT INSERTION AND EXPERIMENT MONITOR

The FIM software, shown in figure 2.5-2, processes users' programs and experiment definition to control the injection of faults and collection of data from the system under test. The experiment interface, illustrated in figure 2.5-3, has four modes of processing:

Library Preparation - Assists in preparation of task development, image generation, and attribute extraction. The fault class library groups domains and methods for selection of the fault insertion method during experiment definition.

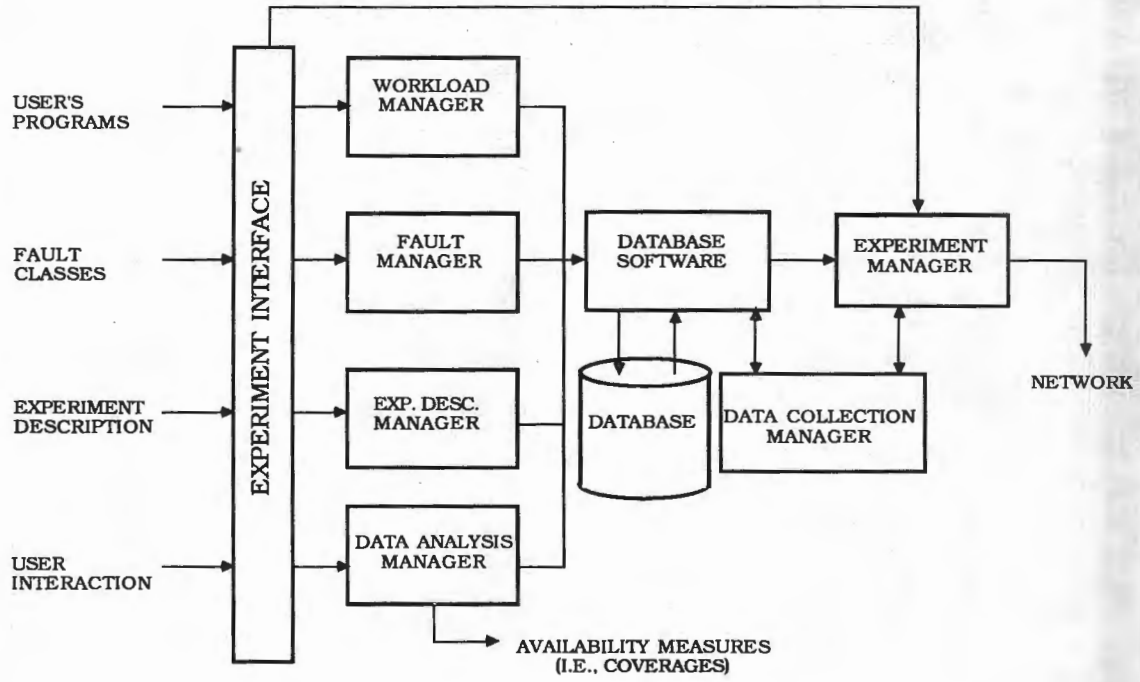



FIGURE 2.5-2. FAULT INSERTION MONITOR SOFTWARE

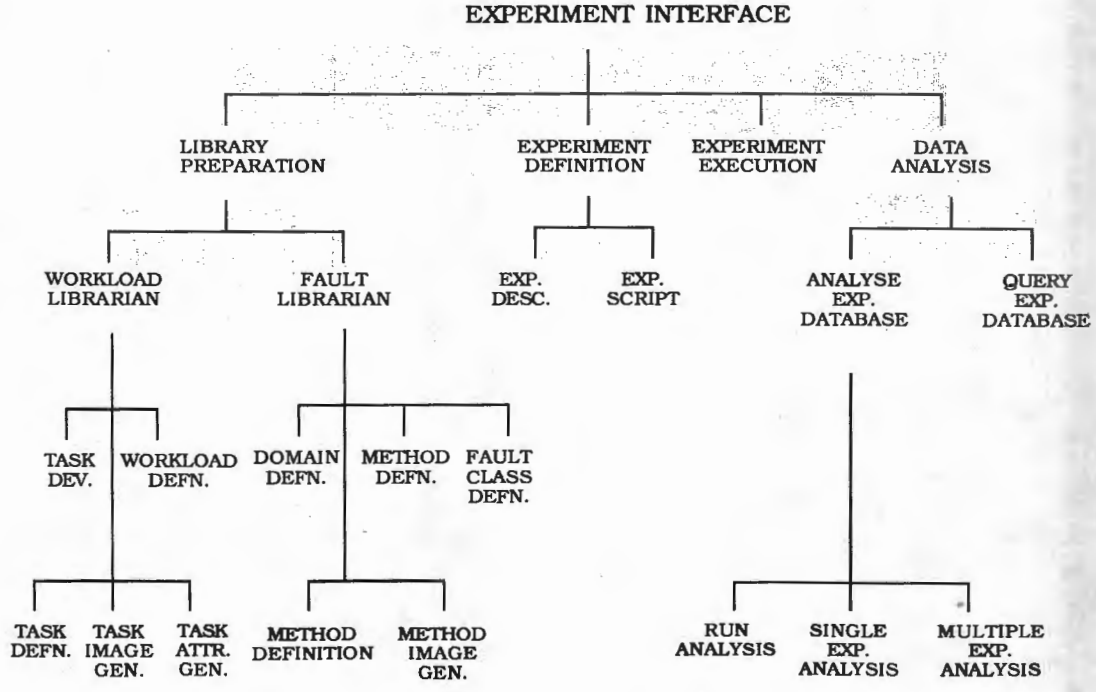



FIGURE 2.5-3. FIAT EXPERIMENT INTERFACE TREE

- Experiment Definition - Assists in developing an experiment description and in script generation.

- Experiment Execution - Utilizes the prepared scripts and executes the sequence of fault injections.

- Data Analysis - Utilizes the prepared experiment definition and provides two types of analysis: (1) user defined queries using Informex Software Query Language (SQL), and (2) generic FIAT data analysis routines.

The data collection and monitoring operate via the FIRE modules shown in figure 2.5-4. The function of each module is as follows:

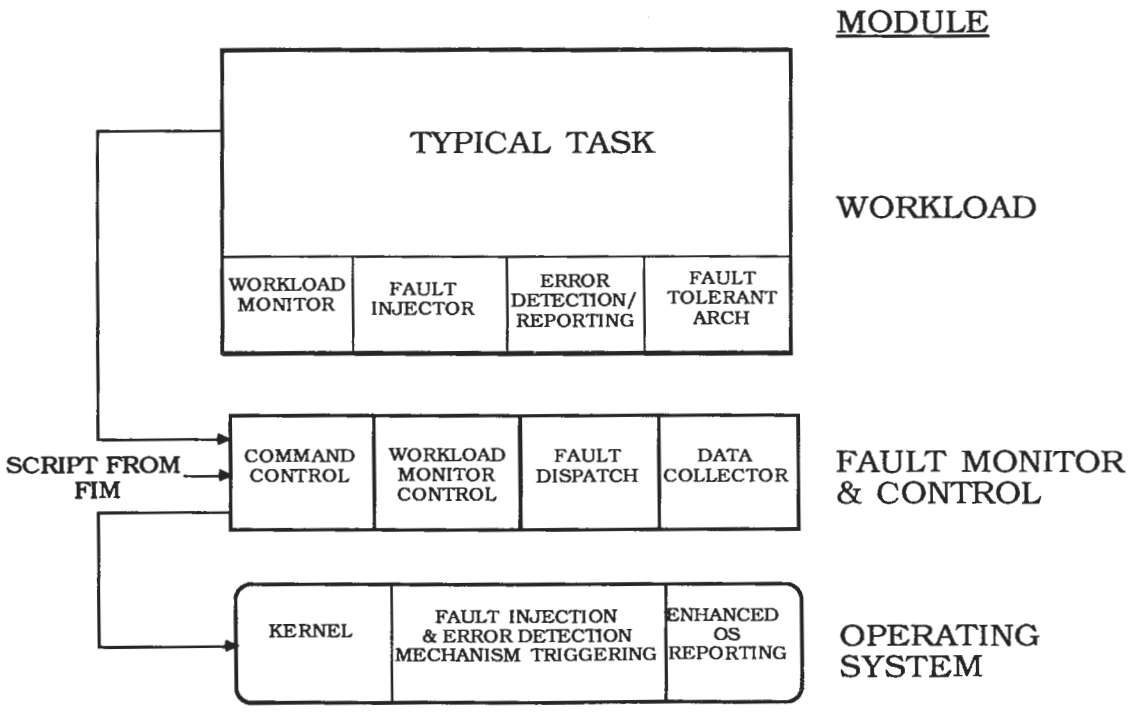



FIGURE 2.5-4. DETAILED VIEW OF FAULT INJECTION RECEPTOR SOFTWARE

- Command Controller - Stores and organizes commands from the experiment manager in a command queue and synchronizes the commands, such as task starting and fault insertion timing within the work load control flow.

- Workload Monitor Controller - Provides the functions of monitoring and control of the workload, task control, communications, and synchronization.

- Fault Dispatcher - Manage fault insertion of workload tasks, communications, and the operating system. Operating system faults include memory

and register faults and triggering of existing hardware or software error detection mechanisms.

- Data Collection - Transfers records generated by the monitor and error report scripted commands to a local experiment database storage.

As an example of the use of FIAT, a distributed checkpointing system was implemented using two FIRE machines. The block diagram, figure 2.5-5, consists of a two-computer system. The primary computer informs the secondary computer of task initiation and the time frame for the next task initiation. The primary machine executes the task and the secondary machine waits for the next interaction. If the time between interactions exceeds the time frame for completion (indicating primary failure), the secondary then initiates a recovery process and becomes the primary. If the primary detects that no secondary exists (indicating secondary failure), it creates a new secondary.

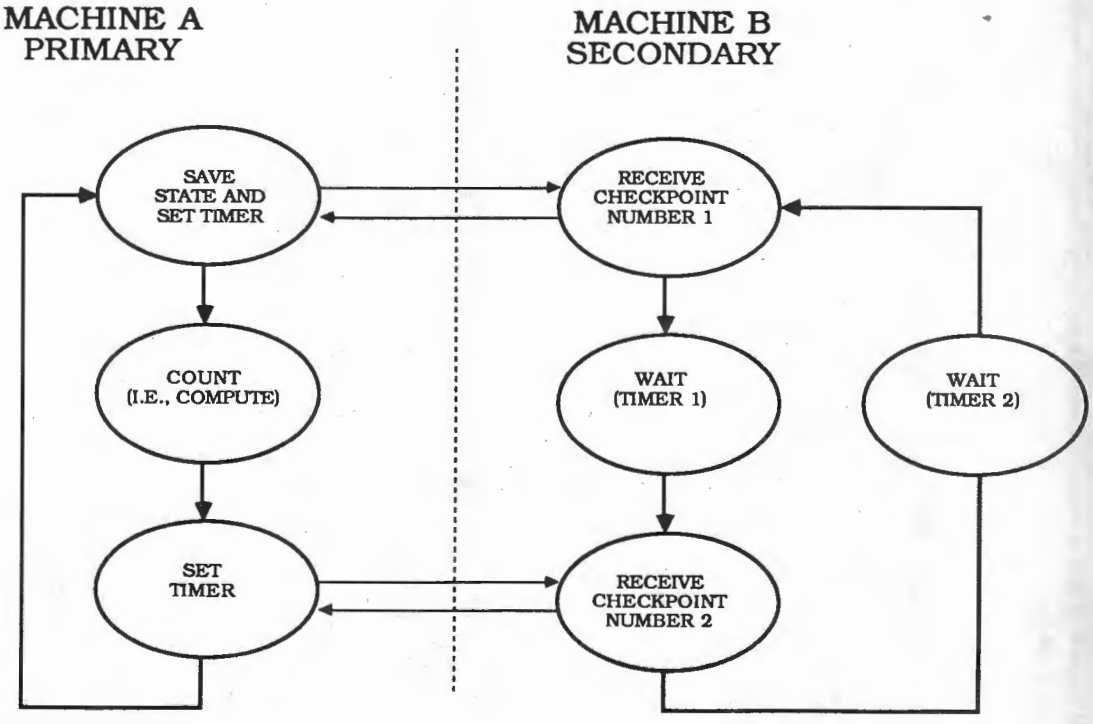



FIGURE 2.5-5. REAL-TIME CHECKPOINTING SYSTEM

Experimental results of inserting faults into the checkpointing system are shown in table 2.5-1. The faults considered were bus faults having double bit compensating errors. These double errors would not be detected by hardware parity and must be detected in software. Note the large latency time and that one-quarter of the inserted faults were not detected. To illustrate how fault tolerance strategies may be compared, the checkpoint system was improved by

TABLE 2.5-1. ERROR DETECTION STATISTICS - CHECKPOINTING ONLY

```
METHOD:  DETECTION STATISTICS

Experiment = 1a
Detection Statistics:

Number of Fault Injections ... 206
Number of Fault Detection  ... 153
Detection Coverage = 74.271845%

Average Detection Time = 4.219567 Seconds
Minimum Detection Time = 1.406250 Seconds
Maximum Detection Time = 6.218750 Seconds
```

TABLE 2.5-2. ERROR DETECTION STATISTICS - CHECKPOINTING AND CHECKSUMS

```
METHOD:  DETECTION STATISTICS

Experiment = 1b
Detection Statistics:

Number of Fault Injections ... 91
Number of Fault Detection  ... 86
Detection Coverage = 94.505495%

Average Detection Time = 3.102834 Seconds
Minimum Detection Time = 1.203125 Seconds
Maximum Detection Time = 4.203125 Seconds

END OF METHOD:  Detection Statistics
```

adding double checksum comparison to the primary and secondary machines. A checksum is appended to code and data blocks during compilation. This code is regenerated when reading blocks from memory during run-time and is compared with the code determined at compilation to check for errors. Hence the detection methods are timeouts for the first experiment and checksums for the second. Improvements in the results may be seen from table 2.5-2. Note the increased fault detection probability and the decreased detection time.

### 2.5.4 Limitations

Fault emulation fault insertion has limitations primarily in its inability to force low level errors, such as gate output S-A-0 or S-A-1. These limitations can often be worked around by injecting faults or errors at another location. Since designers are interested in the behavior of the entire system this may not be a serious limitation.

## 2.6 Physical Fault Insertion Methodology

### 2.6.1 Method

In this technique faults are actually inserted into real hardware by computer controlled switches holding the voltage levels high or low on individual wires at connector pins of selected devices. Because only the device terminals are accessible, these faults are usually pin-level S-A-0 or S-A-1 faults. Some testing has been performed using inverted levels (i.e., change a zero level to one and conversely). The existence of faults is detected by the real hardware comparator monitoring systems. Measurements include the fault insertion time and detection time. Statistics are collected on the number of faults inserted, detected, detected in error, and not detected.

These experiments can be run at the gate, pin, or resistor or other electronic component. Since real system hardware is used for testing, the common method of fault injection is at the pin level of logic circuits including gates; flip-flops; memory clips; processors; and Small Scale Integration (SSI), Large Scale Integration (LSI), and VLSI components.

### 2.6.2 Application

Physical fault insertion is used for validation of completed hardware/software systems. This method is a validation tool. Modeling and simulations will have been used prior to final implementation. Those tools will have provided predictions and reliability estimates. A final level of testing using faults injected on real hardware provides the realism, and minimizes the assumptions and modeling errors that may exist in the other techniques.

This technique is most useful during full scale development. Some applications of this technique have been made using development hardware of portions of the system, such as the communications interfaces between multiprocessors.

2.6.3 Example Results

Physical fault injection has been used for several systems including the Bendix 930, the CSDL FTMP, and more recently on a dual-dual FCS for the Lockheed L-1011 Tristar. This system used four Rockwell/Collins CAPS-6 digital computers. Figure 2.6-1 illustrates the experimental setup for data acquisition experiments (Benson, Mulcare, and Larsen 1987). The additions from prior experiments include an external clock and an associated DR11C interrupt so that fault detection is quickly reported to the PDP-11 through the DR11C interrupt. Considerable software modifications were made to analyze, reduce, and present the data for reports. Prior efforts injected faults on the pins of the CSDL FTMP (references 4, 6, and 7, from Benson, Mulcare, and Larsen 1987) system which also uses the CAP-6 processors. Investigations into this methodology are in progress. Figure 2.6-2 shows the setup for data acquisition on this current effort on the CSDL FTMP at NASA Langley.

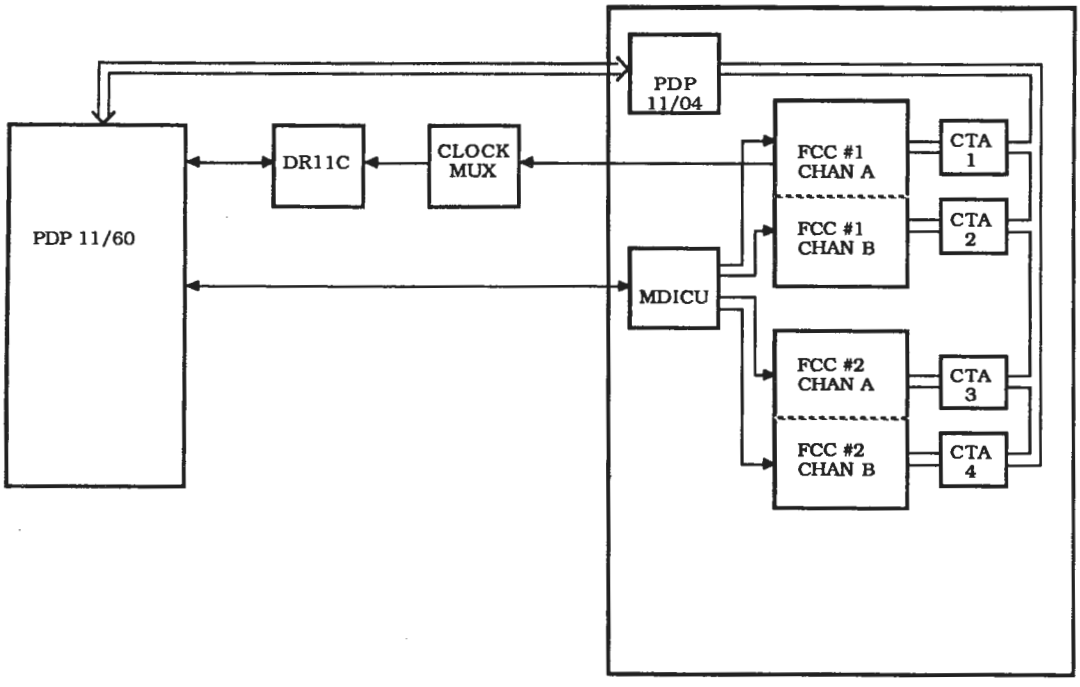



FIGURE 2.6-1. FAULT INSERTION AND INSTRUMENTATION SYSTEM (FIIS)

Both of these systems share common features in fault insertion circuitry (FI). The FI is connected to the UNIBUS in a manner similar to the Collins Test Adaptor (CTA) shown on figure 2.6-1.

The FI provides an interface between the UNIBUS and the system components under test. The fault injection is accomplished in the FI by signals sent to Field Effect Transistors (FETs) inserted between the socket and device, as illustrated in figure 2.6-3. These FETs also allow monitoring of the signals at device

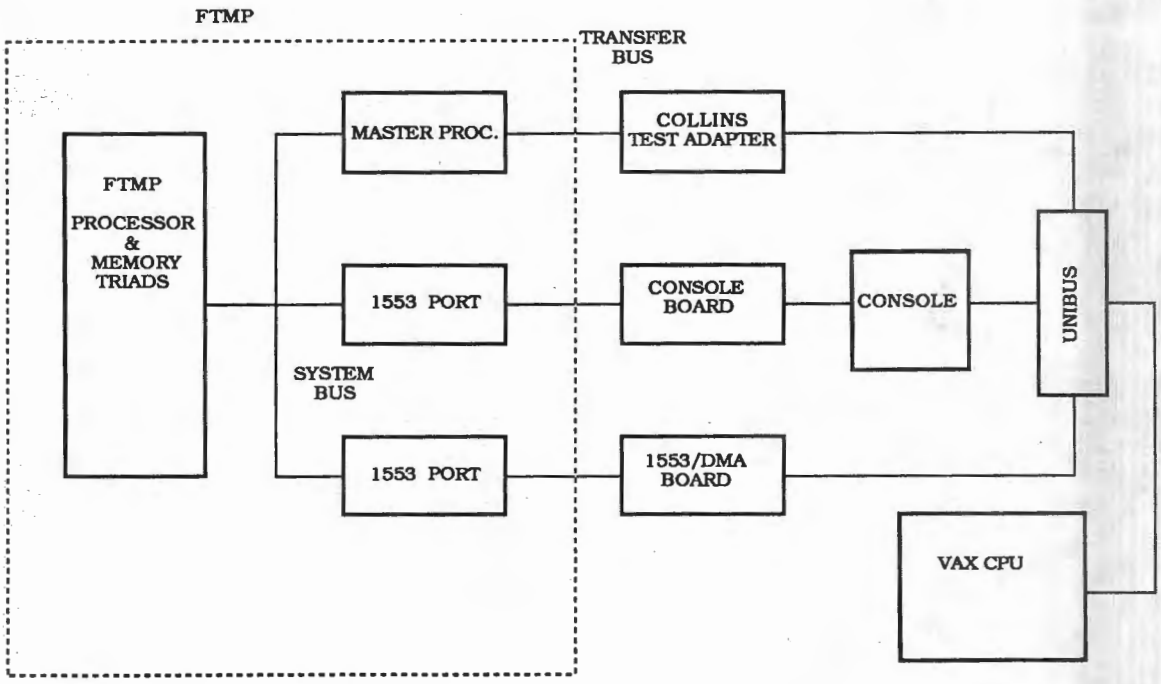



FIGURE 2.6-2. FTMP VAX DATA ACQUISITION ENVIRONMENT

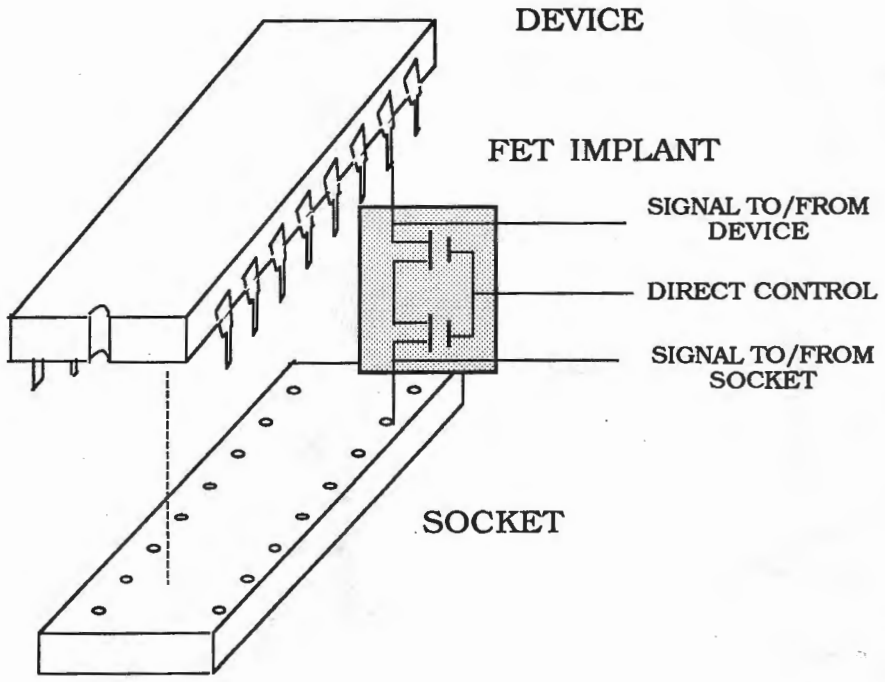



FIGURE 2.6-3. DEVICE PIN ACCESS FOR FAULT INJECTION AND MONITORING

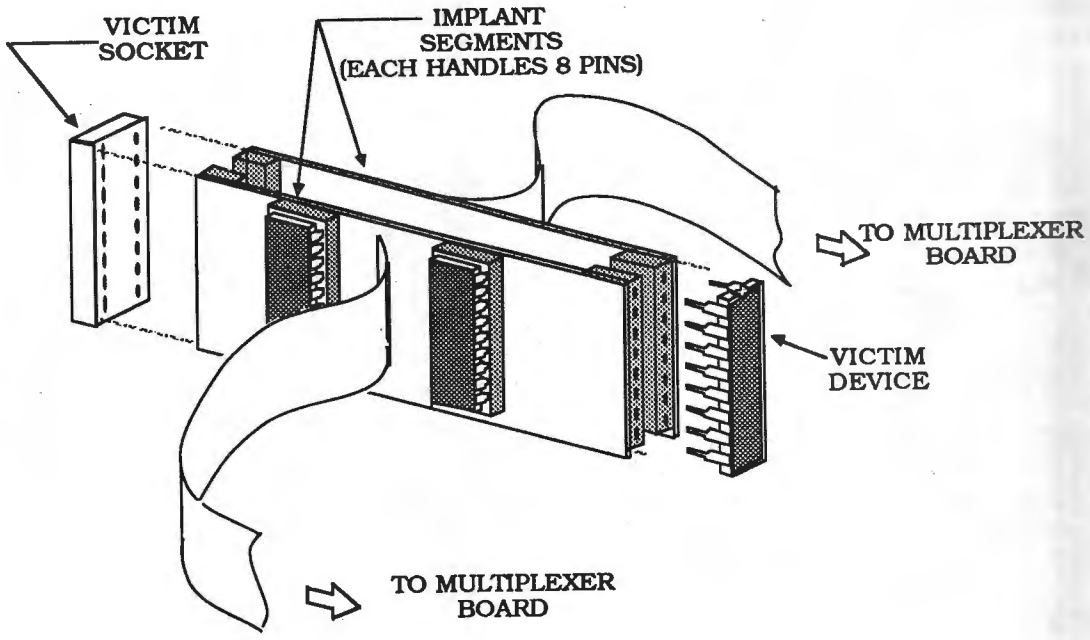
terminals. Turning on the FET establishes a direct connection between the device and the socket. This is the normal condition when no faults are injected. The connection can be severed by turning the FET off. In the off state, any desired signal can be applied to the device or the socket pin. Usually the pin selected is an input. It should be noted that an output pin is usually connected to an input pin of some other device. Added flexibility is provided by multiplexing signals to and from the package and socket under control by the VAX or PDP-11. A choice of eight possible injection signals is provided to the experiment via the multiplex system:

- Direct connection to the socket
- Multiplex A
- Multiplex B
- Multiplex C
- F(AB) Boolean combination of signals from Mux A and B
- F(A,B,C) Boolean function F(AB)and output of Mux C
- Ground, logic zero
- External signal

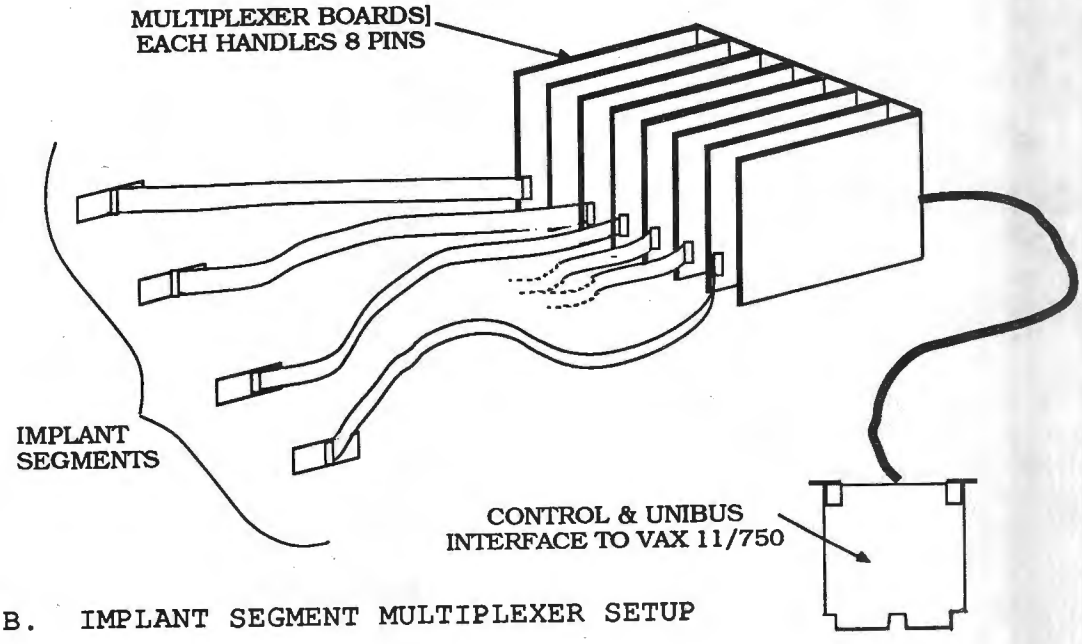
Each of these eight signals may be inverted before application to the victim pin, thus providing sixteen possible sources. The usual experimental run uses S-A-1, S-A-0, and "complemented signal" types of faults. This setup has powerful flexibility. For example, certain faults in integrated circuits can change a NAND gate into a NOR gate. It is possible with the fault injector to simulate such a fault by extending the pins of the target gate with the FETs and generating the Boolean functions. The main utility of this injector is to inject faults on tristate lines. The direction of the fault can be made a function of other signals on the device, signals that determine the state of the tristate control pin. Thus, it is possible to inject faults into the data pins of memory chips, bus drivers, and other tristate devices.

The injector hardware is packaged as shown in figure 2.6-4, with FET pairs on an extender segment. These extenders are used to provide space for the multiplex signals and interface to the sockets. Each multiplexer board accommodates eight pins. The six boards are connected to the instrumentation processor via a unibus card into the VAX or PDP-11.

The recent experiments using the FIIS at NASA Ames provided information on the behavior of a completed FCS operating in a simulated aircraft environment. The CAP-6 processors used in this system had approximately 1,200 pins. For this system a nearly complete coverage of single faults was injected on all possible pins. These results are an important addition to the information obtained from the earlier work on the Bendix 930 and on the FTMP. In these experiments over 2,700 simulated faults were applied to the chip pin level. Most testing was done open loop. The more persistent faults were subjected to closed loop




A. DEVICE TEST ADAPTER




B. IMPLANT SEGMENT MULTIPLEXER SETUP

FIGURE 2.6-4. FAULT INJECTOR HARDWARE

testing in order to involve the built-in fault detection mechanisms. The test results, summarized in table 2.6-1, are in reasonable accord with similar data, confirm well with the fault models, and indicate the value of this type of testing for practical system validation efforts.

The data shown in table 2.6-1 are the result of injecting 2549 "care" faults. While running the baseline FCS program under open loop control conditions, 2461 were detected and 88 were undetected. Open loop tests were conducted to speed up the process because the closed loop system provided a layer of error detection and recovery. Of the 88 undetected faults, 15 were invert faults. Of the invert faults, 1.71 percent were detected while 4.37 percent of stuck-at faults were undetected, indicating that invert faults are less latent than stuck-at faults. Each of the 88 undetected faults were injected while executing the self-test program. All these faults were detected indicating that the processors "care" about these cases. All the micromemory and processor faults injected were detected. The 100 percent coverage of the self-test program indicates the value of running self-test in the background, if possible, every few minutes even though no errors are detected by the system in its present mode.

TABLE 2.6-1. EXPERIMENTAL RESULTS SUMMARY FROM FIIS

| DEVICE | # FAULTS INJECTED | # FAULTS DETECTED | # FAULTS UNDETECTED | # DON'T CARES | PERCENT DETECTED | PERCENT UNDETECTED |
|---|---|---|---|---|---|---|
| Data Path | 1535 | 1502 | 33 | 83 | 97.85 | 2.15 |
| Control | 1014 | 959 | 55 | 83 | 94.58 | 5.42 |
| S-a-0/S-a-1 | 1670*** | 1597 | 73 | 123 | 95.63 | 4.37 |
| Invert | 879 | 864 | 15 | 43 | 98.29 | 1.71 |
| 2901 | 423 | 423 | 0 | 25 | 100.00 | 0.00 |
| Micromemory | 231 | 231 | 0 | 9 | 100.00 | 0.00 |
| Input Pins | 1805 | 1741 | 64 | 87 | 96.45 | 3.55 |
| Output Pins | 744 | 720 | 24 | 79 | 96.77 | 3.23 |
| Self-Test | 1687 | 1687 | 0 | 0 | 100.00 | 0.00 |
| Totals | 2549 | 2461** | 88* | 166 | 96.55** | 3.45** |

NOTE: (1) Injected faults do not include "don't cares"
(2) Percent "don't cares" = 6.11
(3) * = Detected by self-test
(4) ** = Excludes self-test
(5) *** = 791 S-a-0, 806 S-a-1

Investigations are under way at NASA Langley, using the advanced data acquisition environment shown in figure 2.6-2, to investigate the FTMP error detection and faulty unit isolation functions (Padilla 1989). Earlier tests disclosed that the FTMP occasionally disabled the wrong units. This behavior continued

until the system crashed after 4,000 to 6,000 faults were injected. These investigations uncovered reasons for the failures due to the fault identification procedure incorrectly identifying that only one Line Replaceable Unit (LRU) detected the error. This event triggers a software component, designed to handle Byzantine faults, which keeps track of the accuser and disables it after a second event. In this case one fault causes two LRUs to be identified as in error, one bad unit and one good unit.

As a result of these investigations, a new methodology for selection of the pins for fault injection is proposed (Padilla 1990). To investigate the abnormal behavior of FT systems, tests should be based on two principles:

- Faults should be injected in signals having high fan-outs (board enables and other control signals), thus maximizing the amount of interaction (number of potential errors).

- Information to determine the system state should be observable at all times in real time.

These principles, although simple in concept, require a completely new experimental environment. The data acquisition environment overburdened the original injection and test system. The new data acquisition system, figure 2.6-2, has three orders of magnitude increase in data rates over the old system.

Some of the initial experiments using the new system disclosed that LRUs not being faulted were being disabled by the fault management software. Although the system classified the events as caused by real permanent faults that occurred during the experiments, no faults could be found at the conclusion. All the supposed faulty units were reactivated either manually or by rebooting the system.

### 2.6.4 Limitations

Faults used in testing are often limited in type and number. Since real hardware is used for the system under test, faults may only be injected at connector pins. Internal gate nodes may not be tested.

The hardware and software must be available or sufficiently well defined that prototypes may be made.

The methodology for selection of pins for fault injection does not have universal acceptance. Systems architecture and design assumptions may limit the faults tested to those locations where designs have addressed error detection and correction. On the contrary, pins may be selected at random, perhaps testing unused gates or overlooking important vulnerabilities.

Physical FI has been used extensively in system validation, with faults inserted at the pin level as S-A-1 and inverted faults. With systems implemented in an SSI or medium scale integration (MSI), pin-level stuck-at faults closely resemble failures which have been observed to occur in such devices. But with LSI and VLSI realizations, failures may be remote from the input/output pins. At these higher levels of integration, FI seldom claims to portray physical

faults accurately. However, the hope is that they prove a first approximation to the metrics under study. The data is not yet available to verify this assertion. Additional information is presented with detailed examples in section 3.

Although faults may be remote from the boundaries, promising results have been reported with pin-level FI for LSI and VLSI devices. Schuette et al. (1986) inserted transient faults of the data, address, and control lines of an MC68000 bus, representing faults within data and control sections of the processor. With this FI technique, two error detection schemes were evaluated. Czeck et al. (1987) inserted faults in an FTMP triad by causing one unit to execute a special code, causing a trigger in the error detection mechanism. This method was able to duplicate some of the results presented by Lala and Smith (1983). Even so, McGough and Swern 1983 illustrated a gap between gate and pin-level fault injection.

## 3. ADDITIONAL FAULT INSERTION AND TEST EXAMPLES

### 3.1 Draper Labs FTMP Tests Using Fault Emulation

This section describes the results of a simulation test using fault insertion. This experiment was conducted in the NASA Langley AIRLAB Diagnostic Emulation (DE) facilities, and was described fully in Baker, Mangum, and Scheper (1988).

The primary objective of this experiment was to determine the diagnostic self-test sequences used to uncover latent faults in a logic network providing the key fault tolerance features for a flight control computer. In this experiment, using the setup shown in figure 3.1-1, more than 1,600 faults were injected into a logic gate-level emulation of the Data Communicator Interstage (C/I), a key component of the CSDL FTMP. For each fault injected, diagnostic sequences consisting of over 300 test vectors were supplied to the C/I model as inputs. For each vector in a test sequence, outputs from the C/I were compared with the outputs of a fault free C/I. If the outputs differed, a fault was considered to be detectable for the test vector. These results were analyzed to determine the effectiveness of the various test sequences, to identify latent faults, and to identify ways for improving performance of the diagnostic sequences.

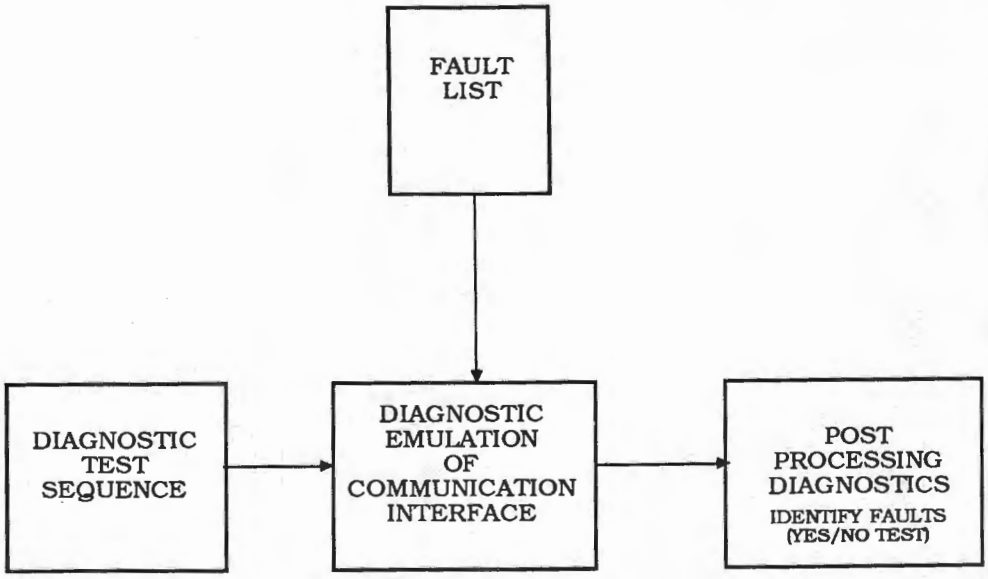



FIGURE 3.1-1. EXPERIMENT DIAGRAM

The C/I of the FTMP was selected because of the following reasons:

- The C/I network provides the critical features of input source congruency, error masking, error reporting, and system reconfiguration.
- The quadruplex C/I network used the full capacity of the AIRLAB DE.
- Documentation for the C/I design at the logic gate level was available.
- Results of these experiments could provide important information about the FTMP design.

The C/I provides the Fault Tolerant Processor (FTP) with certain key fault tolerance capabilities. These capabilities arise for circuitry to support correct replication of simplex source data in all good channels, to detect and mask errors in a single channel, and support reconfiguration by blocking out faulty channels.

The data exchange network associated with a quadruplex FTP is illustrated in figure 3.1-2. Each channel, A, B, C, or D, has transmit and receive registers which can be accessed by the input/output processor or the computation processor via a shared memory bus.

Simplex data (i.e., data available to one FTP channel) is distributed to all FTP channels by a "FROM x" exchange. This will cause the data word in the communicator transmit register of channel A to be sent to all other channels. Data transfers between communicators and interstages are byte serial with two bytes comprising a data word. Data sent to each communicator are distributed to associated interstages.

Each interstage replicates and passes a copy to each communicator. Communicators vote the data received and write the received data into the receive register. Receive registers are then read into the associated channel processor. In this manner simplex data from channel A is read into all good channels.

If each channel has the data required for fault masking, then a "FROM OWN" exchange can be performed. Errors detected during the voting process of an exchange are recorded in the Status Register of each communicator. Each status register can be read by its associated processor. These error reports are intended for use by the FTP in determining and isolating faulty channels.

A write-only Mask Register allows a channel to be excluded from the voting process.

The capability to execute exchanges which bypass the interstage and voter is provided for network testing.

The local data bus provides a large share of the communications. It is a 16-bit tristate bus communicating most of the signals required for redundancy and processor management.

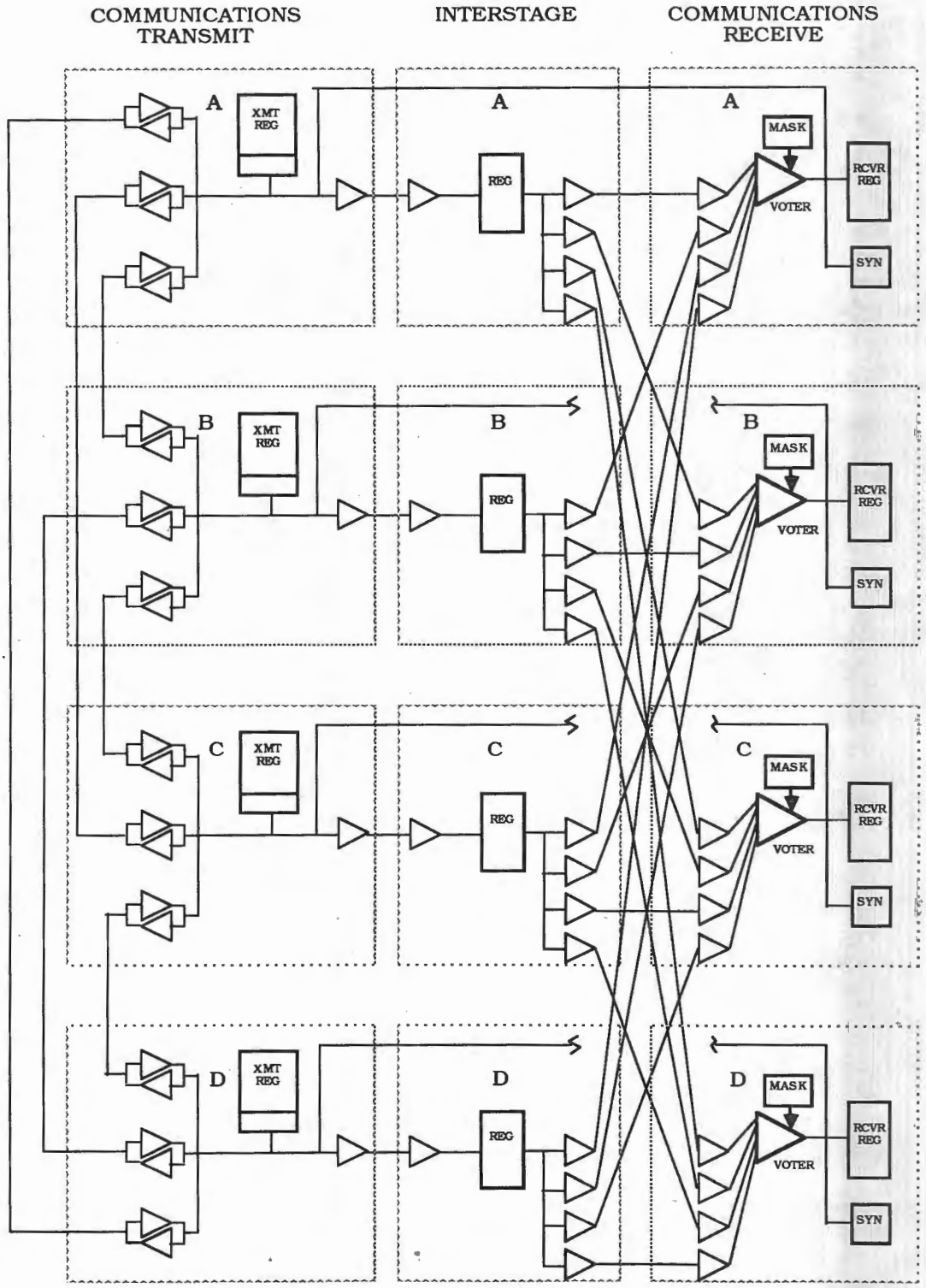



FIGURE 3.1-2. C/I EXCHANGE NETWORK

### 3.1.1 Gate-Level Model For Redundancy Communicator Interstage

During the analysis several assumptions and approximations were found necessary. Some of these proved limiting on the accuracy of the faulted performance prediction. Key features of the model and several limitations arising from these assumptions are discussed in the next paragraphs.

The local data bus was modeled using AND gates at each source enabled by the appropriate source enable line. The resulting model was functionally equivalent to a tristate bus. The OR function is a phantom gate in that there is no identifiable device in the network to which it can be associated. S-A-0 faults were inserted on this phantom gate to simulate the effects of certain bus faults.

The Transmit Register, Mask Register, and Receive Register are all implemented using D-type latches with tristate outputs. The storage portion of these circuits was modeled as D-flip-flops having tristate outputs. The source bus communicates data bytes from the Transmit Register to cross channel links and to the interstages. This bus was also modeled with AND gates and phantom OR gates similar to the local data bus.

The interstages consist of drivers, receivers, and D-type flip-flops. These devices were modeled using the diagnostic emulator gate and the same flip-flop primitives as above.

The cross channel links that distribute data from each communication source bus to all other communicator source buses consists of a bidirectional, tristate bus driver. The circuitry is diagrammed in figure 3.1-3. The model of this portion of the FTP required 384 connections and 96 phantom gates. A decision was made to simplify the circuit to 192 connections and zero phantom gates. This was based upon an incomplete analysis which indicated that the simpler circuit would adequately represent the network behavior. While data is handled by this model, the effects of faults are not properly treated. Consequently, certain gate faults were not found by the C/I diagnostics. In the discussion presented in Baker, Mangum, and Scheper (1988) these faults would have been discovered using the more complex model.

The Status register file is implemented with two 4x16 Random Access Memory (RAM) chips. Only 5x5 bits of the register in use were modeled. This required five five-bit D-flip-flops with address decoder and selection logic to direct the data to and from these registers.

The Mask transfer block is implemented with a 2048x8 bit programmable ROM. A feature of the diagnostic emulator was used to model this device. This feature allows storage of memory contents outside of the range of memory being modeled. In this function, bits are forced to the values contained in the ROM address. Two other programmable logic arrays are used in the FTP. These were modeled directly as logic equations. The adequacy of stuck-at faults for these devices is questionable since bridging faults cannot be represented with such a simple model. A more elaborate model was judged to be beyond the scope of the modeling effort.

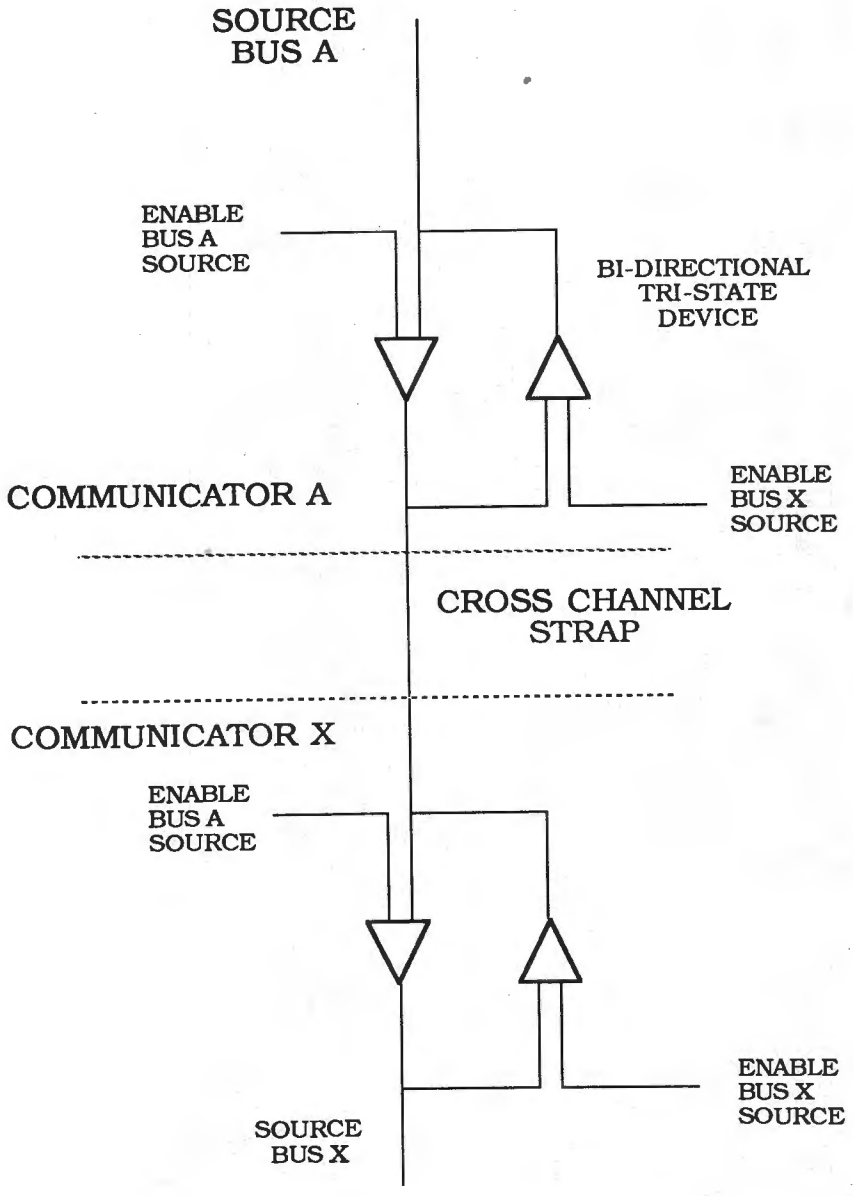



FIGURE 3.1-3. ACTUAL CIRCUITRY FOR CROSS CHANNEL LINKS

### 3.1.2 Fault Detection Effectiveness of Diagnostic Tests

Most of the effort was directed toward developing and validating the data C/I gate-level model and associated software. These areas of activity included:

- Development and testing of experiment support software.
- Activities associated with learning to use the diagnostic emulator.
- Development of a gate-level model of the C/I.
- Development of C/I sequences that would test the C/I in accordance with the CSDL C/I software description.
- Validation of the C/I model.

The system flow diagram for the diagnostic emulation experiment is shown in figure 3.1-4.

The C/I plays a critical role in FTP fault tolerance by ensuring that all good processors receive identical data values, and by masking errors and providing for reconfiguration after faults are detected. An important part of this operation is the use of diagnostic test sequences for the C/I. These sequences are used to uncover latent faults. The experiment was also designed to determine the effectiveness of proposed diagnostic test sequences.

A typical sequence of operation mixes the test sequences with voting and data transfer between processor channels. In operation, about nine transfers are required between the four processors; the test sequences using vectors consisting of self-test and two exchanges are intermixed with the normal transfers. A complete test requires approximately $2^{16}$ test vectors. Assuming that self-test is constrained to using only five percent of the processor through-put capacity, the self-test is constrained to use only five percent of the C/I throughput capacity. The self-test will require approximately two minutes to execute.

Estimates of the time needed to conduct a fault injection experiment using the diagnostic emulator are also of interest. Consider an experiment that requires the injection of 2,000 faults with the C/I self-test sequence being run for each of the faults. Assume that the NANO Fast QM-1 is used and that only self-test must be run (i.e., the simulation is 100 percent used for test sequences). The QM-1 will execute approximately 50,000 times slower than real time. The total simulation time required would be 19 years. If the VAX FORTRAN version of the DE were used, the simulation would require 3.8 centuries. These unacceptably long run times provided motivation for restricting the experiment. If the required simulation time could be reduced by a factor of about 100, then 200 hours of run time on the QM-1 would be needed. Such a simulation would be well within the feasible range.

A reduction in time was planned for and accomplished by noting that all of the transactions needed to share opinions in an actual FTP would not be needed in an experiment. The additional factor of 100 reduction was obtained by reducing

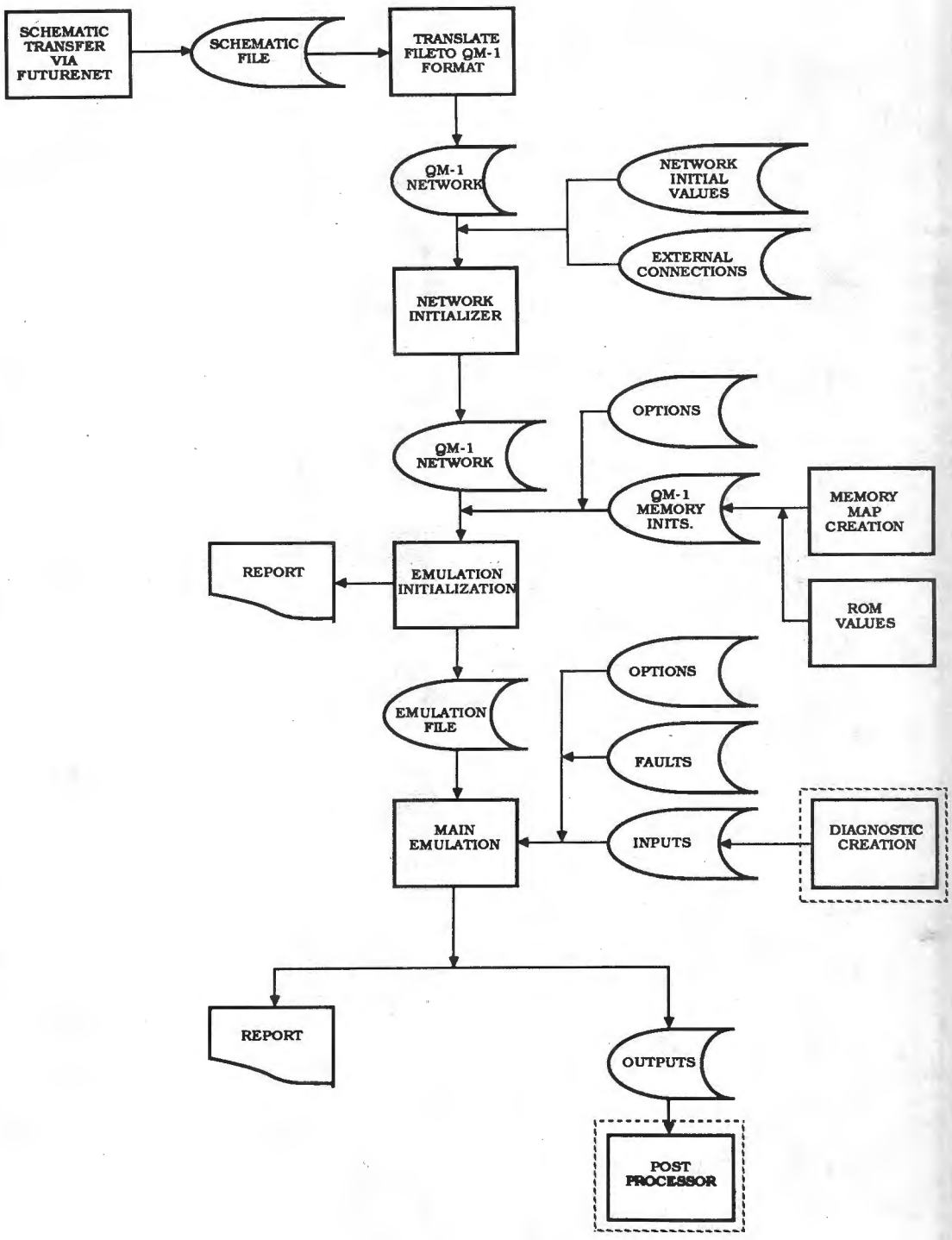



FIGURE 3.1-4. SYSTEM FLOW DIAGRAM

the number of test vectors. This was accomplished by eliminating the test vectors on the Programmable Array Logic (PAL) failure modes since these were modeled not to have errors. Further it was decided to test only the C/I vectors for the quadruplex and triplex configurations and only one for the PAL chips. The experimental tests used are shown in table 3.1-1. The C/I model is shown in figure 3.1-5.

### 3.1.3 Experimental Results on Hardware/Software

The experimental data presented are drawn from the test report by Baker, Mangum, and Scheper (1988). These data are not adjusted or modified to account for certain problems in the models. For each self-test, the percentage of faults detected is relative to the number of faults injected. It should be noted that any conclusions need to be drawn from the complete report and not this synopsis. Data shown is to illustrate the kinds of experimental results available and not the detailed conclusions.

TABLE 3.1-1. TESTS USED FOR EXPERIMENT

- VOTER TESTS

  Tested a single voter PAL chip (2 bits) in the quad and triplex configurations. Used both upper and lower bytes to give two test vectors per vote.

- MASK TRANSFORM TESTS

  Tested mask transform logic for quad and duplex configurations.

- CURRENT STATUS UPDATE TEST

  Tested the current status update PAL.

- PRESENCE TEST

  Equivalent to CSDL "Presence".

- BASIC TEST

  RTI test which tested a range of C/I functions beyond "Presence".

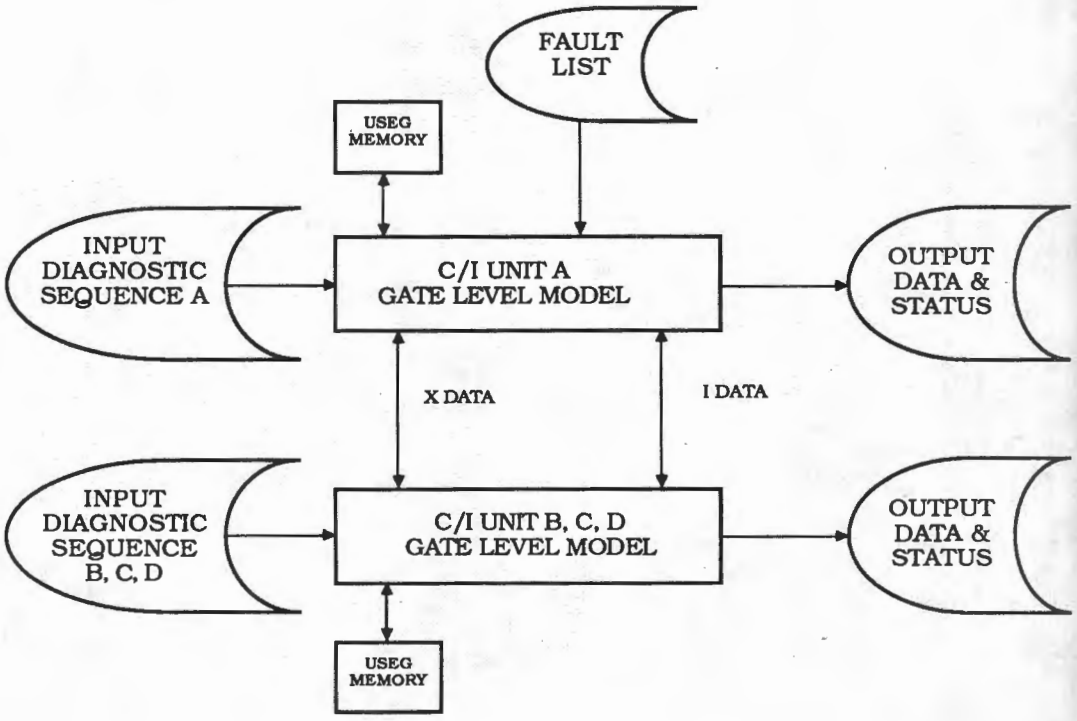



FIGURE 3.1-5. C/I MODEL FOR DIAGNOSTIC EMULATION

Figures 3.1-6 and 3.1-7 illustrate some of the detailed results in the report. The first shows the effectiveness of the diagnostic test run on the simulation. The second shows the performance of the non-voter versus the voter tests in detecting faults.

Table 3.1-2 summarizes the undetected faults for the 2,000 injected faults. If the results were adjusted for assessments of the undetected faults, and if the identification (ID) logic faults are inconsequential, the adjusted performance of the C/I self-tests would exceed 98 percent and could approach 100 percent coverage.

Much of the effort in this experiment was directed toward recapturing the C/I design, and validating the model derived from the recaptured design. Well established Computer Aided Design (CAD) tools will be required to expedite the modeling process to deal effectively with VLSI designs.

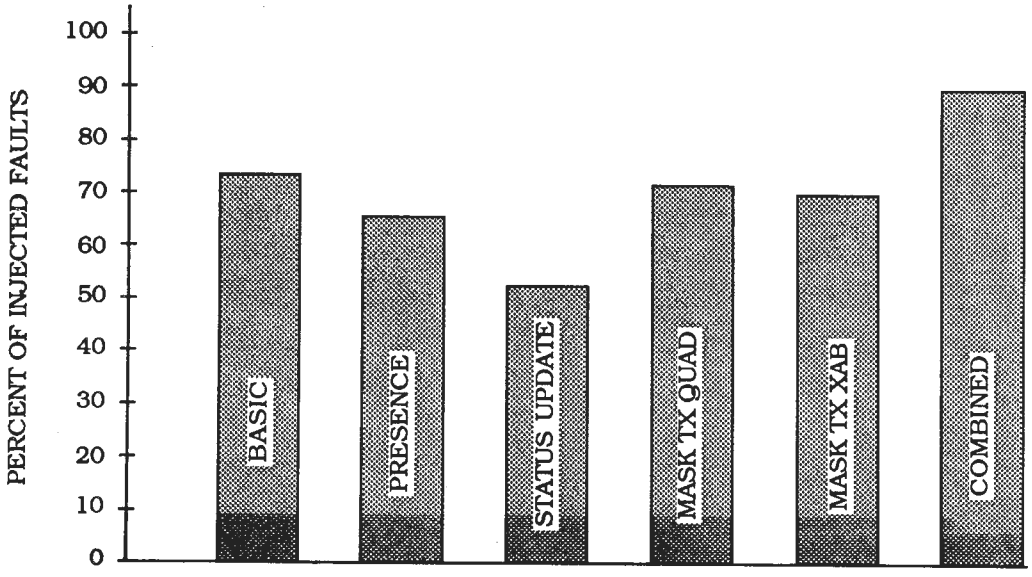



FIGURE 3.1-6. DIAGNOSTIC TEST PERFORMANCE

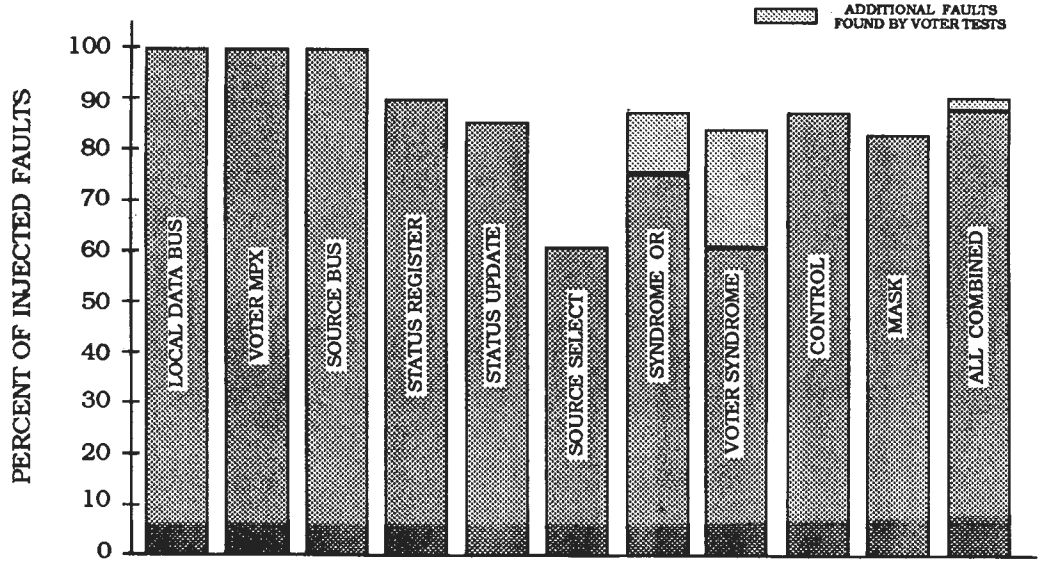



FIGURE 3.1-7. COMBINED PERFORMANCE OF ALL NON-VOTER TEST

TABLE 3.1-2. SUMMARY OF UNDETECTED FAULTS

| FAULT SET | UNDETECTED FAULTS | SOURCE | GOAL | ESTIMATED UNDETECTABLE |
|---|---|---|---|---|
| Source Select | 47 | Cross channel model and ID logic | | 0 |
| Control | 26 | ID logic and cross channel + product interface and reset | | 3 |
| Voter Syndrome | 19 | | Run Voter Quad Test | 0 |
| Status Update | 16 | Not known | Run Voter Quad Test | 0-16 |
| Syndrome Or. | 14 | Not known | Run Voter Quad Test | 0-14 |
| Status Reg. | 15 | Status reg. address decode | Modify Self-Test | 0 |
| Mask | 4 | ID logic | | 0 |
| Local Data Bus | 0 | | | 0 |
| Voter Mux | 0 | | | 0 |
| Source Bus | 0 | | | 0 |
| Combined | 141 | | | 3-33 |

## 3.2 Electromagnetic Induced Transient Injection Test Examples

### 3.2.1 Electromagnetic Transients and Fault Injection

It has been estimated that approximately 80 percent of sudden computer failures, or crashes, are caused by some form of electric transient (Iyer and Rosetti 1986). This statistic serves nicely to illustrate the crucial role that transient EM fields may play in flight system fault tolerance and reliability.

The avionic industry is experiencing a rapidly growing dependence on extensive digital control of complex and crucial flight systems. These systems necessarily demand high reliability and fault tolerance. Digital systems, however, are notoriously vulnerable to extraneous electric fields and transient pulses of voltage.

Modern aircraft are also utilizing more composite types of materials in their structural design. These materials offer high strength and lightweight structures which are very beneficial for increased fuel efficiency. However, these materials are ordinarily less efficient at reflecting and attenuating EM fields. Furthermore, a material joint between an all-metal part and a composite part can serve as an efficient penetration point for EM fields by disrupting the surface currents induced on the two materials. In short, modern aircraft electronic equipment will likely be subjected to significant EM induced transients in the wiring and equipment.

Guidelines are being developed for methods of determining upset susceptibility of redundant systems to EM transients. The studies of transient susceptibility and transient injection, discussed in this section, will lead to increased understanding of upset mechanisms and thereby improved tolerance strategies.

EM transient fields arise from a wide variety of sources. The most significant sources in terms of very high potential energy are lightning and high power Radio Frequency (RF) transmitters, such as for radio, microwaves, and radar.

Lightning pulses are similar to Nuclear Electromagnetic Pulses (NEMP), both having similar waveforms in their signature transient fields. Both may typically be characterized in the time domain by a double exponential waveform with a very rapid rise time, and a fall time from 10 to 100 times longer than the rise time. An example of this waveform is illustrated in figure 3.2-1.

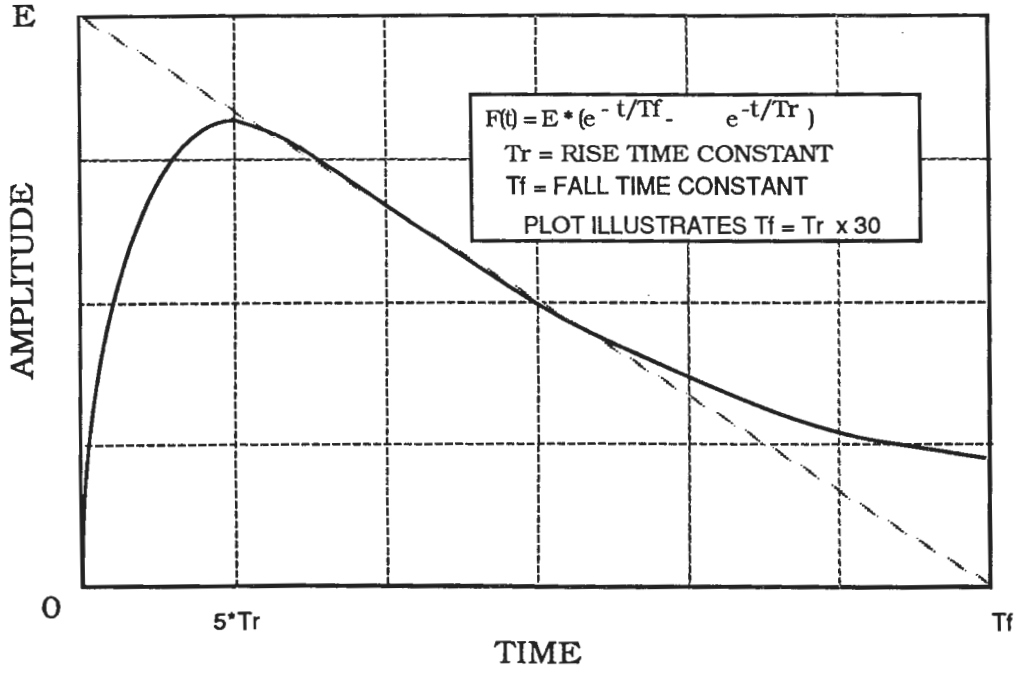



FIGURE 3.2-1. TIME-DOMAIN TRANSIENT WAVEFORM GENERATED BY LIGHTNING

These transients predominantly cover the frequency range from tens of kiloHertz (kHz) to hundreds of megaHertz (MHz). An adequate simulation waveform for laboratory and first order analysis of equipment response purposes is an exponentially damped sinusoid which reaches about 60 percent of maximum amplitude on the order of four cycles (SAE AE4L 1988, Rasch 1987, Cooley and Shortess 1988).

Aircraft are regularly exposed to High Energy RF (HERF) fields from radio, microwave, and radar. Effects on aircraft systems from these power transmitters is a function of the aircraft materials and electronics technology, as well as the power levels. Important factors include the transmitter modulation characteristics (AM, FM, pulsed, continuous wave) and geometry, such as the orientation of the aircraft in high field regions and the proximity of the source antenna.

Ordinarily, the voltage and current waveforms on an antenna will be the same as the EM field waveform. However, an exponentially damped sinusoidal EM field will induce an exponentially damped sinusoidal voltage/current onto a line or within a component. The coupling mechanisms for transients in aircraft wiring are often much more complex events. The aircraft structure and wiring acts like a complex antenna for remote EM sources. Complex RLC and M parameters govern the coupling from currents on the aircraft structure to critical wire and circuits within. Induced transient waveforms inside an aircraft may be derivative signals of complex filter functions of the incident EM fields. The most dominant coupling mechanisms include current or voltage induced directly by the EM field onto a wire or component (essentially acting as part of an antenna). In addition, important interactions include inductive or capacitive coupling from some conductive medium or aperture near a wire or component, and other interactions, such as fluctuations in the ground potential as the transient passes.

Methods for simulating or electrically reproducing the EM induced transients has to date been accomplished through laboratory bench test techniques. It is currently unknown what the implications may be of introducing a transient through only one specific wire in a test experiment. One feature of EM transients is that they may affect all the components of a redundant system at the same time. This massive injection of faults could effectively circumvent the fault detection and recovery process. It would seem likely that the system would be affected in parts. Some parts of the system may be most vulnerable to perhaps an inductive coupling of a current to a wire, whereas other parts of the system will have difficulty with a ground potential fluctuation. Designing a test program to address all of these issues is very challenging. Of course, economics and time are always the practical limitation of any test capability.

Another consideration is the timing of the transient injection. It may be logical to inject the faults using the same clock used for the rest of the testing system. However, if this timer is also driving the test subject, the result could be a synchronous injection of the fault occurring at the same time as changing system states. This would likely influence the data with an unrealistic bias. A technique for random asynchronous fault injection would

more closely resemble actual environments. Detail studies could benefit from different time phase injections relative to system clock pulses.

Experimental investigations of EM transient induced faults may be further divided into two realms of testing depending upon the level of system definition in the test. As discussed in section 2, tests may be performed entirely as simulations on computers.

A real or conceptualized system may be modeled down to the functional block, VLSI chip, or the transistor gate level using available circuit and logic simulators. The simulated system is then stepped in time through a series of tasks as fault signals are injected into circuit nodes using ideal voltage or current sources. This technique has the advantage of being able to completely control where faults are injected, and at precisely what time and amplitude. Simulations can trace exactly what the injection produces throughout the entire system. The limitation of this method is in the expense (both time and money) of the computer time required to step the entire system through any substantial amount of real time, not to mention analyzing the enormous amount of data which is generated. Furthermore, a software simulation can never precisely parallel the actions of a real system.

At the other extreme, hardware testing physically injects EM transient currents or voltages into a real system.

An actual system may be brought into the laboratory, and the system altered to allow transient injection and monitoring of pertinent system data. Care must be taken not to perturb the true functionality by loading the system's hardware with the transient injection equipment or the data monitoring equipment. The obvious advantage of the real system method is that a real system is observed in an actual functional environment while subjected to controlled EM transients. A very broad range of system states may be investigated in this manner over long periods of time. The disadvantages to this method are also numerous. It may be very difficult to access some pertinent data points; fault injection levels are much less controllable (although more realistic), and tracking fault effects through the system may be difficult.

### 3.2.2 Upset and Failure Detection

Component disruptions in FT systems must be handled through conventional fault tolerant methods such as redundancy and reconfiguration. Digital systems can be particularly vulnerable to EM transients. System functions can be disrupted in two ways:

- The components themselves may be permanently damaged by a high potential or current. It is known that charge accumulation levels above nine picocoulombs injected into a node within a CMOS microcomputer VLSI chip can cause permanent damage (Nichols et al. 1985). The speed and high magnitudes of many EM transients can easily lead to such charge accumulations.

- Component functions may be disrupted without damage. This is termed an "upset". The normal functionality of a system or component may be altered

by the passing of a transient. The loss of function may persist after the transient has gone, although no actual physical damage occurs.

A simple example of a serious disruption would be a logic gate temporarily stuck in a single state. This particular upset could cause a microprocessor to access an erroneous address space, program execution to branch to a subroutine intended for a completely different function, or incorrect mathematics to occur in an internal register.

Testing components and systems for EM transient fault tolerance is also a new area of research. A few pioneering efforts have been made on general purpose microprocessors executing simple codes, but these systems are not necessarily representative of actual avionic systems. Two recent works (Belcastro 1989 and Carreno, Choi, and Iyer 1989) provide sources for references to earlier studies. To date there are very few studies which have been performed on actual avionic systems; section 3.2.3 describes two such studies on an avionic electronic engine controller. These studies are detailed in Belcastro (1989) and Carreno, Choi, and Iyer (1989).

The two studies covered in section 3.2.3 are particularly beneficial in that each study was performed in a different way using the same electronic engine controller. One study completely utilized the software simulation approach and the other study utilized actual hardware systems. Aside from these strategic differences, the two testing methods have much in common. It is likely that future EM transient testing strategies will not deviate too far from these fundamental practices.

To monitor a system for fault occurrences, the tester needs to know the "normal" system function. The known "normal" function detail must match the same level of detail as the desired testing level of the target system. If the testing is to be done down to the transistor level in a logic system, then the "normal" function of the system at the transistor level must be known.

This knowledge can be gained in several ways as discussed in section 2. One extreme (all in software) is to run a testing algorithm through the system (simulated or actual) without the injection of any faults. The system can then be rerun with precisely the same algorithm with fault injection. The second extreme (all in hardware - used primarily for actual hardware testing) is to run two or more systems in parallel with one system remaining non-faulted as the reference comparison for normal operation.

The multiple real hardware method is limited by such factors as system size, cost, availability, and manufacturing repeatability. Using real hardware is usually faster and more convenient for fault detection testing since detection may occur in "real time" through direct comparison of system states. The single system method is often more economical, slower, and may provide better comparison data since exactly the same algorithm is run through the same system.

The types of system data collected during EM transient testing will probably not differ much between the two testing strategies. This system data will normally include control lines, address buses, data buses, and microprocessor status registers. The system control lines establish the operational mode of the

system. These will often give the most obvious indication of incorrect operation. (The control lines are also probably one of the most significant sources for EM transient conduction to the sensitive digital components.)

The system data buses should be monitored for signs of incorrect command modes and erroneous data accessed (or induced). This data may provide the earliest indication of upset or failure. Similarly, the address buses provide indication of invalid memory access and program execution. This is obviously crucial data for detecting and tracking upset/failure response. Microprocessor status registers may be helpful in tracking a fault occurrence and response. The microprocessor status registers typically include operational information flags, such as parity, configuration, self-tests, watchdog timer, and arithmetic overflow.

### 3.2.3 Digital Engine Controller Test Examples

Two complementary studies, Belcastro (1989) and Carreno, Choi, and Iyer (1989), serve as excellent examples for describing the proposed processes of determining EM transient fault behavior in avionic systems. Belcastro (1989) actually documents the plan for a test which had not been completed at the time of its publication. However, the testing should be complete by the time of this publication. Carreno, Choi, and Iyer (1989) documents in detail an actual test completed in late 1989. These are two of the first such tests to be performed in the area of EM transient upset detection in actual avionic systems. Both of these examples offer tremendous insight into the current state of the art in EM transient fault modeling, detection, and injection. It must certainly be anticipated that this type of work will greatly increase in the coming years.

Both tests utilize the same Hamilton Standard electronic engine control (EEC) unit. This controller is a commercial unit for Pratt & Whitney engines. A block diagram of the EEC system is shown in figure 3.2-2. The fault tolerant strategy utilizes a dual-channel system with an extensive self-testing algorithm to determine which channel is in control of the engine.

#### 3.2.3.1 Physical Transient Insertion

The testing strategy of Belcastro (1989) tests real hardware using a single system run in a non-faulted condition. This run establishes normal mode reference data and ranges. Another run is then made in which the hardware is subjected to EM transient waveforms coupled inductively onto wiring in the EEC. The EEC simulation strategy and a functional block diagram of the test system are shown in figures 3.2-3 and 3.2-4, respectively.

The EEC hardware is altered for test to allow access of the data bus, address bus, and the control lines of the microprocessors in each channel. Digital data is recorded through a DAS8200 digital analysis system using up to 240 input capture lines.

Upset detection is defined as the occurrence of one of three conditions:

- Selected parameter values for engine speed, throttle resolve angle, and inlet air temperature out of range for n cycles.

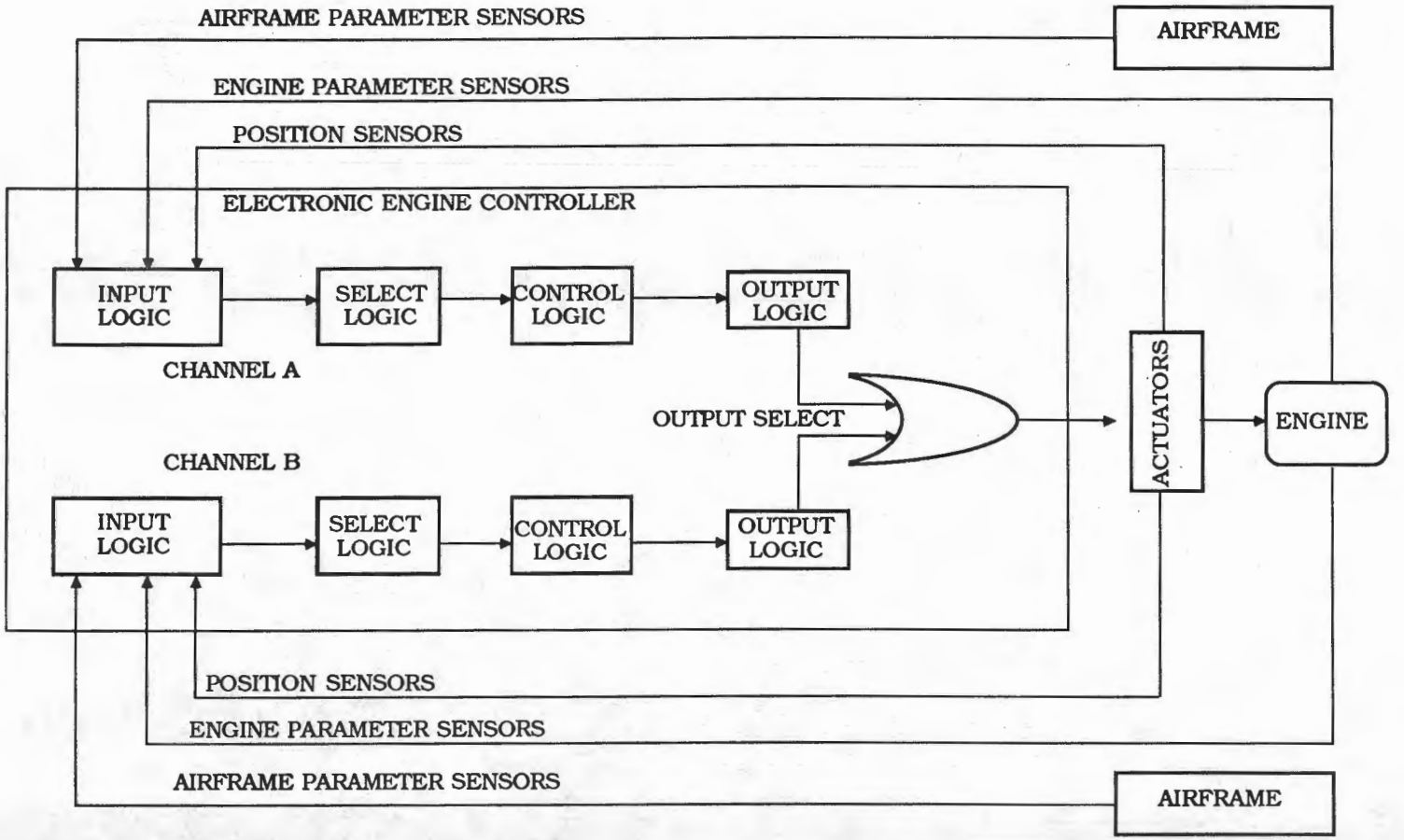



FIGURE 3.2-2. FUNCTIONAL BLOCK DIAGRAM OF THE HAMILTON STANDARD ELECTRONIC ENGINE CONTROLLER

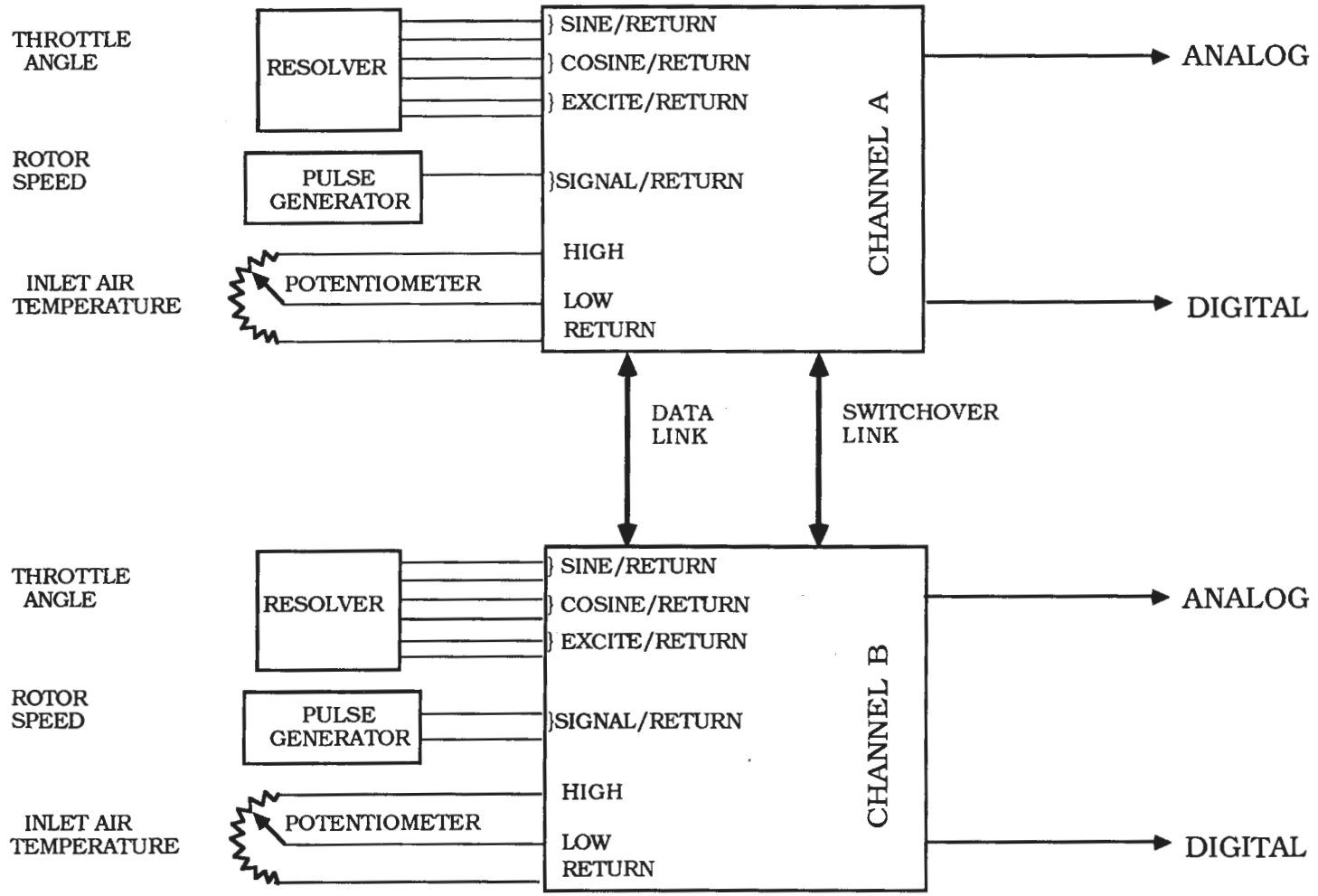



FIGURE 3.2-3. SIMULATION OF EEC INPUTS FOR EM TRANSIENT FAULT INJECTION

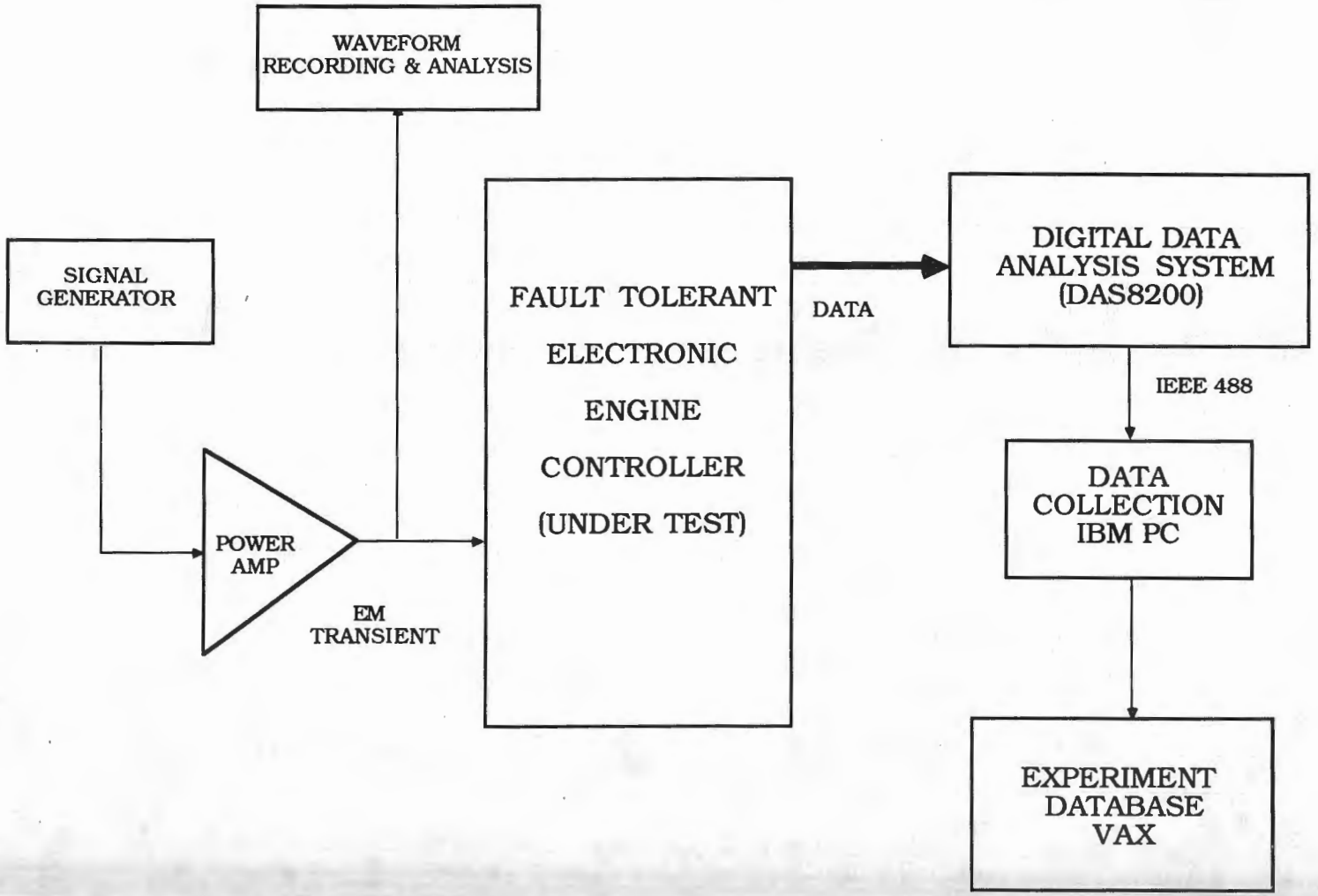



FIGURE 3.2-4 EEC INSTRUMENTATION FOR EM TRANSIENT FAULT INJECTION

- Control commands obtained from the data bus out of range for a given flight mode for n cycles.

- Invalid memory space accessed for n cycles.

Indication of any of these events in the monitored data will result in time-stamped data being recorded for that particular test run; that is, only data from upset/failure detection will be recorded.

EM transient injection is accomplished with an inductive coupler using an exponentially damped sinusoid transient. This test equipment is capable of delivering a 1,000 watt signal over a frequency range of 10 kHz to 220 MHz.

The EEC will be cycled through eight simulated flight conditions utilizing the simulation strategy depicted in figure 3.2-3 with nominal flight parameter values, including most actual engine parameters, stored in a ROM. The EEC will experience idle, acceleration, take-off, climb, cruise, deceleration, reverse, and partial power.

The objectives of initial testing are to help establish a methodology for upset detection strategies, to add to the EM transient fault tolerance database, and to further define characteristic induced-waveform threshold norms leading to upset conditions.

The results of the actual experiment outlined by Belcastro (1989) will complement the experiments reported by Carreno, Choi, and Iyer (1989).

3.2.3.2 Simulated Transient Pulse Injection

The study by Carreno, Choi, and Iyer (1989) approaches testing of the EEC from a software perspective. One channel of the EEC, excluding the interfaces, was simulated with the aid of an application program supplied by Hamilton Standard. Hamilton Standard also supplied a complete gate-level description of the entire 16-bit HS1602 microprocessor used in the EEC. The gate-level description was then further reduced to a CMOS transistor-level description of the gate-level modules. Capacitance loads in the circuit were calculated from metalization lengths used in the circuit layout.

The simulation was carried out using the SPLICE1 mixed-mode simulation program. This program is capable of performing both analog circuit analysis and logic simulation. SPLICE1 was modified to allow for specification of nodal EM transient injections (as opposed to reconfiguring the circuit model by adding a voltage or current source to the individual nodes one at a time). The SPLICE1 simulator was also modified to allow for manipulation of external data files in order to simulate access of memories (RAM, ROM). The program was expanded to output up to 80 nodes' data only when data changed, and to provide an extended trace facility which generates output for a node each time a gate driving that node is evaluated by SPLICE1. This latter feature is particularly helpful when tracing fault propagation through the nearly 4,000 nodes of the HS1602 microprocessor model.

The simulated EM transient fault was of the double exponential type illustrated in figure 3.2-1. A current source was used for the injection. This current source had a rise time constant of 0.05 nanoseconds and a fall time constant of 0.164 nanoseconds. The amplitude of the source varied so as to produce a well defined charge accumulation in the range of 0.5 to nine picocoulombs. This method was used because it was found that the total charge accumulation was the more significant factor in determining upset rather than amplitude alone.

The operation of the microprocessor was categorized on a six part functional modular level: watchdog self-test, multiplexer, decoder, control, countdown, and arithmetic logic unit (ALU). The microprocessor was stepped in one nanosecond increments (82 steps per 12.2 MHz clock cycle) through 74 instruction cycles which included a watchdog test, parity test, instruction set test, RAM test, ROM sum test, and test case parameter transfers to RAM.

Simulations were performed with "gate distances" of one, two, three, four, and five modeled to the transistor level. Gate distance is defined in figure 3.2-5. It was observed that for a combinatorial circuit a minimum of three gate distances are needed for accurate simulation, and at least four gate distances are needed if at least one latch (i.e. flip-flop) is within the three gate distance.

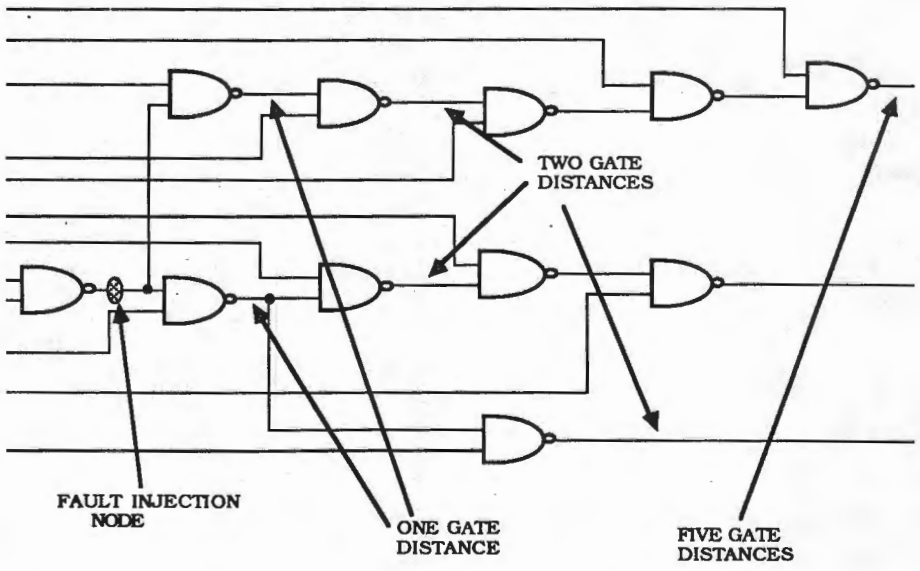



FIGURE 3.2-5. CLARIFICATION OF GATE DISTANCE DEFINITIONS

A non-faulted simulation run was used to generate the normal operation comparison data. This data was also compared to data obtained from an actual hardware unit including a comparison of such details as electrical delays, gate loading, etc. Correct digital data was observed in the simulation including

transitional times to within three nanoseconds of the actual system. These results suggest a high confidence in the modeling accuracy.

A total of 2,100 EM transient injections were performed by randomly selecting seven nodes from each of the six primary functional modules mentioned above. Charge accumulation levels of 0.5, one, two, three, four, five, six, seven, eight, and nine picocoulombs were injected at five different times during the program execution.

Errors were defined as first order if a logic error (difference from the comparison normal run) occurred one clock cycle after the transient injection. The error becomes second order if the logic difference persists two or more cycles after the transient fault injection. These definitions are also used to define pin errors if the device's pins were actually affected. A functional error was noted if the processor's operational function deviated from normal.

Table 3.2-1 summarizes the total error occurrence for the entire test as a function of the accumulated charge level induced by the simulated transient injection. Table 3.2-2 lists the occurrence of all first order errors (most sensitive detection) encountered within each of the six functional modules.

Both of these tables provide significant information about the sensitivity of the HS1602 EEC microprocessor. There is little sensitivity to transient charge accumulations of two or less picocoulombs. There are increasing errors with increasing charge accumulation between two and six picocoulombs. Charge levels above six picocoulombs do not increase upset errors. (Above nine picocoulombs permanent damage begins to occur.)

There is a very strong sensitivity of the microprocessor with respect to functionality. The watchdog self-test and the ALU are both very likely to be upset by this type of EM transient. The decoder, control, and countdown functions are about half as likely as the ALU and watchdog to suffer upset, whereas the multiplexer seems to be rather immune to the transient.

The results of this experiment offer a significant contribution to the growing database for EM transient injection experimentation in avionic systems. These results will be very interesting to compare with the results of the first test described in this section since the two experiments offer two very different perspectives on the same system.

It should be noted that significant computer resources are required. This experiment covers only a very small fraction of the whole system with respect to both components and function time. A simulation run of 12 two-input logic gates and two D-type flip-flops modeled at the transistor level running for 10 milliseconds requires about 130 hours of computer time on a MicroVAX II computer. This will obviously limit the simulation capability of complex avionic systems. However, the first test discussed will not likely produce the highly detailed results of this technique which provide definition of how the microprocessor upset occurs, and particularly in detail within the VLSI chip.

TABLE 3.2-1. ERRORS IN SIMULATED EM TRANSIENT ON EEC

| INJECTION CHARGE LEVEL (Picocoulombs) | LOGIC ERRORS First Order | LOGIC ERRORS Second Order | PIN ERRORS First Order | FUNCTIONAL ERRORS |
|---|---|---|---|---|
| 0.5 | 1 | 0 | 0 | 0 |
| 1.0 | 2 | 0 | 0 | 0 |
| 2.0 | 9 | 0 | 5 | 7 |
| 3.0 | 38 | 10 | 23 | 17 |
| 4.0 | 52 | 14 | 30 | 24 |
| 5.0 | 64 | 15 | 34 | 29 |
| 6.0 | 72 | 17 | 37 | 29 |
| 7.0 | 76 | 19 | 41 | 29 |
| 8.0 | 77 | 22 | 42 | 29 |
| 9.0 | 79 | 23 | 43 | 29 |

TABLE 3.2-2. ERRORS IN SIMULATED EM TRANSIENT ON MICROPROCESSOR MODULE

| MICROPROCESSOR FUNCTIONAL MODULE | TOTAL FIRST ORDER ERROR INCIDENCE | ERROR INCIDENCE PERCENTAGE (Per Injection) |
|---|---|---|
| WATCHDOG | 132 | 37.7% |
| MULTIPLEXER | 0 | 0.0% |
| DECODER | 68 | 19.4% |
| CONTROL | 70 | 20.0% |
| COUNTDOWN | 70 | 20.0% |
| ALU | 130 | 37.1% |

## BIBLIOGRAPHY


Arnold, T. F., "The Concept of Coverage and Its Effect on the Reliability Model of a Repairable System", IEEE Transactions on Computers, Vol. C-22, pp. 251-254, March 1973.

Avizienis, A. and J. Laprie, "Dependable Computing: From Concepts to Design Diversity", Proceedings IEEE, 74(5):629-638, May 1986.

Avizienis, A. and D. Rennels, "Fault-Tolerance Experiments with the JPL Star Computer", COMPCON '72 Digest of Papers, pp. 321-324, September 1972

Baker, Robert, Scott Mangum, and Charlotte Scheper, A Fault Injection Experiment Using the Airlab Diagnostic Emulation Facility, NASA CR 178390, NASA Langley Research Center, March 1988.

Bavuso, Salvatore J. and Paul S. Miner, "Characterization of the Faulted Behavior of Digital Computers and Fault Tolerant Systems", Presented at the Hawaii International Conference on System Sciences, Honolulu, HI, January 1989.

Bavuso, Salvatore J., et al., Applications of the Hybrid Automated Reliability Predictor, NASA Technical Paper 2760, NASA Langley Research Center, December 1988.

Becher, Bernice, Diagnostic Emulation: Implementation and User's Guide, NASA CR 178391, NASA Langley Research Center, December 1987.

Belcastro, Celeste M., "Laboratory Test Methodology for Evaluating the Effects of Electromagnetic Disturbances on Fault-Tolerant Control Systems", NASA Technical Memo 101665, NASA Langley Research Center, November 1989.

Benson, J., D. Mulcare, and W. Larsen, Hardware Fault Insertion and Instrumentation System: Experimentation and Results, DOT/FAA/CT-86/34, FAA Technical Center, March 1987.

Butler, Ricky W. and Anna L. Martensen, The Fault-Tree Compiler (FTC), Program and Mathematics, NASA Technical Paper 2915, NASA Langley Research Center, July 1989.

Butler, Ricky W. and Allan L. White, SURE Reliability Analysis, Program and Mathematics, NASA Technical Paper 2764, NASA Langley Research Center, March 1988.

Carreno, Victor A., G. Choi, and R. K. Iyer, "Analog/Digital Simulation Study of Transient Induced Logic Errors and Upset Susceptibility of an Advanced Control System", NASA Technical Memo XXX, Draft, November 1989.

Carter, William C., et al., Design for Validation: An Approach to Systems Validation, NASA CR 181835, NASA Langley Research Center, May 1989.

Cohen, Gerald C., Charles W. Lee, and Daniel L. Palumbo, "Methods for Evaluating Integrated Airframe/Propulsion Control System Architectures", IEEE-Naecon, Dayton, OH, May 1987.

Cohen, Gerald C., et al., AIAA-89-2703, "Experiences With a Prevalidation Methodology for Designing Integrated/Propulsion Control System Architectures", AIAA/ASME/SAE/ASEE 25th Joint Propulsion Conference, Monterey, CA, July 1989.

Cooley, W. W. and D. Shortess, Lightning Simulation Test Technique Evaluation, DOT/FAA/CT-97/38, FAA Technical Center, October 1988.

Courtois, B., "Some Results About the Efficiency of Simple Mechanisms for the Detection of Microcomputer Malfunctions", 9th International Symposium of Fault Tolerant Computing, pp. 71-74, 1979.

CSDL, The Charles Stark Draper Laboratory, Inc., Document #43-334, CBD Announcement Ref SS017, Issue PSA-9214, November 1986.

Cullyer, W. J. and C. H. Pygot, "Application of Formal Methods to the VIPER Microprocessor", IEEE Proceedings, Vol 134, Pt. E, No. 3, May 1987.

Czeck, E. W., Z. Z. Segal, and D. P. Siewiorek, Software Implemented Fault Insertion an FTMP Example, Department of Computer Science, Carnegie-Mellon University, January 15, 1989.

Czeck, E., D. Sieworek, and Z. Segal, Software Implemented Fault Injection, an FTMP Example, NASA CR-178423, Carnegie Mellon University, October 1987.

Czeck, Edward W., Daniel P. Siewiorek, and Zary Z. Segal, Predeployment Validation of Fault-Tolerant Systems Through Software-Implemented Fault Insertion, NASA CR 4244, NASA Langley Research Center, July 1989.

Czeck, Edward W., et al., Software Implemented Fault Insertion: An FTMP Example, Department of Computer Science, Carnegie-Mellon University, January 1987.

Decouty, B., G. Michel, and C. Wagner, "An Evaluation Test of Fault Detection Mechanisms Efficiency", 10th International Symposium on Fault Tolerant Computing, pp. 225-227, 1980.

Dotson, Kelly J., "Examples of Nonconservatism in the CARE III Program", NASA Technical Memo 100524, NASA Langley Research Center, January 1988.

Dotson, Kelly J., Analysis and Testing of the SURE Program, NASA Technical Paper 2817, NASA Langley Research Center, August 1988.

Dzwonczyk, M. J., et al., Avionic Architecture Studies for the Entry Research Vehicle, NASA CR 181828, NASA Langley Research Center, May 1989.

Ferguson, F. J., Inductive Fault Analysis of VLSI Circuits, PhD Thesis, Carnegie-Mellon University, Electrical and Computer Engineering Department, October 1987.

Ferguson, F. J., J. P. Shen, and W. Maly, "Inductive Fault Analysis of MOS Integrated Circuits", IEEE Design and Test of Computers, 2(6):13-26, December 1985.

Finelli, G., "Characteristics of Fault Recovery Through Fault Injection on FTMP", IEEE Transactions on Reliability, R-36(2):164-170, June 1987.

Finelli, George B., AIAA-88-4436, "Results of Software Error-Data Experiments", AIAA/AHS/ASEE Aircraft Design, Systems and Operations Conference, NASA Langley Research Center, September 1988.

Goetz, F., "Design for Detection, an Attempt at Complete Fault Detection of State", Digest of Papers, COMPCON '78, pp. 325-328, 1972.

Iyer, R. K. and D. J. Rossetti, "A Measurement Model for Workload Dependence of CPU Errors", IEEE Transactions Computers, vol. C-35, pp. 511-519, June 1986.

Johnson S. J., "Evaluation of Fault-Tolerant Parallel Processor Architectures Over Long Space Missions", NASA Memorandum 4123, NASA, 1989.

Johnson, Sally C., "Assist User's Manual", NASA Technical Memo 87735, NASA Langley Research Center, August 1986.

Kurlack, R. and J. Chobot, "CPU Coverage Evaluation Using Automatic Fault Injection", 4th AIAA/IEEE Digital Avionic Systems Conference, pp. 294-300, 1981.

Lala, Jaynarayan H. and Basil T. Smith, III, Development and Evaluation of a Fault-Tolerant Multiprocessor (FTMP) Computer, Vol. II, FTMP Software, NASA CR 166072, NASA Langley Research Center, May 1983.

Marchall, P., "Updating Functional Fault Models for Microprocessors Internal Buses", 15th International Symposium of Fault Tolerant Computing, pp. 58-64, 1985.

Martensen, Anna L. and Salvatore J. Bavuso, "Tutorial and Hands-On Demonstration of a Fluent Interpreter for CARE III", NASA Technical Memo 4011, NASA Langley Research Center, November 1987.

McGough, John G., Digital Systems Validation Handbook Volume II, "Latent Faults", DOT/FAA/CT-88/10, 1988.

McGough, John G. and Fred L. Swern, Measurement of Fault Latency in a Digital Avionic Mini Processor, NASA CR 3462, Bendix Corporation, October 1981.

McGough, John G. and Fred L. Swern, Measurement of Fault Latency in a Digital Avionic Mini Processor, Part II, NASA CR 3651, NASA Langley Research Center, January 1983.

McGough, J. G., Swern, F., and S. Bavuso, "New Results in Fault Latency Modeling", Proceedings of the IEEE EASTCON Conference, pp. 299-306, August 1983.

Meissner, C. W., Jr., J. R. Dunham, and G. Crim, "NASA-LaRC Flight-Critical Digital Systems Technology Workshop", NASA Conference Publication 10028, NASA Langley Research Center, April 1989.

Migneault, Gerard E., "On the Diagnostic Emulation Technique and Its Use in the Airlab", NASA Technical Memo 4027, NASA Langley Research Center, October 1988.

MIL-HDBK-217E, "Reliability Prediction of Electronic Equipment", Distribution Statement A, DOD/AMSC, October 1986.

Nichols, W., et al., "Trends in Parts Susceptibility to Single Event Upset", IEEE Transactions Nuclear Science, vol. NS-32, no. 6, December 1985.

Padilla, Peter A., "In-Circuit Fault Injector User's Guide", NASA Technical Memo 100478, NASA Langley Research Center, June 1987.

Padilla, Peter A., "FTMP Data Acquisition Environment", NASA Technical Memo 100636, NASA Langley Research Center, July 1988.

Padilla, Peter A., "Fault Recovery Characteristics of the Fault Tolerant Multi-Processor", NASA Memo L-16630, NASA Langley Research Center, to be published as a NASA report 1990.

Rasch, N., User's Manual for AC-20-53A Protection of Airplane Fuel Systems Against Fuel Vapor Ignition Due to Lightning, DOT/FAA/CT-88/3, FAA Technical Center, October 1984.

Rothman, Elizabeth, et al., HARP: The Hybrid Automated Reliability Predictor, Introduction and Guide for Users, Version 6, Department of Computer Science, Duke University, NASA Langley Research Center, April 1989.

SAE Committee AE4L, Recommended Draft Advisory Circular: Protection of Aircraft Electrical/Electronic Systems Against the Indirect Effects of Lightning-Induced Transients, Report AE4L-87-3, Revision A, October, 1988.

Schuette, M., et al., "Experimental Evaluation of Two Concurrent Error Detection Schemes", 16th International Symposium of Fault Tolerant Computing, pp. 138-143, 1986.

Smith, T. Basil, III, and Jaynarayan H. Lala, Development and Evaluation of a Fault-Tolerant Multiprocessor (FTMP) Computer, Volume I, FTMP Principles of Operation, NASA CR 166071, NASA Langley Research Center, May 1983.

Strickland, Michael J. and Daniel L. Palumbo, Fault Tolerant System Performance Modeling, AIAA, Boeing, 1988.

Trivedi, Kishor S. and Robert M. Geist, A Tutorial on the CARE III Approach to Reliability Modeling, NASA CR 3488, Duke University, Prepared for NASA Langley Research Center, 1981.

NASA Conference Publication 2114, Working Group Meeting I, "Validation Methods for Fault-Tolerant Avionics and Control Systems", NASA Langley Research Center, March 1979.

NASA Conference Publication 2114, Working Group Meeting II, "Validation Methods Research for Fault-Tolerant Avionics and Control Systems", NASA Langley Research Center, December 1979.

Yang, X., W. York, and D. Siewiorek, "Fault Recovery of Triplicated Software on the Intel iAPX 432", Distributed Computing Systems, pp. 438-443, 1985.

## GLOSSARY OF TERMS

BYZANTINE RESILIENCE. A fault tolerant process which is tolerant of intermittent faults that can send good information part of the time.

COVERAGE. The conditional probability of the system successfully recovering from a component fault and continuing to perform the intended functions correctly, given the presence of the fault. Coverage is the measure of effectiveness of a system's utilization of redundant hardware. Coverage can be qualified and applied to many different components of a system and phases of recovery process. Examples include, fault detection coverage, fault isolation coverage, latent fault coverage, sensor failure coverage, and memory failure coverage.

ELECTROMIGRATION. Drifting of metal atoms toward the cathode of a cathode ray tube.

FAILURE. The deviation of system behavior from specifications (arithmetic failure, storage failure, flight control function failure.)

FAILURE, HARD. Repeated use of the same input and initial conditions results in the same incorrect response.

FAILURE MECHANISM. Any situation that could produce an error condition. Examples of failure mechanisms include metal migration, voltage overstress, and lack of air-conditioning.

FAILURE, PERMANENT. Repeated use of the same input and initial conditions results in the same incorrect response.

FAILURE, SOFT. Repeated use of the same input and initial conditions does not result in the same incorrect response.

FAILURE, TEMPORARY. Repeated use of the same input and initial conditions does not result in the same incorrect response.

FAILURE, TRANSIENT. Repeated use of the same input and initial conditions does not result in the same incorrect response.

FAULT. The phenomenological reason for a failure (open wire, stuck-at fault, design fault, etc.). In general, any condition preventing a digital component from correctly changing state when directed to change by input parameters. For electrical components there is a one-to-one correspondence between faults and failures. The situation is not so simple with digital circuits. For if the circuit is S-A-1, any input causing a one output will be correctly processed; a little like the stopped clock that is correct twice per day. For a processor

having a million or so logic gates, it is not possible to test for all the combinations of input and output states.

FAULT, LATENT. A fault which has not yet caused a failure. (For example, a fault in a memory chip that is not being used for the foreground program or in this particular mode of the system is a latent fault.)

FAULT, STUCK-AT. A logic signal which remains at zero (S-A-0) or one (S-A-1).

FAULT TOLERANT SYSTEM. A system that continues to function although certain components may have faults.

SYSTEM RELIABILITY. The probability of performing a given function from the some initial time, t=0, to time t.

## ACRONYMS AND ABBREVIATIONS

| | |
|---|---|
| AC | Advisory Circular |
| AE4L | SAE Subcommittee (Lightning) |
| AIAA | American Institute of Aeronautics and Astronautics |
| AIRLAB | Avionics Integration Research Laboratory |
| ALU | Arithmetic Logic Unit |
| AM | Amplitude Modulated |
| AMSC | Document number prefix used by the Department of Defense |
| ASEE | American Society of Electrical Engineers |
| ASME | American Society of Mechanical Engineers |
| ATTR | Attribute |
| B-GLOSS | Gate Logic Software Simulator developed by Bendix |
| C/I | Communicator Interstage |
| CAD | Computer Aided Design |
| CAP | Collins Application Processor |
| CARE III | Computer Aided Reliability Evaluator |
| CBD | Commerce Business Daily |
| CMOS | Complementary Metal-Oxide Semiconductor |
| CPA | Central Processor - A |
| CPU | Central Processing Unit |
| CR | Contractor Report |
| CSDL | Charles Stark Draper Laboratory |
| CTA | Collins Test Adaptor |
| DE | Diagnostic Emulation |
| DEFN | Definition |
| DEV | Development |
| DMA | Direct Memory Access |
| DOD | Department of Defence |
| EEC | Electronic Engine Control |
| EM | Electromagnetic |
| EMI | Electromagnetic Interference |
| EMR | Electromagnetic Radiation |
| ESS | Electronic Switching System |
| EXCHNG | Exchange |
| EXP | Experiment |
| FAA | Federal Aviation Administration |
| FCC | Flight Control Computer |
| FCR | Fault Containment Regions |
| FCS | Flight Control System |
| FET | Field Effect Transistor |
| FI | Fault Insertion circuitry |
| FIAT | Fault Injection Automated Testing |
| FIIS | Fault Insertion and Instrumentation System |
| FIM | Fault Injection Manager |
| FIRE | Fault Injection Receptor |
| FM | Frequency Modulated |

| | |
|---|---|
| FMECA | Failure Modes and Effects Criticality Analysis |
| FT | Fault Tolerant |
| FTC | Fault Tree Compiler |
| FTMP | Fault Tolerant Multiprocessor |
| FTP | Fault Tolerant Processor |
| GE | General Electric |
| GEN | Generation |
| GGLOSS | Generalized Gate-Level Logic System Simulator |
| GLOSS | Gate Logic Software Simulator |
| GPC | General Purpose Computer |
| HARP | Hybrid Automated Reliability Predictor |
| HDBK | Handbook |
| HERF | High Energy Radio Frequency |
| I/O | Input/Output |
| IBM | International Business Machines |
| ICIS | Intercomputer Interface Sequencer |
| ICS | Intercomputer Sequencer |
| ID | Identification |
| IEEE | Institute of Electrical and Electronics Engineers |
| IGGLOSS | Gate Logic Software Simulator (improved version developed at NASA Langley) |
| IOPA | Input/Output Processor - Channel A |
| IOS | Input/Output Subsystem |
| JPL | Jet Propulsion Laboratories |
| KHz | KiloHertz |
| LAN | Local Area Network |
| LaRC | Langley Research Center |
| LRU | Line Replaceable Unit |
| LSI | Large Scale Integration |
| M | Mutual (when used with RLC) |
| MDICU | Modular Digital Interface Conversion Unit |
| MHz | MegaHertz |
| MIL | Military |
| MOS | Metal-Oxide Semiconductor |
| MPX | Multiplex |
| MSI | Medium Scale Integration |
| Mux | Multiplexed |
| Naecon | National Avionics and Electronics Conference |
| NAND | Not AND |
| NASA | National Aeronautics and Space Administration |
| NEMP | Nuclear Electromagnetic Pulse |
| OS | Operating System |
| PAL | Programmable Array Logic |
| PAWS | Padé Approximation With Scaling |
| PC | Personal Computer |
| PMS | Physical Message Switch |
| PROC | Processor |
| QUAD | Quadruple |
| RAM | Random Access Memory |
| RCVR | Receiver |
| RDFCS | Reconfigurable Digital Flight Control System |
| REG | Register |

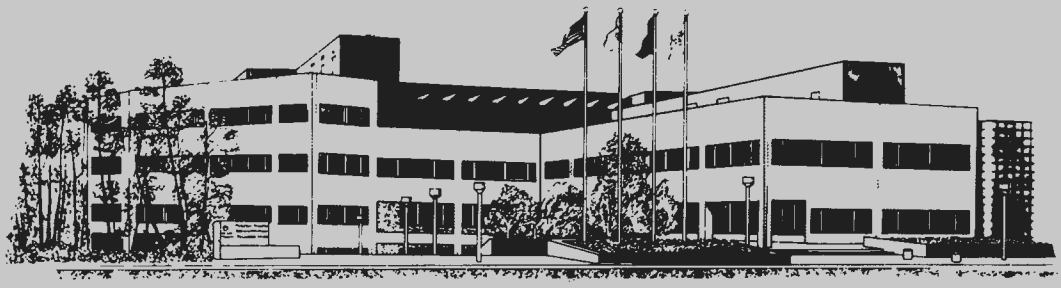
| | |
|---|---|
| RF | Radio Frequency |
| RLC | Resistance/Inductance/Capacitance |
| ROM | Read Only Memory |
| RT | Remote Terminal |
| RTI | Remote Terminal Interface |
| S-A-0 | Stuck-At-Zero |
| S-A-1 | Stuck-At-One |
| S-GLOSS | Gate Logic Software Simulator developed by Stevens Institute |
| SAE | Society of Automotive Engineers |
| SEU | Single Event Upset |
| SHRD | Shared |
| SQL | Software Query Language |
| SSI | Small Scale Integration |
| STEM | Scaled Taylor Expansion Matrix |
| SURE | Semi-Markov Unreliability Range Evaluator |
| SYN | Synch |
| TTL | Transistor-Transistor Logic |
| TX | Transmit |
| UNIBUS | Universal Bus |
| USEG | Unsegmented |
| VLSI | Very Large Scale Integration |
| XAB | Transmit Compare A B |
| XMT | Transmit |
| XOR | Exclusive OR |

U.S. Department of Transportation
**Federal Aviation Administration**

DOT/FAA/CT-88/10

# HANDBOOK-VOLUME II
# DIGITAL SYSTEMS VALIDATION

## CHAPTER 6
## DIGITAL DATA BUSES FOR AVIATION APPLICATIONS



**PREPARED BY:**

**COMPUTER RESOURCE MANAGEMENT, INC.**
**950 HERNDON PARKWAY, SUITE 360**
**HERNDON, VIRGINIA 22070**

**PREPARED FOR:**

**FEDERAL AVIATION ADMINISTRATION**
**TECHNICAL CENTER**
**ATLANTIC CITY INTERNATIONAL AIRPORT, NEW JERSEY 08405**

NOTICE

This document is disseminated under the sponsorship of the U.S. Department of Transportation in the interest of information exchange. The United States Government assumes no liability for the contents or use thereof.

The United States Government does not endorse products or manufacturers. Trade or manufacturers' names appear herein solely because they are considered essential to the objective of this report.

# PREFACE

Digital data buses are used extensively in the current generation of civil aircraft. These buses are used in flight control and avionics applications to transfer data and to perform complex calculations. At the present time, the Federal Aviation Administration (FAA) has no published criteria or procedures for evaluating these complex systems, other than the general characteristics described by Federal Acquisition Regulation (FAR) 25.671, FAR 25.901, FAR 25.1309, and special conditions established by each unique aircraft type certification board.

This gap is being filled in two ways. First, standards have been developed for each major type of data bus, with the consequence that evaluation of a candidate data bus architecture can take place by comparing the "as-built" functional characteristics and system operation against the requirements published in the standards. Second, data collection, a necessary part of implementing digital data buses in new and derivative aircraft, has been an ongoing activity for the last 3 to 5 years. Information, data, and "lessons learned" from ongoing inspection, analysis, and documentation activities associated with certification testing for the Boeing 757/767 and other aircraft which use Aeronautical Radio, Incorporated (ARINC) 429, Military Standard (MIL-STD) 1553B, and General Aviation Manufacturers Association (GAMA) Avionics Standard Communication Bus (ASCB) data buses are providing guidelines and criteria for evaluation and certification of the next generation of aircraft.

This body of standards and experience is the topic of this tutorial. The first two sections ("Introduction" and "Data Bus Architecture and Topology") provide an overview of major characteristics of data buses in general. These sections include necessary background information for understanding data buses and why they are used. Descriptions of major characteristics applicable to all data buses are also included.

The third section ("Current Aviation Buses"), summarizes current and accepted standards and requirements for five of the most commonly used aviation-oriented data buses. This section is organized to enable efficient access to specifications required to evaluate documentation presented for data bus certification. The final two sections of the tutorial ("Bus Performance Considerations" and "Fiber Optic Data Bus for Avionics Integration") address major performance requirements, applicable to all data buses, which are not currently covered by the standards. These performance requirements cover data latency, failure modes, and factors related to the use of fiber optics.

TABLE OF CONTENTS


| Section | | | Page |
|---|---|---|---|
| 1. | INTRODUCTION | | 6-1 |
| | 1.1. | Background | 6-1 |
| | 1.2. | Bus Requirements | 6-4 |
| 2. | DATA BUS ARCHITECTURE AND TOPOLOGY | | 6-7 |
| | 2.1. | Introduction | 6-7 |
| | 2.2. | The Individual Component | 6-7 |
| | 2.3. | Single Bus Architectures | 6-9 |
| | 2.4. | The System Level | 6-11 |
| | 2.5. | The Choice of an Architecture or Topology | 6-18 |
| | 2.6. | Protocol/Topology Relationships | 6-18 |
| 3. | CURRENT AVIATION BUSES | | 6-23 |
| | 3.1. | Introduction | 6-23 |
| | 3.2. | ARINC 429 | 6-23 |
| | | 3.2.1. History | 6-24 |
| | | 3.2.2. ARINC 429 General Characteristics | 6-24 |
| | | 3.2.3. ARINC 429 Protocol: Command Response | 6-25 |
| | | 3.2.4. ARINC 429 Hardware Characteristics | 6-26 |
| | | 3.2.5. Major Documents | 6-26 |
| | 3.3. | MIL-STD-1553B and MIL-STD-1773 | 6-26 |
| | | 3.3.1. History | 6-26 |
| | | 3.3.2. MIL-STD-1553B General Characteristics | 6-27 |
| | | 3.3.3. MIL-STD-1553B Protocol: Time Division Multiplex, Command/Response | 6-29 |
| | | 3.3.4. MIL-STD-1553B Hardware Characteristics | 6-30 |
| | | 3.3.5. MIL-STD-1773 Hardware Characteristics | 6-30 |
| | | 3.3.6. Major Documents | 6-32 |
| | 3.4. | The GAMA ASCB | 6-33 |
| | | 3.4.1. History | 6-33 |
| | | 3.4.2. GAMA ASCB General Characteristics | 6-33 |
| | | 3.4.3. GAMA ASCB Protocol: HDLC | 6-34 |
| | | 3.4.4. GAMA ASCB Hardware Characteristics | 6-34 |
| | | 3.4.5. Major Documents | 6-34 |
| | 3.5. | DATAC (ARINC Specification 629) Bus | 6-35 |

TABLE OF CONTENTS (Continued)


| Section | | Page |
|---|---|---|
| | 3.5.1. History | 6-35 |
| | 3.5.2. DATAC General Characteristics | 6-36 |
| | 3.5.3. DATAC Protocol: Collision Detection | 6-37 |
| | 3.5.4. DATAC Hardware Characteristics | 6-38 |
| | 3.5.5. Major Documents | 6-38 |
| 3.6. | SAE High-Speed Data Buses | 6-39 |
| | 3.6.1. History | 6-39 |
| | 3.6.2. SAE General Characteristics | 6-40 |
| | 3.6.3. SAE Protocol: Token Passing | 6-40 |
| | 3.6.4. SAE Hardware Characteristics | 6-42 |
| | 3.6.5. Major Documents | 6-45 |
| 4. | BUS PERFORMANCE CONSIDERATIONS | 6-47 |
| 4.1. | Introduction | 6-47 |
| 4.2. | Issues of Latency | 6-47 |
| | 4.2.1. Types of Latency | 6-47 |
| | 4.2.2. Data Latency | 6-47 |
| | 4.2.3. System Delays | 6-50 |
| 4.3. | Failure Modes and Effects | 6-51 |
| 4.4. | Transmission Error | 6-53 |
| 4.5. | Reliability for Flight Safety | 6-59 |
| 5. | FIBER OPTIC DATA BUS FOR AVIONICS INTEGRATION | 6-61 |
| 5.1. | Elements of Fiber Optic Data Bus | 6-61 |
| 5.2. | Optical Cables | 6-62 |
| 5.3. | Optical Path | 6-62 |
| 5.4. | Splices and Connectors | 6-62 |
| 5.5. | Optical Couplers | 6-62 |
| 5.6. | Size | 6-63 |
| 5.7. | Reflections | 6-63 |
| 5.8. | Connectors | 6-63 |
| 5.9. | Splicing | 6-64 |
| 5.10. | Technology - Optical Bus Transmitters and Receivers | 6-64 |
| 5.11. | Maximizing Bus Efficiency | 6-64 |
| 5.12. | Transmission Losses | 6-64 |
| 5.13. | Receiver Losses | 6-65 |
| 5.14. | Transmitter Losses | 6-65 |
| 5.15. | Optical T/R Power Margin | 6-66 |
| 5.16. | Topology Analysis | 6-66 |
| 5.17. | Fiber Optic Network Based Losses | 6-68 |

TABLE OF CONTENTS (Continued)


| Section | Page |
|---|---|
| APPENDIX | |
| A - MODULATION AND SIGNALING METHODS | 6-69 |
| BIBLIOGRAPHY | 6-74 |
| GLOSSARY | 6-76 |
| ACRONYMS AND ABBREVIATIONS | 6-78 |

LIST OF ILLUSTRATIONS


| Figure | | Page |
|---|---|---|
| 2.1-1 | INDIVIDUAL SUBSYSTEM STRUCTURE | 6-8 |
| 2.3-1 | SINGLE BUS ARCHITECTURES | 6-10 |
| 2.4-1 | TYPICAL AVIONIC HIERARCHIAL AND PARALLEL ARCHITECTURES | 6-12 |
| 2.4-2 | PRELIMINARY ARCHITECTURE OVERVIEW | 6-13 |
| 2.4-4 | INTEGRATION CONCEPTS FOR FLIGHT CRITICAL SUBSYSTEMS | 6-17 |
| 3.2-1 | GENERIC ARINC BUS LAYOUT | 6-24 |
| 3.3-1 | GENERIC MIL-STD-1553B BUS LAYOUT | 6-28 |
| 3.4-1 | GENERIC ASCB BUS LAYOUT | 6-33 |
| 3.5-1 | GENERIC DATAC BUS LAYOUT | 6-36 |
| 3.6-1 | GENERIC SAE LINEAR BUS LAYOUT | 6-39 |
| 3.6-2 | GENERIC SAE RING BUS LAYOUT | 6-39 |
| 4.2-1 | DATA LATENCY ILLUSTRATION IN HIERARCHIAL ARCHITECTURE | 6-49 |
| 4.3-1 | SIMPLE FAULT TREE EXAMPLE | 6-55 |
| 4.5-1 | SINGLE CHANNEL-DUAL OUTPUT (BUSES A AND B) BUS CONTROL | 6-59 |
| A-1 | AMPLITUDE MODULATION | 6-70 |
| A-2 | PHASE MODULATION | 6-70 |
| A-3 | WAVEFORMS GENERATED BY FIVE DIFFERENT ENCODING SCHEMES FOR A FIXED BINARY SIGNAL | 6-73 |

LIST OF TABLES


| Table | | Page |
|---|---|---|
| 1.1-1 | SUMMARY OF BUS CHARACTERISTICS | 6-3 |
| 2.4-1 | COMPARISON OF HIERARCHICAL AND PARALLEL AVIONICS BUS ARCHITECTURES | 6-15 |
| 2.5-1 | MEASURES OF MERIT | 6-19 |
| 2.5-2 | ATTRIBUTES | 6-20 |
| 2.6-1 | ALTERNATIVE TOPOLOGIES AND PROTOCOLS | 6-21 |
| 3.2-1 | ARINC 429 HARDWARE CHARACTERISTICS | 6-27 |
| 3.3-1 | MIL-STD-1553B HARDWARE CHARACTERISTICS | 6-31 |
| 3.3-2 | MIL-STD-1773 HARDWARE CHARACTERISTICS | 6-32 |
| 3.4-1 | GAMA ASCB HARDWARE CHARACTERISTICS | 6-35 |
| 3.5-1 | BOEING DATAC BUS CHARACTERISTICS | 6-38 |
| 3.6-1 | SAE RING BUS HARDWARE CHARACTERISTICS | 6-43 |
| 3.6-2 | SAE LINEAR BUS HARDWARE CHARACTERISTICS | 6-44 |
| 4.3-1 | DIGITAL DATA BUS SYSTEM FAILURE MODES AND EFFECTS | 6-54 |
| 4.4-1 | FAILURE CLASSES [MIL-STD-1553B] | 6-56 |
| 4.4-2 | TYPICAL ERROR-CORRECTION TECHNIQUES [MIL-STD-1553B] | 6-57 |
| 4.5-1 | POTENTIAL FAILURES RESULTING IN LOSS OF BUS CONTROL SINGLE CHANNEL - DUAL OUTPUT (BUSES A AND B) | 6-60 |
| 5.1-1 | OPTICAL BUS TECHNOLOGY LIMITS | 6-61 |
| 5.17-1 | STAR-COUPLED NETWORK LOSSES | 6-68 |
| A-1 | IMPORTANT PARAMETERS OF ENCODING TECHNIQUES | 6-72 |

# 1. INTRODUCTION

## 1.1. Background

Advances in digital technology have made possible greater processing power and memory capacity in smaller and smaller packages (Schmitter and Baues, 1984). Reductions in size mean that this processing capability can be located where it is needed, thus supporting the distribution of intelligence throughout the system network. In the specific case of modern civil aircraft, distributed intelligence has meant increasing numbers of distinct computer-controlled avionics, flight control, propulsion, and structural systems, with corresponding explosive growth in the volume of data being processed throughout the system.

As data processing becomes increasingly distributed throughout the aircraft, the need to support integration of this data and to allow access to it, as required, by both central and remote processors becomes apparent. Microprocessors, sensors, and servomechanisms must be appropriately interconnected to support communication and to enable fault tolerant designs to be implemented. Integrating aircraft subsystems offers numerous advantages, including the following (McSharry, 1983):

- Reduced crew workload through such capabilities as autoland and autopilot.
- Enhanced aircraft performance and capability.
- Increased hardware efficiencies (and resulting decreases in cost) through reduction in hardware duplication, complexity, and function.
- Improved flight safety through features such as flight envelope limiters.

Maximum use of common data allows the development of functional capabilities otherwise not possible. In addition, a common interface enables improvements in individual subsystems and flight tasks to be made with minimal disturbance of higher-order information transfer functions. Enhanced modularity simplifies subsystem interconnection so that retrofitting can be accomplished without the cost and complexity associated with modifying the entire system. Maintenance also becomes easier since dynamic testing of installed equipment can take place through the data bus interface by means of standardized maintenance equipment (Kiernan and Sims, 1980).

Prior to the mid-1970s, discrete subsystems passed information back and forth through analog point-to-point wire bundles. Traditionally, there has been considerable independence in the design of these subsystems. Components such as sensors were provided separately for each subsystem. However, advanced aircraft designs often require that these systems have significant interaction and a common data source. The combination of the need to integrate these

systems by function and to avoid unnecessary duplication of hardware provides the impetus for developing integration techniques and supporting architectures, which both reduce the overall costs and increase performance.

The analog wire approach could not match the demands for distributed processing. It increased weight (due to the large quantity of wiring required) and economic costs, while decreasing rapid access to a database shared by all subsystems. The system designers had to develop an alternative approach: the data bus. A data bus is a system for transferring data between discrete pieces of equipment in the same complex.

Commercial and transport aircraft currently employ only single-level data bus architectures. These architectures use one of the following: (1) centralized control, (2) bidirectional information transfer system, or (3) direct-connect, unidirectional information transfer system. Microprocessor-based flight control and avionics systems (as represented by the Boeing 757/767, the Lockheed L1011-500, and the Airbus A310/A320) use data bus architectures based on either the ARINC 429-5 or the MIL-STD-1553A/B specification and standards. These architectures use shielded, twisted pair wires for the transmission medium. They interconnect microprocessors that primarily use bit slice processors. The microprocessors provide the required internal processing speed (7-14 Megahertz (MHz) clock rate) and the inherent reliability, stability, and flexibility required for flight essential and flight critical control systems. In this generation of digital systems, the individual processors are run in a bit or frame synchronized manner. The data are exchanged between redundant computers via dedicated serial data buses (either wire or fiber optic). Internally, the data are exchanged by high-speed dedicated transfer buses/backplanes.

The extensive use of existing bus structures has proven that multiplexed data transfer systems can achieve a degree of integration. The present bus characteristics are ideally matched to many intraavionics subsystem data transfer requirements. These requirements include sensor data collection, central processing, and distribution of results to peripheral areas. Continued use of bus networks for the intrasubsystem data transfer will be required.

Unfortunately, current protocols and architectures may not provide the characteristics needed to support future overall system integration requirements. The next generation aircraft will have multiple information transfer systems which require interchange of data. These systems will communicate with one another through global memory storage interface units. Consequently, such aircraft will require total airframe/system integration (on a full-time/full authority basis). With subsystems integrated in this manner, a negative change in one could possibly result in erroneous data and information being propagated throughout the entire system.

A solution to this potential problem is the development and use of an information transfer system which will efficiently interconnect, in a hierarchical order, multilevel, multiplexed buses and bus architectures. With such an approach, software-intensive, fault-tolerant executive and operating systems can be created. These operating systems provide the processing of functions required of multisystem inputs using the local terminals. A high-speed, higher-order transfer system will probably employ contention or token-passing

protocols. These protocols provide each active unit (within the information transfer system structure) the capability of performing its own data processing autonomously. The responsibility for passing completed data onto the bus at the appropriate time belongs to each active unit.

Some of the more common subsystems are likely to be combined into logical units (components). New subsystems or groups of architecturally related functions will be implemented as common units. Each of the major systems will probably be designed as an integral unit. Each unit will have its own unique intra-multiplexed topological (bus) network. Each asynchronous information transfer function and topological network must be interconnected, using high bandwidth buses to create integrated data and management bases. Information flow can be directed and managed from these bases.

Data buses currently in use, such as ARINC 429, MIL-STD-1553B, and GAMA ASCB do not have the capability to handle such massive integration (see table 1.1-1). Consequently, new approaches must be developed for the interconnection of avionic subsystems to ensure the integrity of the data at all times. New bus standards are being developed to support higher-order data. The new standards use operational protocols that provide high-speed interconnection of subsystems and common sensors, sufficient subsystem independence, and fault tolerance. These protocols also distribute control of the common data bus at both the subsystem black box level and the aircraft/application level.

TABLE 1.1-1. SUMMARY OF BUS CHARACTERISTICS

| | ARINC 429 | MIL-STD-1553B | GAMA ASCB | (DATAC) ARINC 629 | SAE Linear/ Ring |
|---|---|---|---|---|---|
| Maximum Bit Rate (Hz) | 100K | 1M | 667K | 1M-2M | 10M-20M/ 50M |
| Bidirectional | No | Yes | Yes | Yes | Yes |
| Bus Controllers | No | Yes | Yes | No | No |
| Defined Data Formats | Yes | No | Yes | Yes | Yes |
| Low Cost Components | No | No | Yes | Yes<br>No | No |

The next generation of flight control and avionics systems architectures will be dramatically different from the current generation. It will include multiple microprocessors in each computing channel, more local processing within a processor, and the transfer of preprocessed data within the bus network. In addition, the system architectures will make use of 16/32 bit microprocessors,

which will use high-speed backplane buses (running at 20-50 MHz) for internal (processor-to-processor) interfaces and exchange of data and information. Furthermore, these processors and their fault-tolerant designs will make use of global memory and functional partitioning of executive and applications software to decrease the complexity and increase the reliability of the system.

Although such objectives have yet to be realized, two data bus standards, soon to be in commercial use, will allow incremental progression towards these goals: the Digital Autonomous Terminal Access Communication (DATAC) (ARINC 629) and the Society of Automotive Engineers (SAE) linear and ring bus architectures (see table 1.1-1). These buses will be able to support the integration of avionics, flight, and propulsion systems expected to be implemented digitally in current and future aircraft. This method of implementation will allow the necessary sharing of data between subsystems.

Maximizing data availability between subsystems is in conflict with the need to isolate these systems from propagation of failures from one system to another. Therefore, the integration solution must consider the balance between the need for and type of integration, versus the flight-safety and application-criticality of each subsystem. The solution must also consider various architectural implementations within the different aircraft configurations and applications.

## 1.2. Bus Requirements

The overall advantage of integrating flight-critical subsystems (flight and propulsion controls) with other avionics subsystems can be realized only if efficient, safe, and practical methods of subsystem communication can be implemented. Items to be considered in implementing subsystem communication include the following:

- Architecture topology.
- Bus interface design.
- Interaction with the host processor (controller) and data bus interface.
- Bus protocol.
- Hardware/software failure modes.
- Fault propagation potential.
- Protection mechanisms that prohibit fault introduction or allow detection and management of faults.

The trend in data bus evolution is to implement the following general characteristics:

- Real-time data integration - current data are essential to critical flight operations.

- Protection of subsystems from propagation of failures from other subsystems. There must be an appropriate balance between integration and isolation.

- Decentralized system control.

- Standard interfaces for all equipment connected to the bus.

- Continued reductions in the use of costly hardware/software elements required in centrally controlled, data transfer systems.

- Continued dispersion of microprocessors within subsystems necessitating the interchange of processed data between subsystems.

- Generation of an aircraft database, available to all subsystems, which includes all airframe/performance parameters.

- Maximal use of common sensor data and redundant data sources.

- Further standardization of hardware/software elements by the use of other standards for interchangeability between the avionic systems and aircraft.

This tutorial examines the characteristics of avionics buses in use today. It also reviews two bus concepts proposed for use in the near future. While none of these buses has all the qualities desired for the next generation, the newer network buses offer the greatest potential match with the technology anticipated for the next generation of airframes and avionics.

Data and information for avionics systems integration can be successfully transmitted using these existing or other proposed bus structures. However, each bus has limitations, which must be considered when assessing its suitability for use in a specific application.

Before addressing the specific performance characteristics of the five buses, the topic of bus architecture and topology needs to be considered to:

- Identify the influence of physical structure on data bus performance.

- Understand the impact of dispersed locations of applications processors on computational capability and system flexibility.

# 2. DATA BUS ARCHITECTURE AND TOPOLOGY

## 2.1. Introduction

Data bus architecture and topology refers to the physical layout of the bus, (i.e., the organization of individual components attached to the buses). There are three levels of organization:

- Individual component.

  The primary purpose of a data bus is to allow individual components to transmit and receive relevant data. Each component performs a function. During the time it performs the function, it may need to receive data from or send data to one or more other components. For example, a Flight Control Computer (FCC) requires sensor data as input, performs a set of functions or equations using that data, and then sends commands to actuators or other avionics systems. Each component, therefore, requires an interface compatible with the physical structure of the bus.

- Individual bus.

  An individual bus is a topological configuration that allows the exchange of information between two or more individual components. The individual bus can support a minimum of two components up to the maximum capacity authorized by the bus architecture or structure. A specific bus may integrate related subsystems, which perform similar tasks ("the navigation bus") or integrate all subsystems ("the system bus"). Consequently, the complexity of individual buses can vary.

- System level.

  When multiple buses are used, a system level is required to integrate the individual buses. The extent of this integration depends on the degree of overall system integration required and the demands of the overall architectural design. The system level bus ensures that data are distributed to all levels that need the data at the proper time.

Each of these levels is described in more detail below.

## 2.2. The Individual Component

The generic structure of the bus/box interface is shown in figure 2.1-1.

Bus isolation refers to the physical attachment of the component to the bus medium. Typically, this attachment is accomplished by either direct coupling or transformer coupling. Direct coupling is used with short lengths of wire (normally one foot or less) with the result that impedance mismatches are not

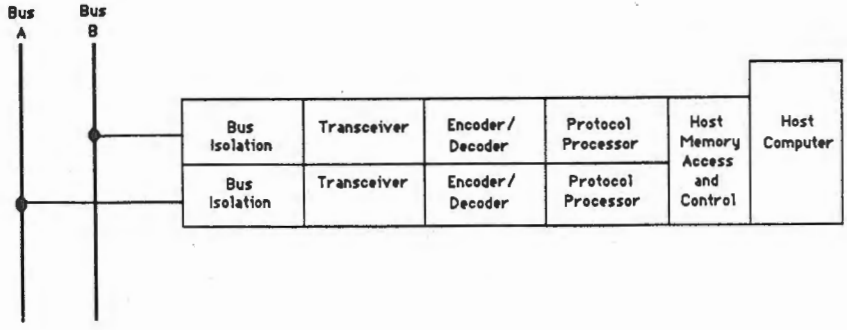



FIGURE 2.1-1. INDIVIDUAL SUBSYSTEM STRUCTURE

critical. Transformer coupling is used with longer lengths of wire (up to 20 feet) to achieve the specified signal-to-noise ratio and system error rate performance, and to maintain the proper impedance balance on both sides of the transformer over the longer lengths of connecting cable.

The transceiver is a combined transmitter and receiver. It is responsible for the interface between the digital data provided by the component and the analog format of the bus. The receiver element also has the task of monitoring the data sent along the bus until it detects "its name". Information intended for the component is detected and picked up by the receiver. The transmitter, in contrast, places information on the bus. Additional functions of the transceiver are to provide clock generation (ensuring that data are introduced to the bus at the correct time) and to evaluate received data for errors.

The encoder/decoder converts the digital data provided by the component into an analog format that is compatible with the bus. It also converts the analog data removed from the bus into a digital format. Since data buses support only serial data transfer while the processors expect parallel data, the encoder/decoder must provide serial/parallel data conversion.

The protocol processor adds or removes protocol information (such as error correcting codes) from transmitted messages. The processor is also responsible for checking the accuracy of the data coming from the bus. In evaluating data transmission protocols, the error detection and correction techniques, which could be used by the protocol processor, include Vertical Redundancy Check (VRC), Longitudinal Redundancy Check (LRC), and Cyclic Redundancy Check (CRC).

VRC appends one additional overhead bit (a "1" or a "0") to a data word to implement either odd or even parity. VRC does not detect double bit errors. LRC views a frame as a block of characters and appends an additional character consisting of the parity bit for each bit position in the character. Even when

used with VRC, some patterns of even number errors remain undetected. CRC generates a frame check sequence for a frame, which is exactly divisible by some predetermined number. The frame check sequence may be verified at both ends of the transmission. Only rare combinations of errors remain undetected with this system.

Forward error correction codes are used when the receiver alone corrects data errors. The codes are calculated and transmitted along with the data. For acceptable correction, data rates are reduced by at least 50 percent. Backward error correction (retransmission) is used to resend messages when the receiver signals the transmitter that an error occurred in the transmission.

In some cases, the protocol processor translates and implements the type of message (command, status, data) and the requirements for special actions. When a bus controller is not used, two protocols may be required. The first protocol is the data transmission protocol itself. The second gains access to the bus in order to transmit data.

Host memory access and control have the capability to interrupt the main processor as required, without waiting for the response cycle through a hard-wired Direct Memory Access (DMA) channel.

The host computer is the component that actually produces data. Its functions include computation, management, and instruction. Once the host computer has completed its functions, it may request data from the bus or send its own already-processed data to the bus.

## 2.3. Single Bus Architectures

To achieve orderly transmission and reception of data, rules must be established for individual components to gain access to the bus network. These rules depend on the basic topological structure of the bus network itself. To date, the following structures have been used:

- Point-to-point.
- Linear.
- Star.
- Ring.

Figure 2.3-1 (Schmitter and Bauss, 1984) shows three of the above structures. The point-to-point topology is not included in this figure. It is similar to the linear bus except that each component is individually connected (by one or more independent wires) to other components within the system. There is no competition in gaining access to the transmitting medium, because each component is physically linked by its own wire to every other component to which it transmits. On the other hand, if a hard-wire connection does not exist, the components cannot exchange data as they would in the case of a shared, common bus. With this concept, a single transmitter can broadcast to all receivers simultaneously as long as the receivers are connected to the same wire.

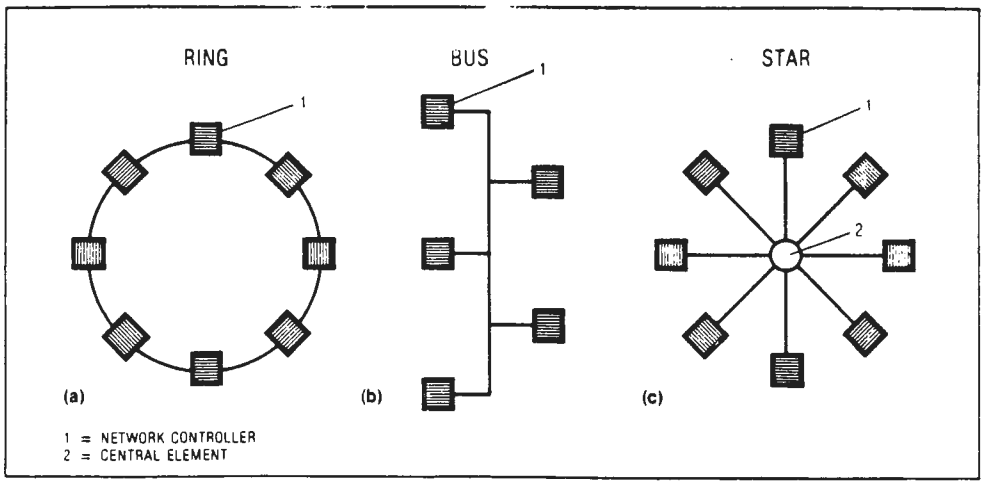



FIGURE 2.3-1. SINGLE BUS ARCHITECTURES
(from Schmitter and Baues, 1984)

With the linear bus concept, all components are attached to a common bus medium that contains multiplexed data operating in a half-duplex mode. A linear bus can be governed in either of two modes:

- By means of a centralized bus controller, which initiates information transfers to (from) specified components in a command (response) mode.
- Through decentralized operation using contention logic where a transmitter, which wishes to send data, waits until there is an opening on the bus.

The ring bus concept provides an easy method of joining together any number of stations, up to the specified limit of the addressing field, due to its use of simple point-to-point links.

With the ring bus, there are no restrictions with respect to the following:

- The minimum number of stations (above two).
- The physical or logical position of a station. (Note: the inherent token addressing dispenses with the need to have sequential addressing for speed.)

In addition, the point-to-point links allow the use of any medium defined in the specification - including the possible use of mixed media (i.e., wire and fiber optic).

The star bus topology always requires a central bus controller as the active element with peripheral processing accomplished by the other application processors (boxes) attached to the central element. Star topologies are

particularly suited for fiber-optic coupled systems as a result of the development of the star fiber optics couplers and the nature of the optics required.

## 2.4. The System Level

A single bus may integrate the processing of all the components comprising the vehicle system, or it may be used with components performing related functions. The former is an example of a parallel bus (see the right-hand side of figure 2.4-1). As an alternative, a hierarchical organization of multiple buses may be used (see the left-hand side of figure 2.4-1). Each set of components performing similar functions (e.g., navigation) is attached to the same bus. Integrated data produced from this bus is then passed on to a second bus, which brings together data from other low-level buses. By means of this hierarchical arrangement, progressive layers of buses, carrying ever higher levels of data, are formed. As an example, figure 2.4-2 provides a multilevel concept composed of four digital information transfer bus structures (Sensor, Management, Systems, Actuator) and one or more dedicated analog bus structures.

The sensor bus contains data that are time critical and necessary for critical system functions including:

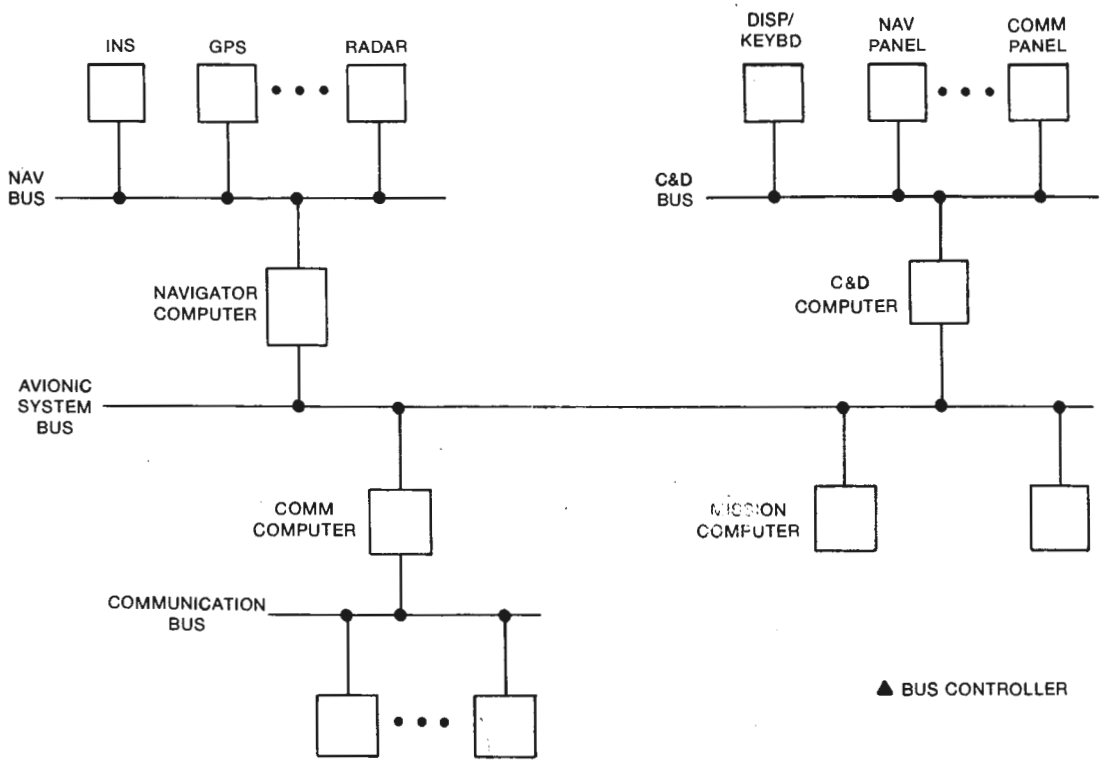
- Body accelerations and angular rates.
- Attitude angle and rates.
- Navigation and position (angles and deviations).
- Pilot inputs (column, wheel, throttle).
- Surface position (deflections and accelerations).

The data handled by the management bus are generally non-time-critical data that provide control information and system configuration. These data include the following:

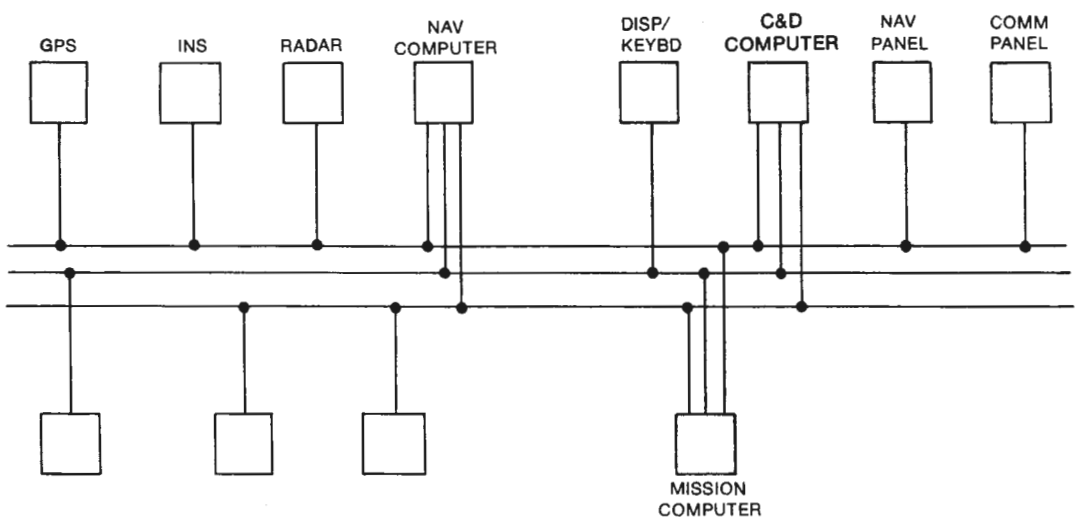
- Pilot selected parameters and modes.
- Initialization data.
- Reference angles.

The systems bus transfers time-critical data that are provided (by the aircraft avionics and flight controls systems) at a constant update rate to perform mission/flight-phase oriented and automatic functions. These data include the following:

- Auto-throttle position and rates.
- Autoload (deviations, deflections, and commands).
- Attitude reference/control.




Typical Avionic Hierarchical Architecture




Typical Avionic Parallel Architecture

FIGURE 2.4-1. TYPICAL AVIONIC HIERARCHIAL AND PARALLEL ARCHITECTURES

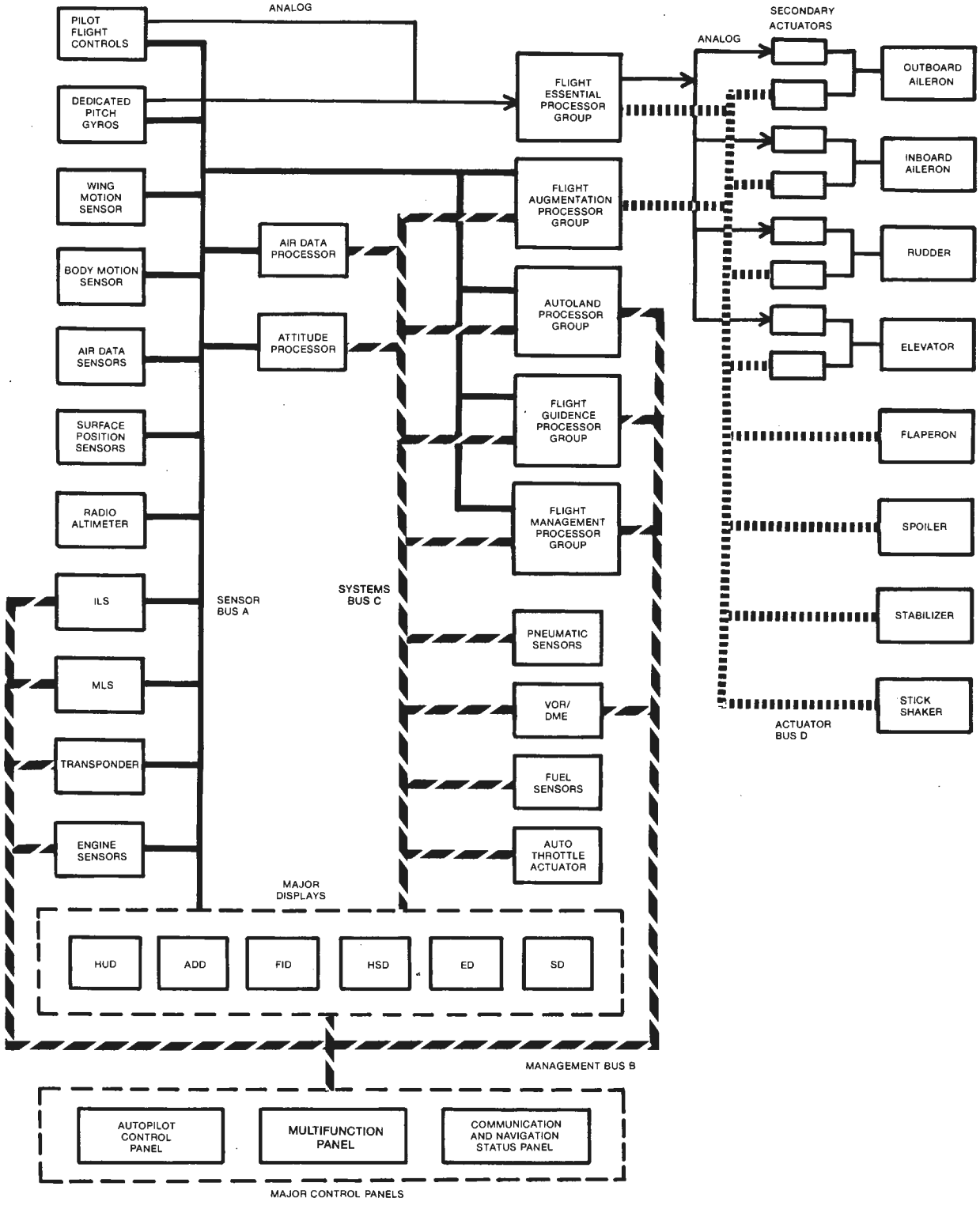



FIGURE 2.4-2. PRELIMINARY ARCHITECTURE OVERVIEW

- Flight management functions.
- Pneumatic (status/control).
- Fuel (flow/rate, quantities).

The actuator bus provides the necessary constant-update-rate data to command and feed control back to the surface controllers and tactile attitude warning devices. These data include the following:

- Deflection command/activator position (aileron, rudder, elevator, spoiler, stabilizer, etc.).
- Stability augmentation (gains/deflections).
- Stick shaker.

The analog (hard-wired) interconnections handle the flight-essential functions that include pitch rate sensors, pilot flight controls, and redundant actuators.

Early bus architectures used a parallel structure. Advances in digital technology make it possible to preprocess functions through distributed intelligence. Consequently, architectures more suitable to the sensors and actuators can be used. Onsite microprocessors can replace the central computer complex. As a result, the concept of hierarchically organized, multiple buses becomes attractive. One advantage of such an approach is that it enables easy differential treatment of critical and noncritical data simply by using different buses.

Both architectural arrangements are still in use. In general, hierarchical buses are more efficient. Data transmitted by hierarchical buses tends to receive more preprocessing than data transmitted via the parallel architecture. There is less chance of overloading system bus under hierarchical architectures. However, this also means that data tends to take longer to reach its final destination since it may have to pass through several buses.

Hierarchical buses support greater functional independence and isolation of individual subsystems. In contrast, parallel buses must carry more data but they offer a simpler architecture which allows greater reconfiguration capability.

Figure 2.4-3 (McSharry, 1983) provides two examples of the implementation of each architecture. In one case, integration is performed by means of a local bus, in the second by an avionics system bus. The avionics system bus sends out processed data while the local bus can transmit both processed and unprocessed data. The advantages and disadvantages of the hierarchical versus parallel approaches and the local bus versus system bus are summarized in table 2.4-1 (McSharry, 1983).

From the control system perspective, three integration alternatives are possible as shown in figure 2.4-4. All three cases make use of various combinations of

TABLE 2.4-1. COMPARISON OF HIERARCHICAL AND PARALLEL AVIONICS BUS ARCHITECTURES (adapted from McSharry, 1983)

| Hierarchical Avionics Bus Architecture | Parallel Avionics Bus Architecture |
|---|---|
| • Local Bus | • Single Bus |
| Minimum data latency | Simpler |
| Lowest intersystem impact | Greater flexibility |
| Greater isolation | |
| • Avionic System Bus | • Multiple Buses |
| Information required at more than one local bus | Higher reliability levels |
| Highest inter-/intra-system impact | |
| Greater data latencies | |

data bus structures (Sensor, Systems, Actuator, and Analog) previously referenced, and either use the FCC as a buffer between the avionics and flight control systems or connect directly to the control system bus(es) with other mission essential computers acting as buffers. In either case, the proposed architecture or topology and the attendant integration must be defined so that one of the following conditions exist.

- Isolation (in terms of fault propagation) is maximized by integration of functions and sensor signal requirements. This may be accomplished through the use of redundant avionics buses or buses dedicated to support avionics, flight control, and other mission dependent functions within the same bus structure. This approach, however, requires higher levels of system reliability to satisfy flight safety requirements.

- Data latency is minimized by use of separated structures in which critical sensor data co-exist with the flight control and mission dependent computation function on the same bus. It may also be minimized by making optional use of existing sensor redundancy with critical sensor data being

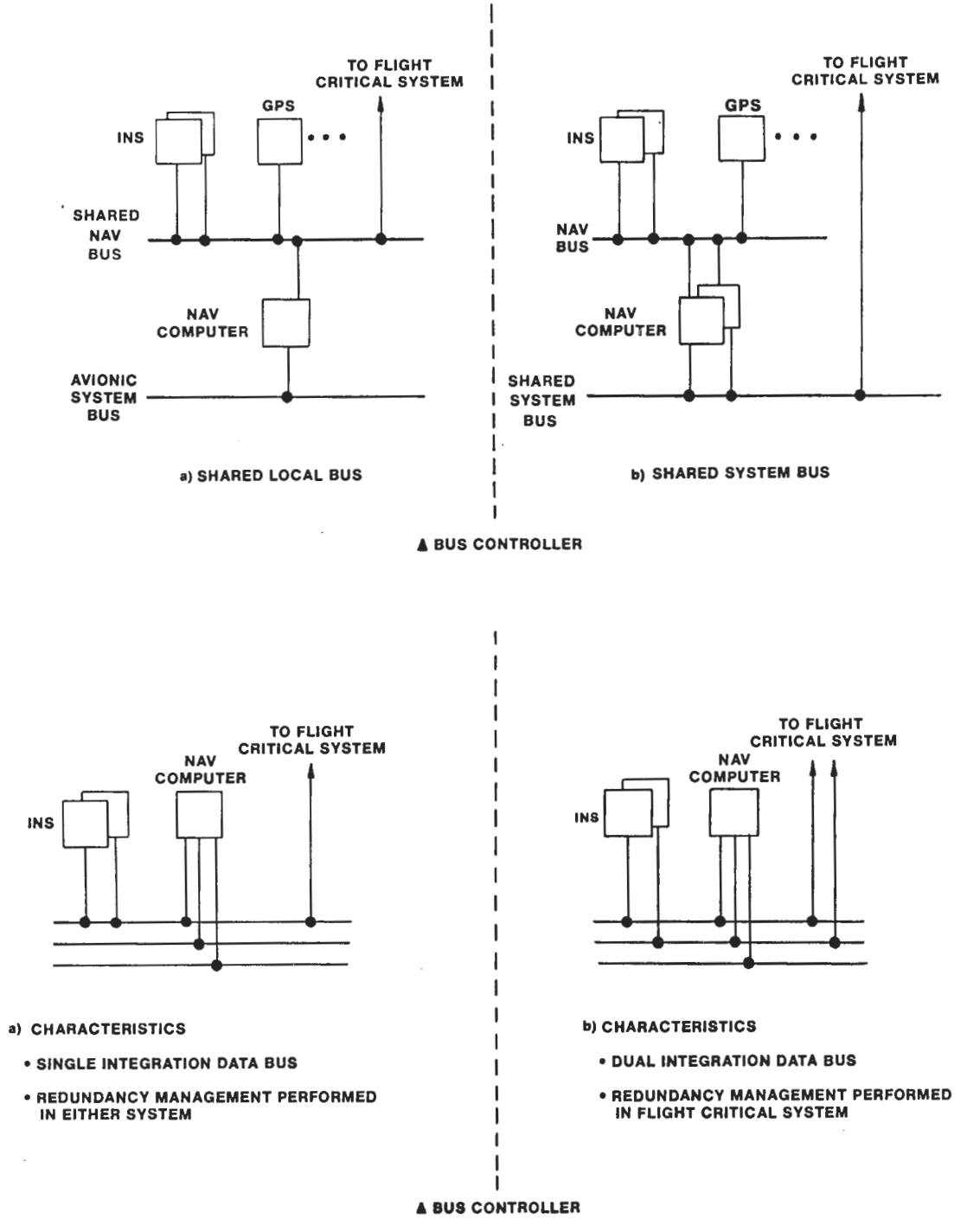



FIGURE 2.4-3. INTEGRATION CONCEPTS
(from McSharry 1983)

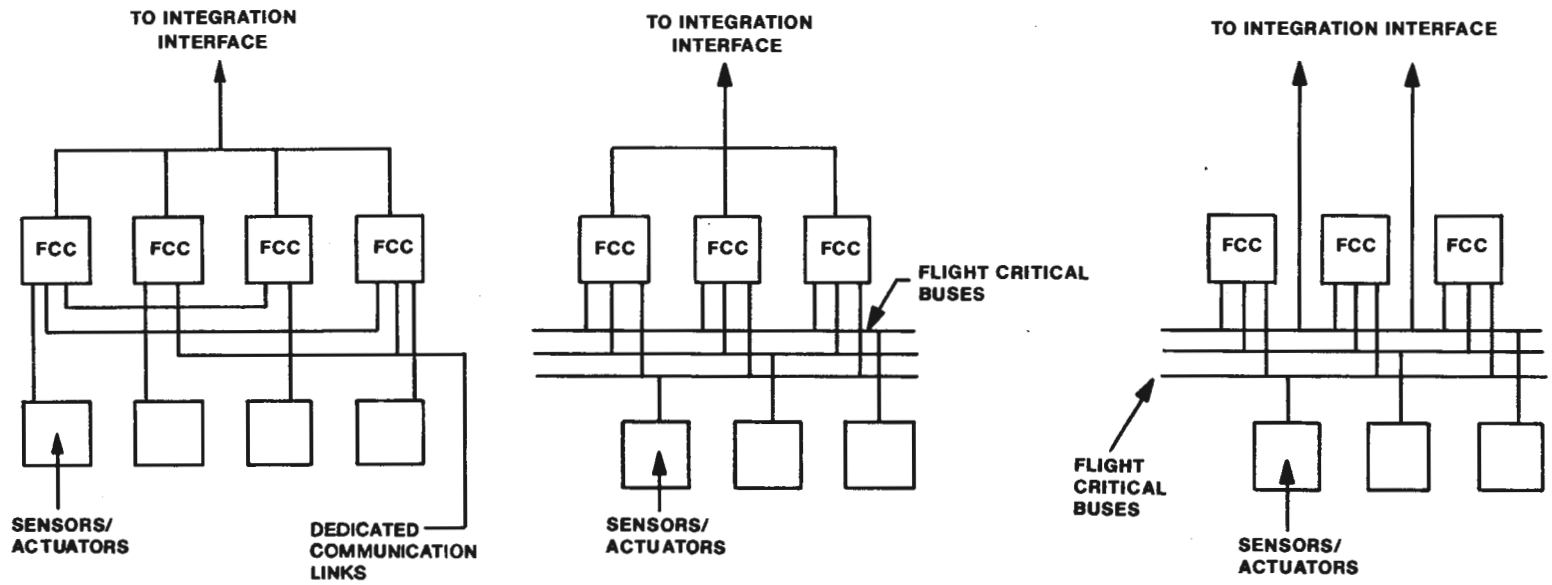



FIGURE 2.4-4. INTEGRATION CONCEPTS FOR FLIGHT CRITICAL SUBSYSTEMS

placed (through multiparty techniques) across the information transfer bus hierarchy. This approach reduces the reliability constraints on each of the various system functions. It can, however, introduce new potential failure points into the flight control and mission-dependent computation functions due to the increased complexity.

## 2.5. The Choice of an Architecture or Topology

The selection of a system architecture (including bus structure, topology, and integration concept) is based on the design requirements and the preference of the system designer, integrator, and implementor. In the concept design phase, a number of candidate architectural concepts, bus architectures, and topologies are postulated. Each is capable of satisfying system requirements within the constraints of the required performance, reliability, and safety criteria levels established by the relevant guidance documents (FARs, Advisory Circulars, and other accepted air worthiness practices). The selection of the final design for the information transfer system will ultimately be a function of selected system components, required interfaces, time-critical events and data, and various measure-of-merit attributes that drive the integrator's decisions.

The measures-of-merit and desirable bus attributes (as presented in tables 2.5-1 (McSharry, 1983) and 2.5-2 (Rich et al., 1983) are guidelines to be used by the system designer and integrator in assessing the integrity of the proposed information transfer system. Architecture, structure, protocol, and integration complexity must be considered to fully understand and evaluate the ultimate performance, reliability, safety, and airworthiness of the final design.

## 2.6. Protocol/Topology Relationships

The technical analyses leading to the selection of a topology and protocol for a given application require that topologies, protocols, media components, and configurations all be analyzed in terms of the constraints imposed upon the detailed system design, and the existing state of technology in each of these areas. Although a number of topologies and protocols currently exist, those most applicable to commercial transport are presented in table 2.6-1. As can be seen from this table, the choices for protocols are dependent upon the topologies, and the extent to which high-speed, distributed, fault-tolerant computation and passage of data are maximized in the system architecture. The topologies themselves are not likely to change in the near future, but the implementation, in terms of redundancy and hierarchical structure, will be designed to provide the critical data at the correct iteration rate with the least time delay at the lowest cost. As more and more aircraft systems are implemented with these concepts, the cost of integration (primarily Bus Interface Units (BIU)) will decrease and advances in electronics technology (at the silicon chip level) will allow additional variations which may not be feasible at the present time.

TABLE 2.5-1. MEASURES OF MERIT
(adapted from McSharry, 1983)

| Measures-of-Merit | Quantitative Measure |
|---|---|
| Flight Safety - ability to maintain control of aircraft | Probability of loss of control |
| Mission/Flight Phase Reliability - ability to satisfy mission requirements | Probability of loss of mission/flight phase capability (i.e., Autoland, etc.) |
| Maintainability - time required to repair and frequency of repair | MTTR, MDT, MTBCF |
| Availability - ability to initiate a mission or flight phase activity/function including full-time, full-authority system (i.e., FADEC, PAS, Envelope Limiting) | Operational availability and inherent availability are usually presented as a percentage (i.e., 99%) or are given as an exposure time (i.e., two minutes). |
| Flexibility - ability to accommodate changes | Reconfiguration cost |
| Reconfigurability - ability to compute or perform mission or flight phase function in presence of failures | Dynamic reconfiguration time or redundancy default (fail safe/fail safe) |
| Computational Capability - throughput of system computers | Total instructions executed per second |
| Data Transfer Capability - ability to send messages in a timely manner and in presence of failures | Maximum data latency, % peak bus loading |
| Pilot Interface - ability to provide cognitive information to pilot | Qualitative |
| Cost - initial procurement and life cycle cost | Dollars |

TABLE 2.5-2. ATTRIBUTES
(adapted from Rich et al., 1983)

| Attribute | Quantitative Measure |
|---|---|
| Fault Tolerance | Probability of error occurring; Reconfiguration time; Probability of propagation |
| Efficiency | Available bandwidth |
| Simplicity | Presence/absence of complexity, and complexity metric rating |
| Data Integrity | Probability of correct data transfer, and number of retries |
| Synchronous/Asynchronous | Time to respond to emergency messages/interruption |
| Adaptable to new technology; Technology Insertion | Qualitative |
| Similarity to existing bus architectures/ structures/protocols | Qualitative |
| Deterministic | Qualitative |

TABLE 2.6-1. ALTERNATIVE TOPOLOGIES AND PROTOCOLS

| TOPOLOGIES | PROTOCOLS | | | | |
|---|---|---|---|---|---|
| | COMMAND RESPONSE (ARINC 429-5) | CSMA/CD (DATAC ARINC 629) | TOKEN PASSING (SAE) | HDLC (ASCB) | TDM/CR (MIL-STD-1553B |
| Point-to-Point | X | | | | |
| Linear Bus | X | X | X | X | X |
| Star | | | | | X |
| Ring | | | X | | |

# 3. CURRENT AVIATION BUSES

## 3.1. Introduction

The first two sections of this tutorial provided a brief overview of the history and likely evolution of aviation digital data buses. This section reviews the specifications for five current and near-future avionics data bus concepts. The three data buses currently in use are ARINC 429, MIL-STD-1553B, and GAMA ASCB.

The two near-future bus concepts are as follows:

- The proposed ARINC 629 (DATAC), which is expected to be certified in the 1990s on the 7J7 and other large commercial transport aircraft.

- The two proposed SAE (Avionics System Division) High-Speed Data Bus Architectures, which are expected to be the standard for civil and, perhaps, military aviation in the year 2000 and beyond.

These five bus concepts are the only buses expected to be certified by the FAA within the next 20 years. Each bus standard reflects an ongoing attempt to develop data buses that will enable greater integration and flexibility. The proliferation of diverse data buses is due to the different needs of the developers. For example, the ASCB bus was developed by GAMA to take advantage of the large number of avionics components being developed by the general aviation avionics industry. Their bus concept was based upon the availability of low-cost commercially-available components that could be easily integrated into the low cost equipment.

Each of the buses will be discussed in terms of the following:

- Major architectural and topological characteristics.

- Protocol used.

- Design specifications.

- Major documents describing the bus and its specifications which can be consulted for additional information.

A separate appendix (appendix A) has been included to explain and define the various modulations, signaling methods, and data code waveforms that are commonly used for these digital data buses.

## 3.2. ARINC 429

### 3.2.1. History

The ARINC 429 bus standard was the first standard developed for the transfer of digital data between avionics system elements in civil aircraft. Created during the mid to late 1970s by the air transport industry (the airlines, avionics manufacturers, airframe manufacturers, and the FAA) to meet the emerging need for distributed intelligence and communication between avionics subsystems, the ARINC bus quickly replaced analog point-to-point wiring systems which were prevalent at that time. Currently, the ARINC bus standard represents the type of bus used in most civil aircraft manufactured during the past decade. A schematic representing the basic organization of the ARINC bus is shown in figure 3.2-1.

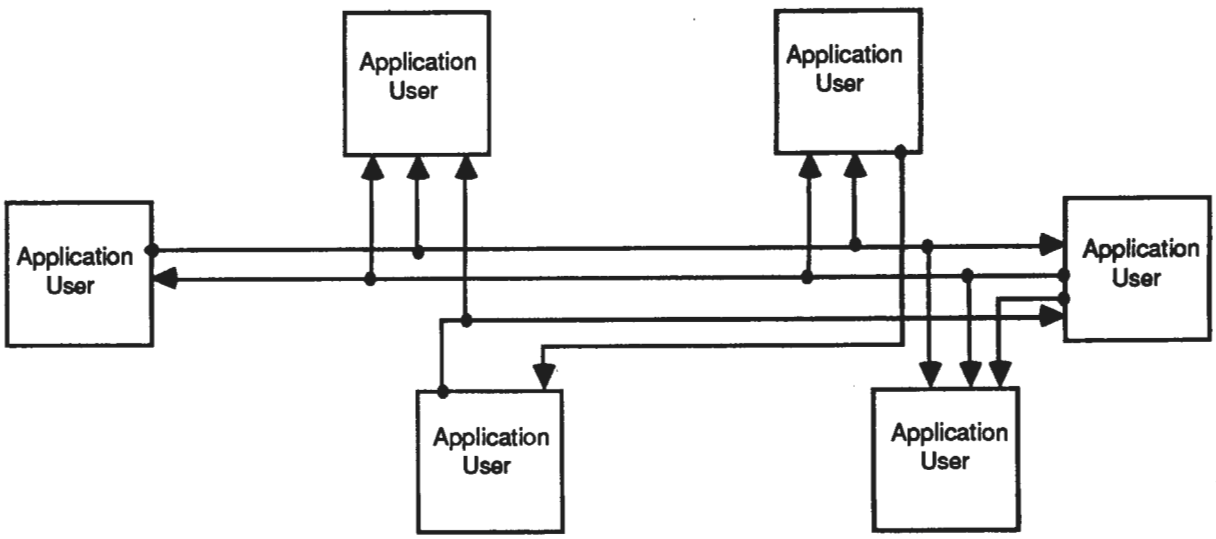



FIGURE 3.2-1. GENERIC ARINC BUS LAYOUT

### 3.2.2. ARINC 429 General Characteristics

General characteristics of the ARINC bus include the following:

- Relatively low-speed, high-reliability data transfer.
- Two data rate options (Note: High and low rate messages cannot be combined on the same bus):

  High-speed (100K bits per second (bps)) - Low-speed (12K to 14.5K bps).

- Unidirectional, asynchronous, serial data transfer.

- Data transfer by means of a single twisted, shielded pair of wires.

- Simplex data transfer only:

  Each component that needs to talk to other components has a wire going to the other components from its transmitter.

  Each component that needs to receive data from other components has a wire from that component to its receiver.

- Transmission by "broadcast" mode: receivers are not required to respond that data has been received (although they frequently do).

- Each transmitter able to send to up to 20 receivers.

- Data encoded in binary or binary-coded decimal.

- Data words composed of 32 bits, including label, word type, and a parity bit.

- Transmission accuracy determined by a single parity bit. Data reasonableness checks may also be performed.

### 3.2.3. ARINC 429 Protocol: Command Response

The ARINC 429 bus uses a Command Response (CR) protocol. When a transmitter is ready to send data to a receiver, it first sends a "request to send" word to the intended receiver by means of the bus that connects the transmitter to the receiver. If the receiver is ready, it will respond with a "clear to send" word sent by means of a separate bus that connects the receiver to the transmitter. The data then are sent immediately after transmission of a "data follows" initial word. A "final word", used for error control, signals completion of data transmission.

Once the data have been received, the receiver will process the data in search of errors, such as parity and file size. If no errors are found, the receiver may send a "data received OK" word to the transmitter. If an error is detected, the receiver requests the transmitter to retransmit the record containing the error.

Should the receiver not be ready to accept data, it sends back a modified "clear to send" that states its lack of readiness and may also specify the maximum number of records it is capable of receiving. After 200 millisecond (ms), a "request to send" is again transmitted. If, however, no response is received from the receiver, the transmitter will repeat a "request to send" after 50 ms. If the receiver fails to respond after four attempts, an alert is raised in the system containing the transmitter.

A single message, consisting of an initial word type, such as "data follows", and any data, is defined as a "record". Up to 126 data words may be sent in a single record. A single file may contain up to 127 records. Each record should contain one of the eight initial word types:

- Request to send.
- Clear to send.
- Data follows.
- Data received OK.
- Data received not OK.
- Synchronization lost.
- Header information (used to exchange file size information without actual transmission of file data).
- Poll ("I have no information for you, do you have anything for me?")

Synchronization occurs by means of a minimum of four bit-times which occur prior to each new word. Specific descriptions of these types of words, and the kind of information assigned to each bit, can be found in the ARINC 429 specification ("Mark 33 DITS").

3.2.4. ARINC 429 Hardware Characteristics

Table 3.2-1 presents a summary of hardware characteristics of the ARINC 429 data bus.

3.2.5. Major Documents

The characteristics of the ARINC 429 bus are described in ARINC Specification 429-5 ("Mark 33 DITS", April 1981).

3.3. MIL-STD-1553B and MIL-STD-1773

3.3.1. History

MIL-STD-1553B was originally developed by the military to support the integration of weapon systems. It has subsequently been used in commercial aviation for a number of systems including the Coast Guard Search and Rescue Helicopter HH-65A. Developed at the same time as ARINC 429, it represents the current generation of data buses developed for military and civilian aircraft. A schematic of the generic bus is shown in figure 3.3-1.

The proposed DOD-STANDARD 1773 DATA BUS (Fiber Optic Mechanization of an Aircraft Internal Time Division Command/Response Multiplex Data Bus) was developed for the following purposes:

- It seeks to preserve the multiplex bus techniques which have been standardized in MIL-STD-1553B, and
- It provides guidelines for the application of fiber optic transmission techniques to the MIL-STD-1553B interconnect.

TABLE 3.2-1. ARINC 429 HARDWARE CHARACTERISTICS

| | |
|---|---|
| Transmission Medium | Twisted Shielded Pair |
| Characteristic Impedance | 75 ± 5 Ohms |
| Main Bus Length | Not Specified |
| Media Connections | Direct Coupled |
| Modulation | Baseband (TDM) |
| Signaling Method | RZ Bipolar |
| Transmission Direction | Uni-Directional |
| Transmission Method | Asynchronous Broadcast |
| Transmission Order | LSB First |
| Data Rate | 12-14.5 KHz or 100 KHz |
| Data Code | RZ Bipolar |
| Bit Error Rate | Not Specified |
| Word Error Rate | Not Specified |
| Topology | Serial Bus |
| Number of Terminals/Addresses | Less Than 20 |
| Logical Addresses | Not Specified |
| Media Access | Point-to-Point |
| Data Link Control Protocol | Not Applicable |
| Error Detection | Odd Parity |
| Synchronization | Word |
| Word Size | 32 Bits |
| Data Bits/Word | 19 Bits |
| Words/Message (Min.-Max.) | 1 |
| Word Types | Not Specified |
| Intermessage Gap Time | 4 Bit Times |
| Bus Frame Length | Not Specified |
| Bus Control Transfer Time | Not Applicable |
| Terminal Transmit Interface | Not Specified |
| Terminal Receive Interface | Less Than 20 |

### 3.3.2. MIL-STD-1553B General Characteristics

General characteristics of the MIL-STD-1553B bus include the following:

- Relatively high-speed (1 MHz bit rate), high-reliability data transfer.
- Single (redundant) pathway to connect each bus element (component) to the bus controller terminal. All equipment connected to the bus has access to information sent by means of the bus.

- Data transfer by means of a single twisted, shielded pair of wires.
- As many as 31 terminals per data bus.
- Bidirectional (half-duplex), asynchronous, serial data transfer.
- Data flow controlled by a bus controller. The controller may be a separate component or share a box with other functions.
- Redundant controllers optional (although only one is active at a time). The second controller is activated if the first controller malfunctions.
- Overhead penalty for multiple terminal and controller checking, testing, and switching functions.
- Command-response protocol.
- Data word size 20 bits. Blocks of up to 32 data words are allowed.
- Transmission accuracy determined by a single parity bit.
- Synchronization by a three-bit synchronizing field.

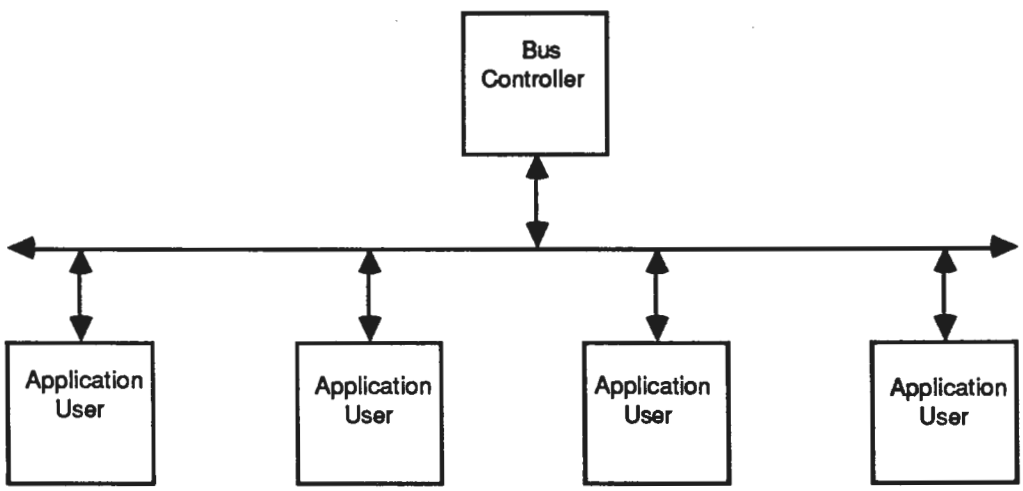



FIGURE 3.3-1. GENERIC MIL-STD-1553B BUS LAYOUT

### 3.3.3. MIL-STD-1553B Protocol: Time Division Multiplex, Command/Response

As the name suggests, data flow control for all transmissions on this bus uses a combination of Time Division Multiplexing (TDM) and command/response modes of operation. TDM refers to "the transmission of information from several signal sources through one communication system with different signal samples staggered in time to form a composite pulse train." (MIL-STD-1553 Designer's Guide, 1983, p. II-2.) With TDM, each component is assigned a pre-allocated timeslot. Consequently, an individual component knows when to transmit its data, how long it can transmit, and when to give up control of the bus to the next scheduled transmitter.

Pre-allocation of transmission slots requires that all message sequences be predefined to enable fixed schedules of data transfer. This is based on the definition of minor and major cycles. Minor and major cycles reflect the fastest and slowest iteration rates, respectively. The slowest iteration rate, which is the data required to be sampled least often, defines the time period for the major cycle. Over the course of a major cycle, all periodic bus events occur at least once and all periodic computations occur at least once. A frame is one major cycle. Within a frame, one or more minor cycles may occur as a function of the required frequency of the most rapidly transmitted periodic data.

The Multiplex Applications Handbook (1982, p. 5-4) offers the following example: "If the major frame is one second long and there are 64 minor cycles, then each minor cycle is 1/64 second or 15.625 ms long. Each periodic message would occur at least once each major frame, up to a maximum of 64 times. If a transaction needed to occur eight times per second, it must occur during one of the first eight minor cycles (64/8 = 8) and every eight minor cycles thereafter. The minor cycle in which the first message occurs is known as the 'phase', while the repetition rate is its 'period'."

Pre-allocation of timeslots means that responsibility for controlling the bus is distributed among all participating components. Since transfer of bus control can be almost instantaneous, little bus transmission time is lost. Consequently, bus utilization can, at least in theory, approach 100 percent (Rich et al., 1983). Practically speaking, this is not the case. Timeslotting is highly fault tolerant in that failure of the component currently controlling the bus does not cause a permanent disruption. Instead, the other components continue transmitting their data at their assigned slots. The only consequence is that the slot belonging to the failed component goes unfilled.

The timeslot protocol typically does not support 100 percent utilization because of problems with fault tolerance with respect to when individual message errors occur. Complete pre-allocation of slots does not allow for retransmission of messages that were not successfully transmitted initially - no slots are available for retries.

One approach for introducing message-level fault tolerance is to use timeslots that are larger than required for single-try message transmission. This leaves room for retransmission of a part of the message, if required. Clearly, this

reduces the amount of message traffic supported by the bus. Providing sufficient slot time for a single retransmission of a message has the major consequence of reducing overall bus capacity by 50 percent.

Note that the message retry capability represents an "unscheduled" data transmission event. Other types of unscheduled transmission can take place during those intervals when data are not already scheduled to be placed on the bus. The bus controller can command a receiver to transmit during these intervals as required. This is an example of how the command/response mode can be superimposed on the basic TDM mode. Using the command/response mode in conjunction with TDM has two advantages: intervals when no data are scheduled to be transmitted can be filled so as to make greater use of the bus's capacity, and the controller can respond to events that do not occur regularly.

3.3.4. MIL-STD-1553B Hardware Characteristics

Table 3.3-1 presents the characteristics of the MIL-STD-1553B bus.

3.3.5. MIL-STD-1773 Hardware Characteristics

Table 3.3-2 presents the characteristics of the MIL-STD-1773 bus, which is the fiber optic counterpart of MIL-STD-1553B. MIL-STD-1773 allows for five possible coupled architectures: reflective star, transmissive star, bidirectional T, unidirectional T, and bidirectional hybrid. The star coupler may be passive or active and can be embedded within the Line Replaceable Unit (LRU) or external to the LRU. Dual speed operation of the MIL-STD-1773 data bus is being examined by a number of vendors to make better use of the bandwidth possible in the bus.

Due to the need for compatibility with MIL-STD-1553B, the MIL-STD-1773 must operate in the time domain and use Manchester II encoding. Matching the Manchester II encoding scheme of MIL-STD-1553B with a fiber optic system results in the average optical power level during each sync code or information bit equaling one-half of the on-power level. Bilevel optical Manchester modulation does have an average optical power of zero when a message is not being transmitted. Consequently, there is a low-frequency component, which has a fundamental frequency equal to the message rate, often 10 Hz or less. Fiber optic receivers are usually ac-coupled to compensate for the photodetector's electrical signal levels, which are not very large in comparison with the magnitudes of amplified drift and offset voltages. Because of this, special signal processing is needed to offset the effect of the low-frequency component.

Several techniques have been developed for dealing with this low frequency component, but these are susceptible to noise from within the system. The result is that the transmitter sections must be more efficiently decoupled from sources of noise in their equipment and must be quieter when not transmitting. In addition, because of the low input power levels to the fiber-optic receivers, the front-end electrical signal levels are much lower in MIL-STD-1773 receivers than in those for MIL-STD-1553B. To obtain satisfactory performance with the greatly reduced signal level, careful shielding is required, as well as decoupling of electrical interference on subsystem lines entering the receiver.

TABLE 3.3-1. MIL-STD-1553B HARDWARE CHARACTERISTICS

| | |
|---|---|
| Transmission Medium | Twisted Shielded Pair |
| Characteristic Impedance | 70 to 85 Ohms @ 1 MHz |
| Main Bus Length | Not Specified |
| Media Connections | Transformer Coupled |
| Modulation | Baseband (TDM) |
| Signaling Method | Biphase Level |
| Transmission Direction | Bidirectional Half-Duplex |
| Transmission Method | Asynchronous |
| Transmission Order | MSB First |
| Data Rate | 1 Megabit/Second |
| Data Code | Manchester II Biphase Level |
| Bit Error Rate | One Per $10^{12}$ Bits |
| Word Error Rate | One Per $10^{7}$ Words |
| Topology | Single Serial Bus (Redundant OK) |
| Number of Terminals/Addresses | 31 Addresses/30 Subaddresses Each |
| Logical Addresses | Not Specified |
| Media Access | Command/Response |
| Data Link Control Protocol | Not Applicable |
| Error Detection | Odd Parity |
| Synchronization | Word |
| Word Size | 20 Bits |
| Data Bits/Word | 16 Bits |
| Words/Message (Min.-Max.) | 1-32 |
| Word Types | Command, Status, Data |
| Intermessage Gap Time | 4 Microseconds |
| Bus Frame Length | Not Specified |
| Bus Control Transfer Time | Not Specified |
| Terminal Transmit Interface | Not Specified |
| Terminal Receive Interface | Not Specified |

Since optical signals cannot assume negative values, the receiver outputs, which are complementary and thus never low at the same time, cannot be used to identify the no-message state in a MIL-STD-1773 system. As a result, the no-message state and the off state of a two-level Manchester II biphase bit cannot be distinguished. In MIL-STD-1773, it is considered good practice to design fiber-optic receivers with three output states, even though the receivers have only two input states. This is done for compatibility with the outputs of wire-based receivers.

TABLE 3.3-2. MIL-STD-1773 HARDWARE CHARACTERISTICS

| | |
|---|---|
| Transmission Medium | Fiber Optic |
| Characteristic Impedance | Not Specified |
| Main Bus Length | Not Specified |
| Media Connections | Not Specified |
| Modulation | Baseband (TDM) |
| Signaling Method | Biphase Level, 2-State |
| Transmission Direction | Bidirectional Half-Duplex |
| Transmission Method | Asynchronous |
| Transmission Order | MSB First |
| Data Rate | Multiple Speed |
| Data Code | Manchester II Biphase Level |
| Bit Error Rate | One Per $10^{12}$ Bits |
| Word Error Rate | One Per $10^{7}$ Words |
| Topology | Single Serial Bus (Redundant OK) |
| Number of Terminals/Addresses | 31 Addresses/30 Subaddresses Each |
| Logical Addresses | Not Specified |
| Media Access | Command/Response |
| Data Link Control Protocol | Not Applicable |
| Error Detection | Odd Parity |
| Synchronization | Word |
| Word Size | 20 Bits |
| Data Bits/Word | 16 Bits |
| Words/Message (Min.-Max.) | 1-32 |
| Word Types | Command, Status, Data |
| Intermessage Gap Time | Not Specified |
| Bus Frame Length | Not Specified |
| Bus Control Transfer Time | Not Specified |
| Terminal Transmit Interface | Not Specified |
| Terminal Receive Interface | Not Specified |

### 3.3.6. Major Documents

The characteristics of the MIL-STD-1553B bus are described in The MIL-STD-1553 Multiplex Applications Handbook, dated February 1982.

The characteristics of the proposed DOD-STANDARD 1773 DATA BUS are described in a tutorial entitled FIBER OPTIC DATA BUSES, National Aerospace & Electronics Conference (NAECON), 19 May 1986. The specific subject is covered in the section entitled MIL-STD-1773 (C. R. Husbands, The Mitre Corporation).

## 3.4. The GAMA ASCB

### 3.4.1. History

The ASCB bus was developed, with the support of GAMA, to provide digital communications in business jets and commuter turboprop airplanes using low-cost commercially available components for the bus interface and an existing protocol (High-level Data Link Control (HDLC)). A schematic of the generic bus is shown in figure 3.4-1.

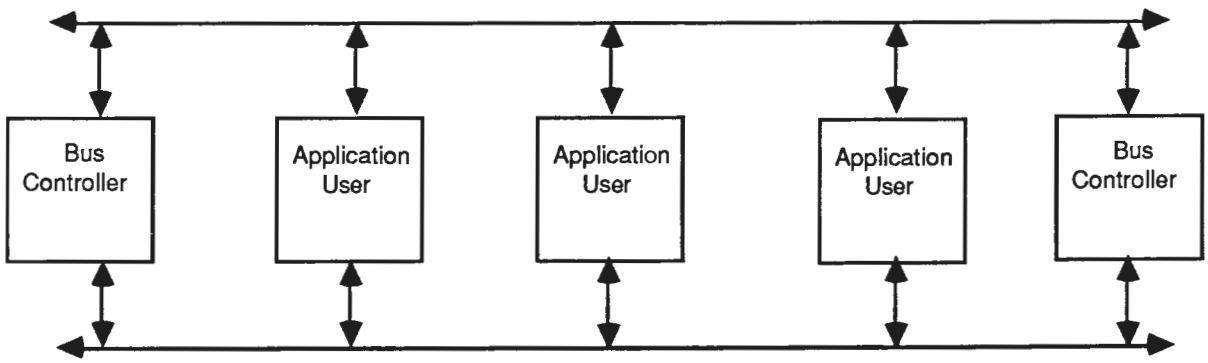



FIGURE 3.4-1. GENERIC ASCB BUS LAYOUT

### 3.4.2. GAMA ASCB General Characteristics

General characteristics of the ASCB bus include the following:

- Medium speed (0.67 MHz bit rate).
- Serial, bidirectional (half-duplex), redundant bus architecture.
- Redundant bus controllers (only one controller is active at a given time).
- Bus controller transmission on both buses simultaneously.
- Application user transmission on one bus; all application users listening to both buses.
- Support for up to 48 receivers.
- Broadcast format.
- Industry standard HDLC protocol.

- Word size of 16 bits, with up to 256 words per message.
- Low-cost interface components (transceivers, HDLC protocol chips, etc.).
- Multiple error-check capabilities, including data counters, CRC, and checksum.

### 3.4.3. GAMA ASCB Protocol: HDLC

With this protocol, transmission order is maintained by the active bus controller. An eight-frame timing cycle is used, with each frame lasting 25 ms. The bus controller requests the various users to transmit their data at appropriate intervals within the frame cycle, as required by the update rate needed for desired system performance. To prevent a user transmission from exceeding the frame time, an interlock is used.

The generic frame sequence is as follows. The active bus controller issues a start of frame word (the synchronization pattern) that announces the start of a new frame. This is followed by a frame control word, which contains information about the specific frame, including the frame number and controller identification. Any requests for user transmission complete the frame period. In this way, users can listen for information relevant to them and know when to expect that information to arrive. Similarly, each user knows when to transmit information: each user has a unique request address and response address. It listens for its address and transmits at the time stated in the frame. Users listen for their addresses on both buses. In response to a request for transmission, the user replies with its response address and the data it is required to transmit.

The transmitting user must have its data ready to transmit prior to its assigned frame. If it is not ready, the transmitter will not have enough data to fill out the frame. When this happens, the entire frame is aborted.

Synchronization is maintained by the unique eight-bit pattern that serves as both a "start-of-frame" word and an "end-of-frame" word. Each time this pattern occurs, users automatically resynchronize. This pattern, "01111110", is called a "flag". Consequently, no data sequence is allowed to contain six "1"s in a row. Whenever five successive "1"s are sent, the transmitter automatically replaces the final "1" with a false "0". The receiver then automatically replaces the false "0" with a "1". Adding and replacing these "1"s and "0"s has no effect other than to reduce the data transmission rate slightly.

### 3.4.4. GAMA ASCB Hardware Characteristics

Table 3.4-1 presents the characteristics of the GAMA ASCB.

### 3.4.5. Major Documents

The characteristics of the ASCB are described in the GAMA Specification dated February 1986.

TABLE 3.4-1. GAMA ASCB HARDWARE CHARACTERISTICS

| | |
|---|---|
| Transmission Medium | Twisted Shielded Pair |
| Characteristic Impedance | 125 Ohms |
| Main Bus Length | 125 Feet |
| Media Connections | Transformer Coupled |
| Modulation | Baseband (TDM) |
| Signaling Method | Biphase Level |
| Transmission Direction | Bidirectional Half-Duplex |
| Transmission Method | Asynchronous |
| Transmission Order | LSB First |
| Data Rate | .67 MHz $\pm$ 0.05% |
| Data Code | Manchester II Biphase Level |
| Bit Error Rate | One Per $10^8$ Bits |
| Word Error Rate | Not Specified |
| Topology | Dual Serial Bus |
| Number of Terminals/Addresses | 48 |
| Logical Addresses | Not Specified |
| Media Access | Not Specified |
| Data Link Control Protocol | HDLC (BOP) |
| Error Detection | CRC |
| Synchronization | Frame |
| Word Size | 2 Bytes |
| Data Bits/Word | 16 Bits |
| Words/Message (Min.-Max.) | 1-256 |
| Word Types | Not Specified |
| Intermessage Gap Time | 8 Bit Times (Minimum) |
| Bus Frame Length | 25 ms |
| Bus Control Transfer Time | 50 ms |
| Terminal Transmit Interface | One Bus Only |
| Terminal Receive Interface | Both Buses |

## 3.5. DATAC (ARINC Specification 629) Bus

### 3.5.1. History

The DATAC bus is being developed by Boeing in response to the ongoing changes in capabilities and complexities of avionics systems being implemented in the next generation commercial transport airplane. The DATAC bus concept is being further developed by Boeing and ARINC as ARINC Specification 629 (Multi-transmitter Data Bus) that is available (as of 1987) in draft form. The bus has already been implemented in the 7J7 for signaling, avionics, and airframe systems, and fly-by-wire. A guiding objective of the bus standard is to support ease of communication between subsystems yet avoid the problems associated with use of a central bus controller. Problems to be avoided include the potential

for single point failure and the need to change the bus controller when any aspect of the bus architecture is modified. A schematic of the generic bus is shown in figure 3.5-1.

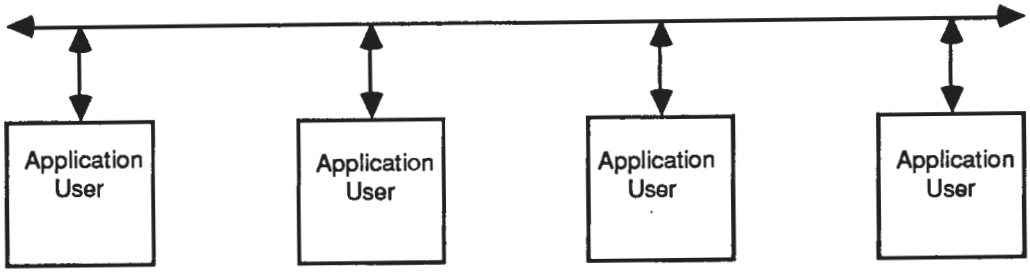



FIGURE 3.5-1. GENERIC DATAC BUS LAYOUT

### 3.5.2. DATAC General Characteristics

General characteristics of the DATAC bus include the following:

- High-speed data transfer (2 MHz bit rate).
- Shielded or unshielded twisted pair or fiber optic mode.
- Connection of as many as 120 terminals.
- Linear topology using carrier-sense multiple access/collision avoidance protocol.
- Multitransmitter implementation.
- Special current-mode coupling devices.
- Direct connection to bus through current-mode coupling devices.
- Self-monitored terminals to reduce the risk of bus failure.
- Modularity: can easily add and delete terminals without affecting overall bus functioning.
- Orderly transfer of data ensured by the protocol, rather than a bus controller.
- Carrier-Sense Multiple Access/Collision Avoidance Protocol.
- Compatibility with ARINC 429.

### 3.5.3. DATAC Protocol: Collision Detection

The DATAC protocol has no central bus controller. Instead, individual transmitters are allowed to transmit whenever they wish. Clearly, there is a strong likelihood that two or more transmitters will attempt to use the bus at the same time. Such collisions result in the transmission of distorted data from both transmitters. Consequently, both messages will have to be retransmitted. Even if a second attempt leads to successful transmission, rather severe inefficiencies in bus use will result since the time taken to send both messages could be many times longer than the lengths of the messages. Also, repeated collisions may result. As a result, use of the Collision Detection (CD) method, in its simplest form, tends to allow less than 20 percent utilization of bus capacity before bus stability becomes problematic (Rich et al., 1983). Higher utilizations radically increase the probability of collisions during a second attempt at message transmission. Once this occurs, the total traffic from the first collision, plus that from the second is thrust downstream in the overall message traffic, increasing the likelihood of additional collisions. In short, at some point the process begins cascading until all terminals in the network become involved and no successful transmissions can be performed.

Although the "pure" form of CD has some severe problems, modifications of this approach are more successful. One approach is to enable each transmitter to detect the presence of a message on the bus and, if the bus is "busy", to delay transmission of its message. This approach is known as Carrier Sense Multiple Access (CSMA). When used in conjunction with CD it is referred to as CSMA/CD and is the basis for the DATAC protocol.

With CSMA/CD the occurrence of interfering transmissions is restricted to that situation in which two terminals begin to transmit "so closely together in time" that neither has yet sensed the other's signal. This short time interval at the beginning of a message is referred to as the "collision window" and is simply due to the propagation delay of the network. The collision window is typically on the order of a microsecond in a wired network over short distances.

Requiring the following three events to occur before a transmitter is allowed to send a message again prevents collisions (Shaw, Herzog and Okubo, 1986, p. 223):

"(1) a frame time, common to all terminals on the bus, must have elapsed
(2) a sync gap, common for all terminals, must have existed on the bus
(3) a terminal gap, common for all terminals, must also have existed on the bus."

To protect against corrupted data, the receiver of the transmitting terminal monitors the transmission and checks that each label transmitted has been authorized, that each label contains the correct channel information, that the number of words allowed in that string has not been exceeded, and that the number of word strings in a message has not been exceeded. Any fault causes the transmitter to be inhibited for the remainder of that messages. It is allowed to try again on the next frame time. This continues until a certain number of successive tries are unsuccessful, at which time the terminal is permanently disabled.

### 3.5.4. DATAC Hardware Characteristics

Table 3.5-1 presents characteristics of the Boeing-developed DATAC (ARINC Specification 629) bus.

TABLE 3.5-1. BOEING DATAC BUS CHARACTERISTICS
(ARINC Specification 629)

| | |
|---|---|
| Transmission Medium | Twisted Pair (Non-Shielded, Insulated) |
| Characteristic Impedance | Not Specified |
| Main Bus Length | <100 Meters (m) |
| Media Connections | Transformer Coupled (Current Mode) |
| Modulation | Baseband (TDM) |
| Signaling Method | Biphase Level |
| Transmission Direction | Bidirectional Half-Duplex |
| Transmission Method | Asynchronous Broadcast |
| Transmission Order | LSB First |
| Data Rate | 2 MHz Bit Rate |
| Data Code | Manchester II Biphase Level |
| Bit Error Rate | One Per $10^{12}$ Bits |
| Word Error Rate | Not Specified |
| Topology | Single Serial Bus (Redundant OK) |
| Number of Terminals/Addresses | Up to 120 |
| Logical Addresses | Not Specified |
| Media Access | Contention |
| Data Link Control Protocol | CSMA/Collision Avoidance |
| Error Detection | Odd Parity |
| Synchronization | Frame |
| Word Size | 32 Bits |
| Data Bits/Word | 16 Bits |
| Words/Message (Min.-Max.) | 1-4096 (256 Words/String, 32 Str/Msg) |
| Word Types | Not Specified |
| Intermessage Gap Time | 14 Bit Time Minimum (Terminal Dependent) |
| Bus Frame Length | 50 ms |
| Bus Control Transfer Time | Not Specified |
| Terminal Transmit Interface | Not Specified |
| Terminal Receive Interface | Not Specified |

### 3.5.5. Major Documents

The characteristics of the DATAC bus are described in draft ARINC Specification 629 (Multi-Transmitter Data Bus) Part 1 - Protocol Description and Part 2 - Label and Word String Definitions, both dated September 1987.

## 3.6. SAE High-Speed Data Buses

### 3.6.1. History

The newest data buses are being developed by the SAE and industry through the SAE Avionics System Division (formerly SAE/AE-9B High-Speed Data Bus Subcommittee). Under this organization, two competing bus standards are being developed: (a) High-Speed Ring Bus Standard; and (b) Linear Token Passing Bus (LTPB) Standard These two bus concepts are currently in the development stage with draft specifications having been produced (through 1988) and prototype hardware having been developed and tested. Schematics of the two buses are shown in figures 3.6-1 and 3.6-2.

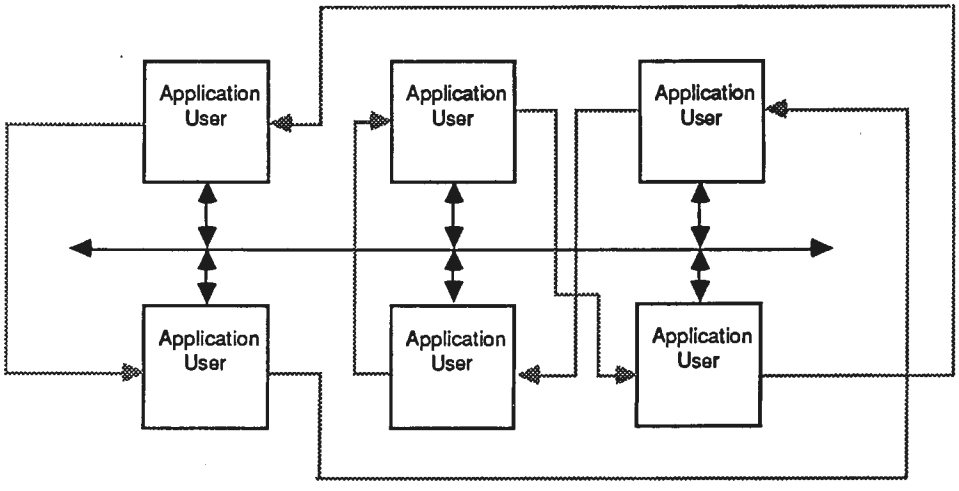



FIGURE 3.6-1. GENERIC SAE LINEAR BUS LAYOUT

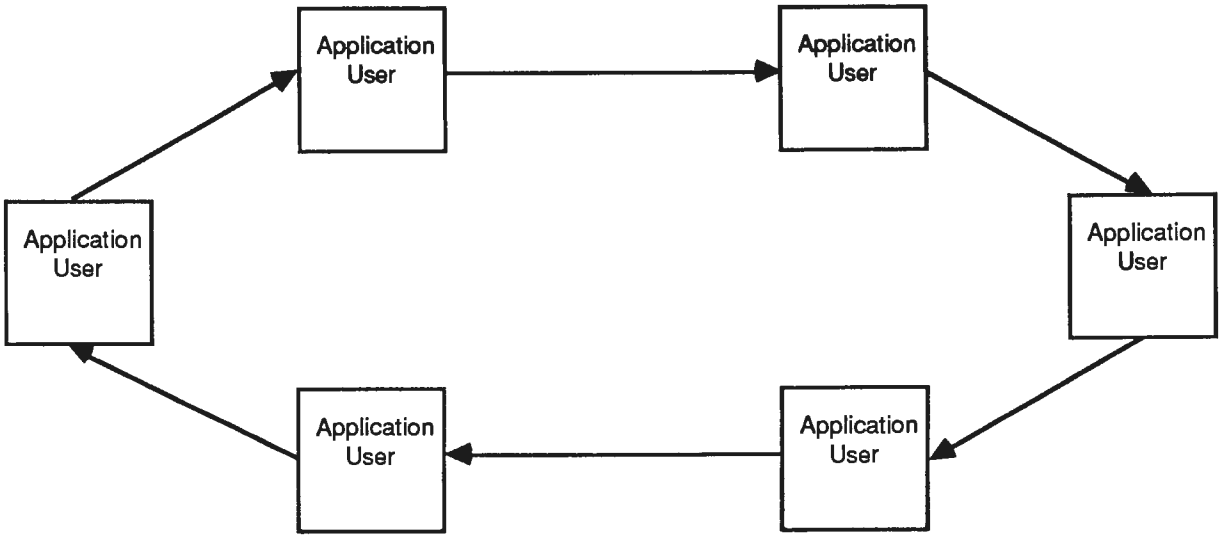



FIGURE 3.6-2. GENERIC SAE RING BUS LAYOUT

### 3.6.2. SAE General Characteristics

General characteristics of the SAE buses include the following:

- High-speed data transfer (20-40 MHz bit rate).
- Shielded or unshielded twisted pair, coaxial cable, or fiber optic media.
- Serial, bidirectional (half-duplex) data flow.
- Connection of as many as 128 physical terminals on each of four buses.
- Both broadcast and multicast modes.
- Modularity: can easily add and delete terminals without affecting overall bus functioning.
- Orderly transfer of data ensured by the protocol, rather than a bus controller.
- Word size of 16 bits (message lengths range from one word to 256 words for the linear bus; up to 4096 words for the ring bus).
- Logical addressing with $2^{16}$ addresses available.

### 3.6.3. SAE Protocol: Token Passing

Like the CD protocol, the token passing protocol also gives control of bus transmission to the transmitters themselves. Instead of a central bus controller, the currently-sending or last-sending station serves as the current controller. While acting controller, the transmitter provides the clock source for the entire bus, transmits its message, then passes a "token," a special message containing a data word which identifies the terminal which is to become the next controller. Each transmitter adds "1" to the token word before passing it on. The transmitter which receives the token becomes the new acting controller, repeating the same process.

When the last terminal controlling the bus completes operations, a message with a non-existent token number is created and transmitted. Since there is no terminal with that token number, a brief lapse occurs in the flow of data. The token-zero terminal is responsible for timing this data lapse and assuming control of the bus after the specified period of inactivity. Thus, the system automatically restarts itself with token zero whenever control lapses regardless of the number of terminals active in the system.

A terminal attempting to use an already active bus system simply monitors the bus for a few cycles to determine the highest token number currently in use. When this terminal identifies a token number to which no other terminal responds, it adopts this number and responds the next time that token number is issued. Since this terminal responds within the appropriate time-frame, the

token-zero terminal does not jump in. The token-zero terminal will restart the cycle after the new terminal has completed its operations and passed on a token to which no other terminal responds.

When a terminal has failed during a cycle and a token is offered to it, the token is ignored. No terminal takes control and bus activity ceases. The token-zero terminal detects the lack of activity and resumes control of the bus. All terminals with token numbers higher than the failed terminal are skipped. Each terminal in the system should be programmed to recognize and respond to this situation.

A terminal recognizes that something has failed in the network when more than two complete cycles have passed since the terminal received its last token offer. The terminal therefore decrements its token number by one. On the next cycle some terminal will recognize the previously ignored token and assume control. All higher numbered terminals have also decremented their token numbers and normal operation resumes. Until the ignored terminals decrement their tokens to rejoin the loop, abbreviated cycles will be run. Normal operations resume ignoring the failed terminal. If the unit recovers, it enters the loop as previously described.

This protocol works even if it is the token-zero terminal which fails, because when the token-one terminal discovers that it is being ignored, it decrements its token to zero and assumes the token-zero terminal functions. This implies that all terminals must have token-zero capabilities.

The token passing approach is highly tolerant of transmitter failure. But as was the case with the basic CD protocol, it is not as fault tolerant with respect to message errors. When a transmitter fails, not only is its data lost but also the data of all transmitters that follow the transmitter in the queue. Transmitter failure results in the transmission process starting over with the token zero terminal.

A potential problem arises because the protocol does not necessarily offer a good environment for managing unscheduled events since a terminal cannot transmit until the token is passed to it. The time between the generation of an unscheduled message and its transmission may be excessively long. Consequently, a true emergency message cannot be handled by the protocol without a scheme for assigning priorities, such as frequent polling of the source.

Another potential problem with this protocol is coping with message errors. For example, the time-out executed by the token zero terminal needs to be kept small in order to maintain high bus utilization. However, should a controller (due to extended error analysis,) pause too long before its next bus operation, the token zero terminal could interpret this as the end of a cycle and start the next cycle.

Consequently, resumption of operation by the pausing terminal could cause a collision with the traffic from the token zero terminal. When this happens, both colliding terminals believe they have failed. Should this occur, all bus activity stops until other bus transmitters recognize the problem and recover by adjusting their tokens accordingly. This may not fully resolve the problem

because when the two failed terminals attempt to rejoin the network, a collision is likely to occur once again. A collision can occur either between the two "failed" systems or between one of the "failed" systems and a correctly operating system.

Rich et al., 1983 suggest that concerns about such collisions could lead to extending the interval defined for token zero time-out. Rules for testing bus activity before initiating a new cycle need to be defined. Both these rules and the token zero time-out interval need to be considered when defining the time interval used by each terminal to decide when to decrement its token. This time interval should be longer than the maximum size and may be extended as a precaution.

The SAE High Speed Data Bus Committee, for both the linear and ring buses, has recognized these problems and has formulated solutions within the protocol.

### 3.6.4. SAE Hardware Characteristics

Table 3.6-1 lists the characteristics of the SAE High Speed Ring Bus (HSRB). This system is a physical token passing ring network which connects a set of stations in a closed loop topology with a serial transmission medium. While the semantics and protocol described are independent of the data transmission rate, the HSRB is intended to operate in the range ten million to one hundred million bits per second.

Ring access is by means of a token which passes from one station to the next to provide equitable access for any given priority of message. Data are transmitted in messages from one station to the next in a unidirectional manner, starting with the sending controlling station passing through the intermediate stations to the receiving addressed station. The receiving station copies the message for its bus interface unit (BIU) user and also retransmits it on the ring. The message passes through the remaining stations until it reaches the sending station that removes it from the ring. The controlling station must then transmit a token. A station is not permitted to transmit more than one message before issuing a token. Only a station that has claimed a valid token is authorized to transmit a message for its BIU user, all other stations are constrained to repeat each message as received on the ring with the exception of setting error reporting and priority reservation bits.

The ring bus offers superior throughput capability when compared with the linear bus due to short point-to-point media links between nodes. In the area of fault recovery and reliability, the ring is less attractive due to the need for failed node bypassing using either mechanical relays or fiber optic switches. Ring reconfiguration may take up to 25 ms when bypasses are activated. In addition, a limit must be placed on the number of consecutive nodes which may be bypassed, due to a lower power budget in the short point-to-point links and the relatively high losses inherent in the bypass devices (both wire and fiber optic).

Table 3.6-2 presents the characteristics of the SAE AS-2 (Formerly AE-9B) Linear Token Passing Bus (LTPB).

The AS-2 proposed LTPB protocol involves four simple states:

- Bus Initialization.
- Normal Token Passing.
- Station Insertion.
- Station Management.

The token is passed from the lowest physical address to the highest physical address and then back to the lowest.

TABLE 3.6-1. SAE RING BUS HARDWARE CHARACTERISTICS

| | |
|---|---|
| Transmission Medium | 50 MBPS Coax, 100 MBPS Fiber Optic |
| Characteristic Impedance | 75 ohm Triax |
| Main Bus Length | 300 m required, 1000 m desired (total station separation) |
| Media Connections | Optical or Transformer Coupling |
| Modulation | NRZ-I |
| Signaling Method | Biphase Level |
| Transmission Direction | Unidirectional |
| Transmission Method | Asynchronous Broadcast |
| Transmission Order | LSB First |
| Data Rate | 10-1000 MBPS |
| Data Code | Unspecified |
| Bit Error Rate | One Per $10^{12}$ Bits |
| Word Error Rate | Not Specified |
| Topology | Ring - 2 to 128 Stations |
| Number of Terminals/Addresses | 128 Physical/512 Subaddresses Each |
| Logical Addresses | $2^{15}$ - Broadcast and Multicast |
| Media Access | Token Pass |
| Data Link Control Protocol | Token or Message Frame |
| Error Detection | CCITT-CRC-16 |
| Synchronization | Frame |
| Word Size | 16 Bits |
| Data Bits/Word | 16 Bits |
| Words/Message (Min.-Max.) | 1-4096 |
| Word Types | 4B5B Block Encoding |
| Intermessage Gap Time | Not Specified |
| Bus Frame Length | 80K Bits |
| Bus Control Transfer Time | 10 Million Data Bits/Second |
| Terminal Transmit Interface | 4 Buses |
| Terminal Receive Interface | 4 Buses |

TABLE 3.6-2. SAE LINEAR BUS HARDWARE CHARACTERISTICS

| | |
|---|---|
| Transmission Medium | Fiber Optic or Electrical Wire |
| Characteristic Impedance | 50 ohms (electrical wire systems) |
| Main Bus Length | 300 m required, 1000 m desired |
| Media Connections | Optical or Transformer Coupling |
| Modulation | NRZ |
| Signaling Method | Biphase Level |
| Transmission Direction | Bidirectional Half-Duplex |
| Transmission Method | Asynchronous Broadcast or Multicast |
| Transmission Order | LSB First |
| Data Rate | 25, 50, or 100 Mbps (Optical Bus)<br>50 Mbps (Wire Bus) |
| Data Code | Manchester Biphase II |
| Bit Error Rate | One Per $10^{12}$ Bits |
| Word Error Rate | < 1 Every 4 Hours at BIR |
| Topology | 1 to 4 Serial Linear Buses |
| Number of Terminals/Addresses | 128 Physical/256 Subaddresses Each |
| Logical Addresses | $2^{15}$ |
| Media Access | Token Pass |
| Data Link Control Protocol | Token or Message Frame |
| Error Detection | CCITT-CRC-16 |
| Synchronization | Frame |
| Word Size | 16 Bits |
| Data Bits/Word | 16 Bits |
| Words/Message (Min.-Max.) | 1-256 Required, 4K Desired |
| Word Types | Not Specified |
| Intermessage Gap Time | 10 Bit Times |
| Bus Frame Length | Not Specified |
| Bus Control Transfer Time | Not Specified |
| Terminal Transmit Interface | 4 Buses |
| Terminal Receive Interface | 4 Buses |

The worst case delay in the AS-2 LTPB is directly dependent on the maximum allowable message length. Message latency can be easily handled by implementation of system level message priorities.

In the linear bus, each component is assigned a logical token number (which may or may not correspond to the physical organization of components along the bus). A component gains access to the bus when it receives a token. On receiving the token, the component is allowed to use the bus for a pre-determined, maximum amount of time. This time limit for transmitting data is determined by the Token-Holding Timer (THT).

When the component completes its message (or times-out) it passes the token to the next higher numbered component. It is the responsibility of the passing component to verify receipt of the token by the receiving component. If, after two tries, the receiving component does not accept the token, the sending component increments the token number and tries again. This process is reiterated until all addresses are exhausted. The token is then set back to zero.

Stations are allowed admittance to the logical ring on a periodic basis. Each station contains a Ring Admittance Timer (RAT). When this timer expires and there is a gap between the local station's address and that of its successor then the token is passed to the sequential address following that of the local station. The normal token passing rules are then applied. Therefore, if any of the stations in the gap desire admittance they will be granted an opportunity during this time period.

### 3.6.5. Major Documents

The characteristics of the SAE High-Speed Data Buses are described in the two draft specifications: (a) HSRB Standard, draft dated March, 1987; and (b) LTPB Standard, draft dated April, 1987. Both of which are available from SAE, Warrendale, Pennsylvania.

To minimize the potential for error due to latency, the node data/status message could include a "time tag" generated when it is sent from the node select computer. When each successively higher level within the architectural hierarchy generates a message or command, it would automatically pass along the time tag of the node data/status message. When the message arrives at the various destinations, within the hierarchy, a comparison is made of the current and new node data/status values and the time tag to ascertain the validity of the command.

In general, the actual latency of a message within a given architecture is determined by the rate at which the bus structure (either autonomously or centrally controlled) allows a "sending" terminal the opportunity to "latch-on" to the bus in order to transmit its message/data. For a centrally controlled bus, to obtain the least possible (i.e., minimum) latency, the controlled bus could be configured to (1) continually poll the terminals within the bus structure, (2) respond to the service request bit in the terminal status word, and (3) initiate the terminal-to-terminal (or terminal-controller) message transfer.

An increase in the distribution of processing tasks to more specialized computers is occurring. Given this trend away from a central general purpose computer, an event-based scheduling scheme (i.e., scheduled or unscheduled system interrupts such as a key press, a flag operation, etc.) may become a good alternative for some applications. When task scheduling is based on events rather than time, the latency of a message becomes more critical. The continuous polling technique is an effective way to reduce the message latency.

For an illustration of the event based scheduling, refer to the local display bus of figure 4.2-1. The display computer is normally operating in response to messages from the application computer. Its bus interface, which controls the local display bus, is continuously polling for keypad entry. When the keypad is pressed, a message is sent back to the display computer signaling an event to which the display computer must respond. The display computer will break out of its normal cycle, process the keypad message, and upon completion of this processing will have available keypad information that can be sent to other devices on the mission computer bus.

In this application two advantages are obtained from the event based scheduling and continuous polling. The latency of the message as it passes from a local bus to a higher level bus is minimized. Component faults in the communication system are identified early to provide time for management of the failure. For example, a simple management scheme would be to retransmit if the status response were not returned with the message error bits clear. A disadvantage of event based scheduling and continuous polling is that testing is more difficult due to the tester's inability to repeat a particular condition. When all scheduling is time based, then a repeatable test scenario can be generated. System response can therefore be accurately evaluated. When operation is based on asynchronous events, only a statistical comparison of results from multiple tests is valid.

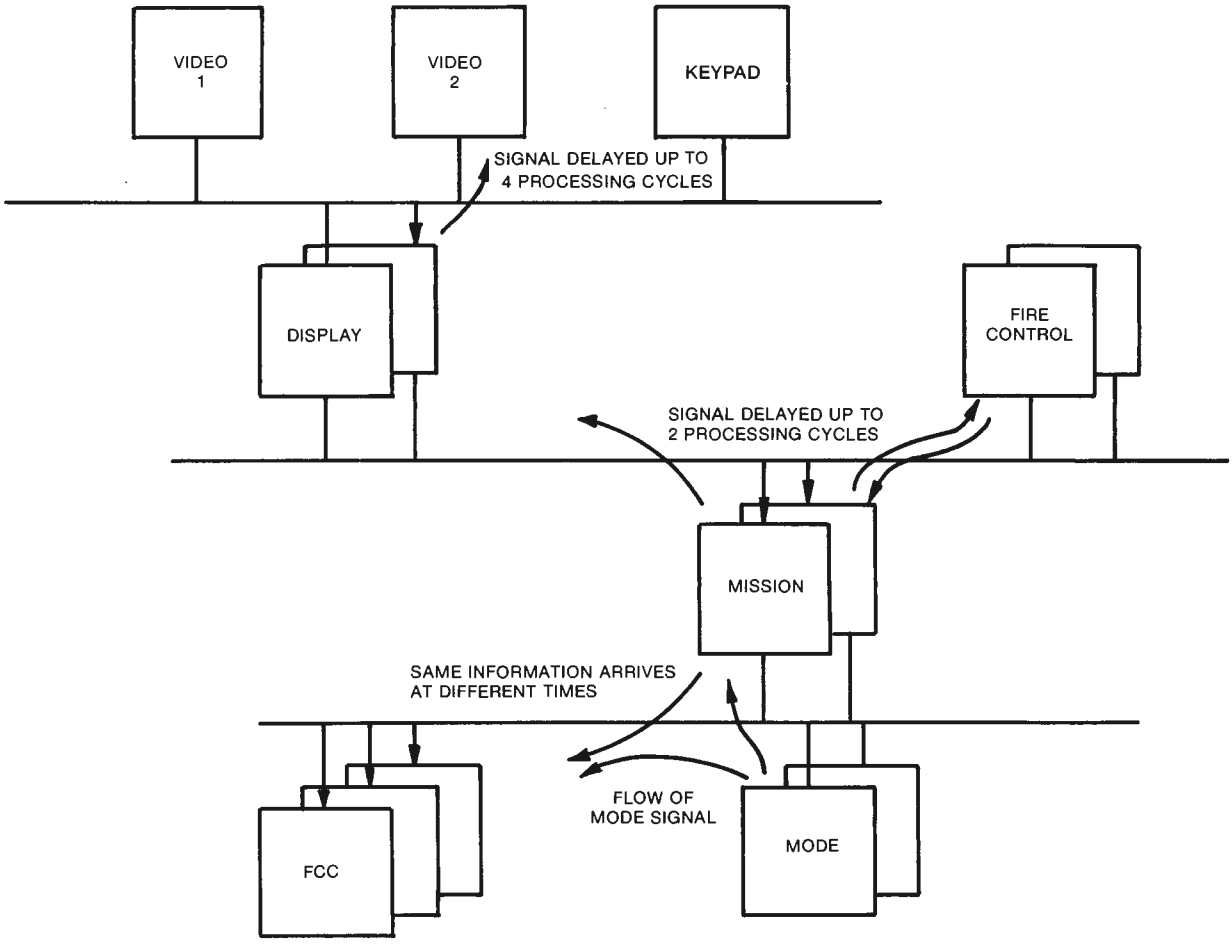



FIGURE 4.2-1. DATA LATENCY ILLUSTRATION IN HIERARCHIAL ARCHITECTURE

On a single hierarchical level, there are several protocols that can be used within the bus architecture framework. The choice of protocol affects message latency. For example, the use of a stationary master that polls all terminals on a regular basis provides minimum latency for a small number of terminals on the bus.

Alternatively when bus control is exchanged among a limited set of master computers, potentially greater latency is possible (depending upon the message table orientation of each master computer). If bus control capability exists in every terminal that may have a time critical message, the message latency will be in the range of several (2-4) ms. If continuous polling is done between message transmissions, latency improves; however, a large bandwidth penalty is paid. Continuous polling can only be used on buses with low activity levels.

4.2.3. System Delays

Average transfer delay is defined as the sum of delays resulting from queuing delay, access delay, and transport delay.

- Queuing Delay.

  Queuing delay is characterized by message arrivals and arrival rate. It is represented (characterized) as a Poisson Distribution. The mean queuing delay consists of the average delay incurred due to a message waiting for a previous message within the BIU to be serviced. The BIUs are effectively a single server queue, and therefore the queuing delay is a delay imposed on the user due to the BIU transmit buffer being full. This delay neglects the user/BIU message processing rate limitations and is dependent only on the message interarrival time as determined from the offered load.

- Access Delay.

  In the case of the CSMA/CD protocol, the mean access delay is determined by considering the two inherent access modes. The delay due to the random mode and the delay due to the ordered access mode are factored with the probabilities of being in their respective states and combined to equal the mean access delay.

  For the random access delay there are two components of delay: (1) delay due to the bus being busy, and (2) delay due to a collision. For the bus to appear busy, at least one other message must arrive before the message that encounters the media active state. Therefore, the probability of the bus being busy is the probability of two or more arrivals within the same time window.

  The probability of a collision can be described by the probability of two or more simultaneous arrivals. The delay due to a collision is determined by the time required to recognize contention, issue the jamming signal (approximately 1 microsecond), wait for the appropriate gap time, and then wait for the appropriate timeslot. Because the load distribution is assumed to be equal among the network BIUs, the average delay for the

timeslot count to reach the assigned time is one-half the total scan time for the timeslot sequence as determined by the loading conditions.

Looking at the access delay encountered by a message under the ordered access mode, two conditions are possible: (1) message ready before the timeslot arrives, or (2) message ready after the appropriate timeslot has passed. For an equal load distribution, the probability of each case is 0.5.

- Transport Delay.
  - The transport service time is determined by the transmission rate, the message length, and the overhead required for each transmitted packet. The overhead includes the following:
  - Gap time between messages.
  - Turnon time.
  - Packet encapsulation.
  - Acknowledge turnon.
  - Propagation delay.
  - Acknowledge message.

System level fault management is further facilitated by the monitoring of network statistics at each node. During operation, the BIUs collect the following statistics:

- Number of collisions.
- Number of collisions during own transmission.
- Number of packet rejects due to decoder buffer full.
- Number of successful transmissions.
- Number of unsuccessful transmissions.
- Number of data transmissions received.
- Number of status responses received.
- Number of commands received.

## 4.3. Failure Modes and Effects

Operation of digital aircraft requires the proper function of a number of interconnected systems, subsystems, and components within the framework of an integrated bus structure. Intermittent or erratic behavior or total failure of

one or more modules can impact the ability of the aircraft to perform its intended function. In some cases, the impact will be transparent as the fault is automatically detected, the failed module identified, and a redundant "like element" (similar or dissimilar) activated or "switched-to" automatically. Continued successive failures (or in the worst case, multiple simultaneous failures) could result in increased pilot workload, loss of function, or in the most severe case, the total loss of aircraft.

Because of the interactive relationships among systems and subsystems in aircraft, failures may propagate. Errors may affect not only the subsystem in which the failed module is embedded but also other subsystems to which erroneous data is transferred. This failure propagation potential between multiple systems/subsystems is greatly magnified by the differing levels of functional integration where data and information are exchanged between and among subsystems (using bus architectures and structures) as a requirement for normal operation.

Failures that could cause loss of essential mission capability or loss of aircraft must be protected against by using equipment redundancy, analytical redundancy, or functional redundancy to provide for continued operation after one or more failures. The redundancy may be applied at the system level (multiple buses or FCCs), at the sensor level (redundant Inertial Navigation System (INS), Altitude Heading Reference System (AHRS), Digital Air Data Computer (DADC), etc.), or at the module level (multiple similar or dissimilar microprocessors located in multiple processor subsystems). Failures that result only in some loss of function, restricted operation, or increased pilot workload, may or may not require redundancy. Redundancy requirements depend on the exact nature of the loss and the probability that such a loss will impact aircraft performance or aircraft flight safety. Failures that reduce the level of hardware redundancy or analytical redundancy, without loss of functional capability, may be able to be tolerated without performance degradation.

Failures can result from external disturbances or internal malfunction and can be either transient or permanent. Transient faults can often be ignored if the system is designed to tolerate such faults. In other cases, a transient fault can cause a more serious failure, such as the interruption of an instruction sequence in a computer, which in turn could cause a time-out or retry sequence, resulting in the completion of the computation using "stale" data. Permanent failures, on the other hand, must be recognized as such, and action must be taken to reconfigure around the failure.

Environmental effects can often be the cause of the failure. In the case of wire buses, heat, power supply surges (spikes), or low voltage levels could cause permanent or intermittent loss of an electronics unit or corruption of the bus data, which in turn would cause incorrect data and/or information to be passed to another unit in the hierarchy. Loss of electrical integrity (due to faulty shielding, grounding, or loss of cable integrity) could result in susceptibility to electromagnetic radiation, thus causing erroneous or erratic behavior.

In general, failures may exist in any one of the five functional elements relating to the integration of two or more subsystems. These functional elements include the following:

- Computers that process the data exchanged between subsystems.
- Data bus interfaces.
- Data bus(es).
- Input/output devices that govern the transmit/receive functions.
- System software.

Table 4.3-1 (McSharry, 1983) summarizes the potential faults which can cause intermittent or erratic behavior or even total failure of the networked architecture to communicate data and information to the various systems/subsystems within the structure.

The following considerations concerning the possible failures of digital data buses must be taken into account: First, transmission failures that may occur; second, effects on subsystems that are connected to the data bus by a bus controller or remote terminal failure; and third, effects of multiplex hardware failure. The navigation system must be self-contained and the aircraft must not become "lost" because of any type of transient. These safety requirements lead to subsystem requirements to store critical data in multiple locations and to recover rapidly from failures and upsets.

There are three failure modes, as follows: First, no transmission; second, incorrect transmission; and third, failure to relinquish control. A fault with these failure modes and some of the related causes are shown in figure 4.3-1.

## 4.4. Transmission Error

If the multiplex terminal hardware detects either an invalid word or a transmission discontinuity, the word and message are to be considered invalid. This message invalidation requirement may cause some systems (i.e., Electrical Multiplex (EMUX)) a problem. Since the EMUX systems usually have bit-oriented data rather than word or multiple word (message) oriented data, errors in a word following the reception of good data will invalidate good data.

Message completion failures should always be detected in a multiplex system. They are detected by the bus controller when the status word is suppressed or the message error flag in the status word is set. The message error flag removes ambiguity as to whether the error occurred before the message was validated by the remote terminal or in the response to the message.

Data transmission errors are handled by special error-handling interrupt software. The software will indicate whether (1) the command is to be retried, (2) the bus is to be used for the retry, and (3) the transmitted data (if any) should be invalidated. Tables 4.4-1 and 4.4-2 (MIL-STD-1553 Designer's Guide,

TABLE 4.3-1. DIGITAL DATA BUS SYSTEM FAILURE MODES AND EFFECTS
(Adapted from McSharry, 1983)

| 1.0 COMPUTER FAULTS | 2.0 DATA BUS AND INTERFACE | 3.0 INPUT/OUTPUT | 4.0 SYSTEM REDUNDANCY/MANAGEMENT FAILURES |
|---|---|---|---|
| 1.1 Computer Hardware Faults | 2.1 Data Bus Hardware | 3.1 Input/Output Device | 4.1 Design Failure (Fault) |
| 1.1.1 Central Processor Unit (CPU) Failure | 2.1.1 Open/Short-Single Bus/Multiple Buses | 3.1.1 Hardware Failures | 4.1.1 Synchronization-Time Skews (Comparison Monitoring of Time Tag Data) |
| 1.1.2 Memory Failure (Error) | 2.1.2 Interface Unit Failure | 3.1.1.1 Sensor, Sensor Output Transducer, Discrete Signal Input Failure | 4.1.2 Generic Failure Modes - Power Interrupts, Software Design Faults, Common Mode Failures |
| 1.1.3 Discrete and Analog Input/Output Failure | 2.1.2.1 Failure to Respond | 3.1.1.2 Induction of Noise (or Other Oscillary Disturbance) Due to EMI/EMC | |
| 1.1.4 Clock Failure | 2.1.2.2 Response to Wrong Address (Failure Diagnosis: Good = Defective) | 3.1.2 Software Faults (Errors) | |
| 1.2 Computer Software Failures (Faults) | 2.1.3 Bus Controller Unit Failure - Site Selection Error | NOTE: Typical Failures Include: | |
| 1.2.1 Failure to Isolate Faulty Entity | 2.1.3.1 Multiple/Redundant Controller Selection - Hardware Arbitration | o Hardover to Max/Min Values<br>o Steady Zero Values<br>o Bias Offset<br>o Reduced Dynamic Response<br>o Sensitivity Change | |
| 1.2.2 Isolation of Good Component as Failed | 2.1.3.2 Multiple/Redundant Controller Selection - Software Algorithm Fault (Error) | | |
| 1.2.3 Processing Erroneous Data | 2.1.3.3 Non-Centralized Controller Hardware Time-Out (Output Stage) Failure | | |
| 1.2.4 Failure to Complete Task - Exceed Task Time Allocation | 2.1.4 Improper Decoding/Encoding of Address, Command, Data Word Structures | | |
| 1.2.6 Observability | | | |

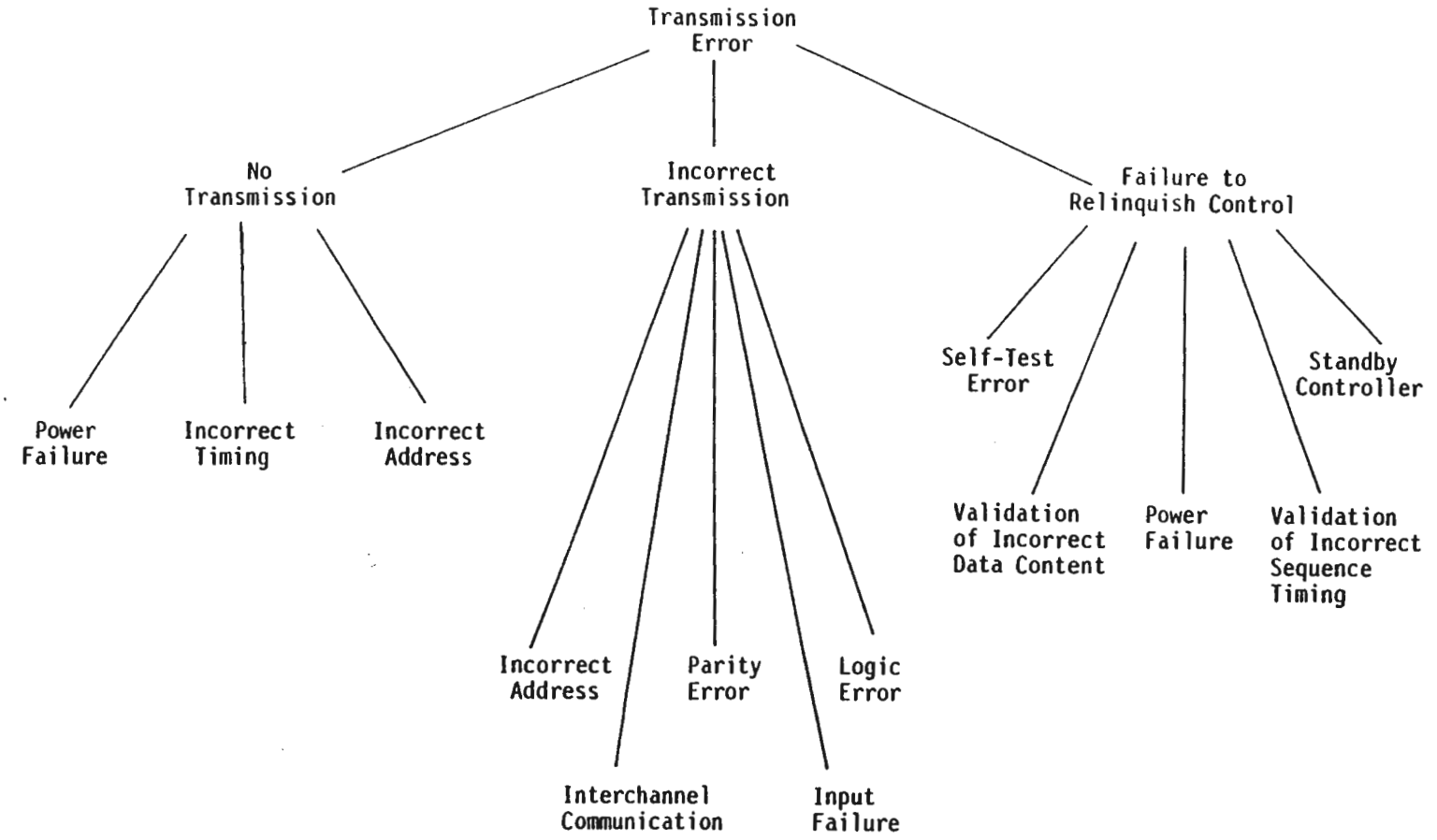



FIGURE 4.3-1. SIMPLE FAULT TREE EXAMPLE

TABLE 4.4-1. FAILURE CLASSES [MIL-STD-1553B]
(MIL-STD-1553 Designer's Guide, 1983)

| ERROR IDENTIFICATION | BUS SYSTEM | SENSOR |
|---|---|---|
| a. Message error | Transmission from bus controller to to terminal was decoded with error condition by receiving remote terminal | -- |
| b. Busy | Remote terminal unable to transmit or receive data at this time | Remote terminal and sensor unable to transmit or receive data at this time |
| c. Subsystem flag | -- | Sensor failure preventing proper sensor actions |
| d. Terminal flag | Remote terminal failure preventing complete action by terminal | Remote terminal portion of sensor interface has failure preventing complete action by terminal |
| e. Parity error (incorrect odd parity) | Error in status word; data not usable | Error in status word; data not usable |
| f. Improper sync | Unknown problem -- ignore; continue to look for valid sync | Unknown problem -- ignore; continue to look for valid sync |
| g. Invalid manchester | Error in message -- ignore data in message | -- |
| h. Improper number of data bits and parity | Error in message -- ignore data in message | -- |
| i. Discontinuity of data words | Error in message -- ignore data in message | -- |
| j. No status word response | Unknown problem -- requires further investigation | Unknown problem -- requires further investigation to achieve error |

TABLE 4.4-2. TYPICAL ERROR-CORRECTION TECHNIQUES [MIL-STD-1553B]
(MIL-STD-1553 Designer's Guide, 1983)

| ERROR IDENTIFICATION TYPES | ERROR CORRECTION TECHNIQUE |
|---|---|
| 1. Bus system failures | |
| a. No status word response | Retry message on same bus n times |
| b. Message error | Retry messages on alternate bus n times |
| c. Parity error | Transmit status word mode code on each bus |
| d. Invalid Manchester | If necessary, transmit initiate self-mode code |
| e. Improper number of data bits and parity | Transmit BITE mode code |
| f. Discontinuity of data words | Analyze failure and determine corrective action, which may involve the following mode commands:<br>Shut down transmitter<br>Inhibit terminal flag bit<br>Transmit reset remote terminal mode code |
| g. Busy | Retry message on same bus after a fixed delay time |
| h. Terminal flag | If necessary, transmit initiate self-test mode code<br>Transmit BITE mode code<br>Analyze failure and determine corrective action, which may involve the following mode commands:<br>Shut down transmitter<br>Inhibit terminal flag bit<br>Transmit reset remote terminal mode code |
| i. Improper sync | Ignore and reset for valid sync |
| j. Subsystem flag | Retransmit the data (address/subaddress) to examine sensor BITE discretes or words |
| 2. Sensor failure | |
| a. Discretes<br>b. BITE data word(s) | Analyze failure and determine system-oriented corrective action |

1983) show the error identification types and the corresponding failure classes and error correction techniques.

- No Transmission. The user should listen to the bus it transmits on for its request address. If no bus controller activity is detected, the user should transfer listening to the other bus for its request address. If no activity is detected on the other bus, the user should continue toggling between the buses in search of bus controller activity.

- Incorrect Transmission. The most serious failure for the bus controller is erroneous transmission. An independent frequency source should be used by the bus controller to provide monitoring and detection of transmission frequency faults. The two common types of transmissions are broadcast (which is sent on all of the channels) and CR (which is sent to a specific address). An error in a broadcast transmission has the potential for system failure if it is incorrectly validated at each of the addresses. An error in a CR has a more limited effect since it only involves one address. Each receiver should incorporate isolation provisions to ensure that the occurrence of any reasonably probable internal LRU or bus receiver failure does not cause any input bus to operate outside of its specification limits (both under voltage or over voltage).

- Failure to Relinquish Control. Subsystem or terminal failures may be detected using Built-In Test (BIT) circuitry. These failures are reported by the setting of the subsystem flag bit or the terminal flag bit in the status word. In aircraft, dual-redundant buses are used, so a terminal failure may be isolated to one bus. Depending on the capability of the remote terminal hardware, the transmit BIT word mode code can be a powerful diagnostic aid. For each fault, the action to be taken must be determined, designed for, and implemented by the system.

Subsystem or terminal failures can also be detected without the use of the optional terminal or subsystem flags. Bad data or nonvarying data from a subsystem may be interpreted as a subsystem failure. Repeated message completion failures to a remote terminal via all possible data paths could be considered as a loss of the terminal function. The system software should be used to detect these failures.

Bus controller operation in the event of failure is important to an integrated data bus system. The primary bus controller should relinquish bus control whenever it suffers a power interruption that might cause erroneous outputs. The primary bus controller should detect its own bus control processing faults and remove itself as controller in a fail-passive manner. Similarly, the backup bus controller should recognize invalid control messages or the absence of valid control messages and revert to active bus controller status. Monitoring techniques should provide coverage for both hardware faults and software errors. Any undetected fault in the primary bus controller, which results in continuous erroneous transmission, will make all standby controllers ineffective. The bus controller is structured such that two independent faults must occur in order to cause erroneous transmissions.

## 4.5. Reliability for Flight Safety

Flight safety requirements allow no more than one unrecoverable failure in the flight control subsystem per $10^9$ flights. This failure rate is consistent with AC-25-13091 and is appropriate for integrated systems. The failure rate must encompass the entire flight control system including the necessary supportive electrical power, hydraulics, and any other subsystem used in the flight-critical capacity. When applied over the 2- and 3-hour mission duration of the aircraft, a maximum failure rate of approximately $5 \times 10^{-6}$ failures per flight hour (for a 2-hour mission) and $3.3 \times 10^{-6}$ failures per flight hour (for a 3-hour mission) can be allowed.

Figure 4.5-1 gives an example for the determination of the loss of bus control. The potential failures for the bus control example are given in table 4.5-1. The total failure rate must be equal to or less than the total allowable defined above. In the example, the loss of bus control, D, is

$$D = (E_1 + E_2 + E_3)(E_4 + E_5 + E_6) + E_7 + E_8 + E_9 + E_{10}$$

By substituting in the values from table 4.5-1, we obtain

$$D = 3.1012321 \times 10^{-5}$$

Therefore, in this example the data bus would fail to meet the reliability requirements for flight safety.

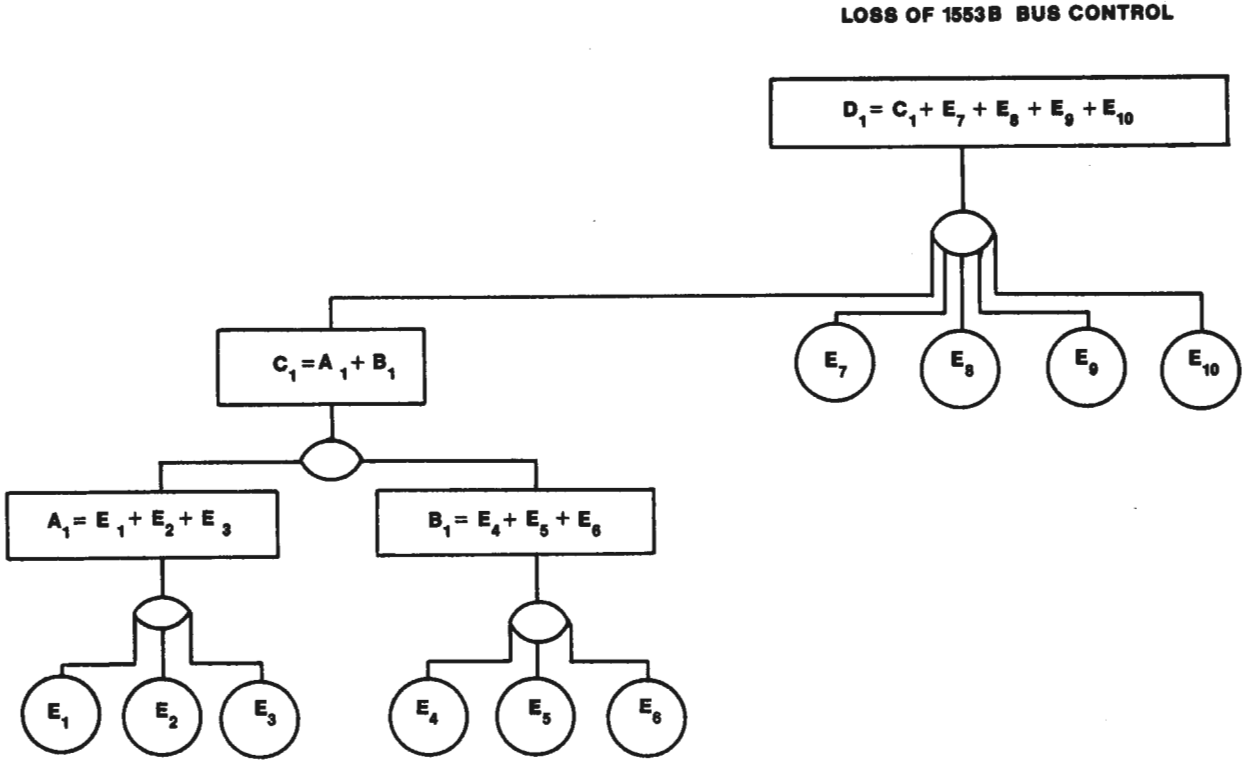



FIGURE 4.5-1. SINGLE CHANNEL-DUAL OUTPUT (BUSES A AND B) BUS CONTROL

TABLE 4.5-1. POTENTIAL FAILURES RESULTING IN LOSS OF BUS CONTROL SINGLE CHANNEL - DUAL OUTPUT (BUSES A AND B)

| ERROR | ERROR SOURCE | FAILURE RATE |
|---|---|---|
| $E_1$ | Bus A - Transformer Failure | $10^{-6}$ |
| $E_2$ | Bus A - Transceiver Failure | $10^{-4}$ |
| $E_3$ | Bus A - Decoder Failure | $10^{-5}$ |
| $E_4$ | Bus B - Transformer Failure | $10^{-6}$ |
| $E_5$ | Bus B - Transceiver Failure | $10^{-4}$ |
| $E_6$ | Bus B - Decoder Failure | $10^{-5}$ |
| $E_7$ | Single Encoder Failure | $10^{-5}$ |
| $E_8$ | Internal Control Logic Failure | $10^{-5}$ |
| $E_9$ | Interface Unit Failure | $10^{-5}$ |
| $E_{10}$ | Microprocessor System Failure | $10^{-6}$ |

# 5. FIBER OPTIC DATA BUS FOR AVIONICS INTEGRATION

## 5.1. Elements of Fiber Optic Data Bus

As stated in previous sections, the bus topology is the physical arrangement and interconnection of the various terminals. In a fiber optic bus, the elements utilized are optical couplers, fiber cable, connectors, and splices.

TABLE 5.1-1. OPTICAL BUS TECHNOLOGY LIMITS

| COMPONENT | FACTORS |
|---|---|
| Couplers | Losses<br>Number of Taps |
| Fiber | Fiber Type<br>Modal Noise<br>Connectors<br>Splicing<br>Reflections<br>Cabling |
| Optical Source | Power<br>Speed |
| Optical Receiver | Sensitivity<br>Intermessage Dynamic Range<br>Intermessage Response Time<br>Clock Recovery |
| Processing/Interface Logic | Speed<br>Power Consumption<br>VHSIC/VLSI & GaAs |
| Topologies | Performance<br>Reliability<br>Flexibility<br>Installation and Maintenance<br><br>Cost |

The design of these elements not only relates to system performance but also to system installation and maintenance. Because optical power losses occur whenever any of these components or functions are inserted in the optical path, performance is affected. Table 5.1-1 presents the components and factors that influence the limits of optical bus technology as it applies to buses used for avionics integration.

## 5.2. Optical Cables

Considerations involved in evaluating optical cables for a fiber optic data bus include fiber design (including modal noise and reflection effects) and cable type and construction.

## 5.3. Optical Path

Basically, the optical path is the fiber optic cable. In designing the proposed avionics architecture, the fiber cable must be selected for minimal loss (across the bus) and wide bandwidth. In addition, the fiber cable must be constructed for strength and endurance during the life of the bus architecture, ease of installation, and long term environmental performance.

## 5.4. Splices and Connectors

Interconnections between the fiber cable elements (controllers, remote terminals, junctions, etc.) can be made with either splices or connectors. Splices in the fiber cable are easier to incorporate and provide lower losses than connectors; however, splices are permanent. Connectors, on the other hand can be mated and unmated hundreds of times with virtually no degradation in performance. Therefore, in the development (and design) of a fiber optic based avionics architecture, optical couplers (connectors) should be utilized for bus interface connections to the physical bus. This will minimize downtime due to repair or change in the architectural structure when remote terminals are added or deleted. In the case of aircraft having pressurized bulkheads, several penetrations through these bulkheads may need to be made. At these penetrations, the fiber optic cable can be run straight through the bulkhead or an optical connector (coupler) can be used on each side of the bulkhead. The tradeoff, in this case, is between the ease of installation and rework using a connector (coupler) system or the lower loss and absence of reflections using a spliced or through cable.

## 5.5. Optical Couplers

The two basic types of optical coupling techniques that are considered for an optical data bus are star couplers and taps or tees.

In a transmissive star, N ports are designated as input ports, and N ports as output ports. The optical energy on any input port is split more or less equally among all output ports, with a splitting loss of 10 log N. Star couplers also have an insertion loss and a port-port variation of 1 to 3 Decibel (dB), each depending on the number of ports. Stars in excess of 100 ports have been fabricated; however, for minimal cost and port-port variations, the practical limit of current technology is 64 ports.

Directional couplers for tapping a transmitter and receiver onto a fiber optic bus are basically like a four-port transmissive star with an excess loss of 0.5 to 1 dB. Typically a tap into the receiver can be accomplished with a 90/10 or 95/5 split providing a 0.5 to 2.0 dB link throughput loss, respectively, and a 10 dB to 13 dB tap-off or reduction of the link power into the bus receiver. For tapping the transmitter into the bus, the throughput loss as well as the coupled transmitter power reduction is 3 dB in commercially available couplers.

## 5.6. Size

Of the available fiber options, the 100/140 micrometer ($\mu$m) or the 85/125 $\mu$m graded index fiber operating at a 0.05 $\mu$m signal length is optimum because:

- Their large core, high numerical aperture, and operation wavelength support many more modes, thus minimizing the modal noise limitation.
- Their large core enables greater Light Emitting Diode (LED) coupled power, thus extending the application of LEDs.
- Their core-clad geometry makes it easier to make low excess loss star couplers.

## 5.7. Reflections

Another consideration in the medium analysis is reflections. Reflections result from an index of refraction discontinuity at connectors, poor splices, or mismatched fiber types. For example, with a star coupler, the main signal passes through the link; however, part of the signal is first reflected at the star coupler dry connector (8 percent) and then again at the transmitter dry connector (8 percent). The resulting reflected signal is down 22 dB with respect to the main signal and delayed by as much as 1 microsecond (1 ns/m). This reflected signal becomes a problem if it overlaps the next bus transmission and shows up as noise superimposed on this data. Therefore, consideration must be given to minimize reflections.

## 5.8. Connectors

Optical connectors, which are suitable for use in a data bus, are low cost, easily installed, and typically low loss. The connector loss depends on the fiber size as well as the quality of the connector. For 100/140 $\mu$m fiber, losses vary from 0.5 to 1.5 dB depending on connector quality. Available multiway connectors have the advantage of simplifying a bulkhead penetration and provide quicker connect/disconnect of a multifiber cable. Although there is no fundamental reason for higher loss in a multiway connector, the losses in currently available connectors average approximately 0.5 to 1 dB more than the loss in a single fiber connector.

## 5.9. Splicing

For field installation, maintenance, and repair the elastomeric splicing system has been identified as the best currently available splicing technique.

## 5.10. Technology - Optical Bus Transmitters and Receivers

Fiber optic bus Transmitter/Receiver (T/R) design is driven by the goal of maximizing bus efficiency. This is necessary to fully utilize the benefits of the bus, minimizing "dead" time, and allowing transfer of significant quantities of data.

An efficient bus T/R is relatively easy to design. However, providing very quick transmitter power output stabilization and very short receiver settling time at the start of a message significantly increases the difficulty and complexity of the T/R design. A fast response clock recovery scheme is also critical in minimizing the amount of time used for non-data. In summary, the more time that is used to perform overhead functions, the less time there is to transmit data, and the lower the efficiency of the bus.

## 5.11. Maximizing Bus Efficiency

One of the principal considerations in maximizing bus efficiency revolves around the unique aspects of an optical transmission. Intensity modulation of an optical carrier provides a unipolar transmission channel, unlike electrical current transmission over wire which may be bipolar. Unipolar signaling causes a Direct Current (dc) shift between signal-on, and signal-off states, which will disturb the operation of conventional receiver amplifiers having ac coupling until the interstage coupling capacitances have had time to accommodate the shift. A similar dc shift occurs between small and large signals.

Thus to avoid a long settling time at the start of messages, receivers designed for data bus application either have a short ac coupling time constant to minimize the disturbance time, or employ dc coupling. If dc coupling is used, more complex circuitry is required for setting the data decision threshold for the received waveform. The shift in average power between signal and no-signal states also complicates laser optical source power stabilization, which is normally accomplished using average power feedback control.

## 5.12. Transmission Losses

Optical bus configurations have considerable and somewhat undefined transmission losses between source and detector, resulting from the couplers and connectors. When combined with source power and detector sensitivity variations, this gives rise to an uncertain received power level. A high-gain, wide-dynamic-range receiver is required. Since time is a premium, long term averaging is undesirable. Alternate methods for rapidly accommodating the dynamic range are required. This is a major concern for optical data bus receiver designers.

## 5.13. Receiver Losses

Three receiver types which provide simple, instantaneous adjustment to message levels are known. In the symmetrical clamp receiver, all signals are bit-by-bit clamped to the same low level. After amplification, decisions about the validity of data are based on the data being above a fixed threshold. Good dynamic range is achievable and start-of-message time constant delays are eliminated. This is not the case with conventional linear or limiting receivers. The technique operates well up to bit rates around 50 Million bits per second (Mbps). Above this level implementation problems arise. (The upper bit rate limit may be extended using lower capacitance hybrid construction.) This technique is a leading candidate for receivers operating at lower data rates.

A second fast response scheme uses a dc coupled receiver (to avoid ac coupling time constants) and a bit-by-bit adaptive threshold decision. The technique is ideally suited to very high data rate reception but dynamic range is limited by amplifier design. Optimum performance is limited by dc offset in the amplifier, which may be a limitation for wide temperature range operation.

High-bit rate reception may also be handled efficiently with a high-pass filtering receiver when the signal is any biphase code or other reduced low-frequency content code. This is because required coupling capacity time constants become small compared to the fixed bus intermessage dead time resulting from propagation delays. Appropriate filters have been designed with a linear phase response in the stop band, providing an intermessage response time as low as six bit times for Manchester coded data.

Conventional point-to-point system optical receivers have well defined sensitivity limits that may be calculated from the thermal and shot noise of the devices. For data bus receivers, a number of compromises in design are necessary to achieve fast response to messages. These generally result in less sensitivity. Similarly, wider dynamic range may generally be achieved in a receiver, which has a long period to adjust to changes in signal level, versus a receiver that is required to adjust almost instantaneously.

The receiver sensitivity is affected largely by the type of photodetector and by preamplifier design. A silicon avalanche photodetector offers greatest sensitivity (at 0.85 $\mu$m); therefore, preamplifier design is less critical. At 0.85 $\mu$m, a silicon PIN diode with a sensitive preamplifier has approximately 10 dB less sensitivity.

## 5.14. Transmitter Losses

For relatively low rate transmission, i.e., <10-50 Mbps, little difficulty exists in designing a transmitter circuit using LEDs. Data modulation may be dc coupled through to the LED and any data format or message length may be accommodated. Very high data rate transmission requires the use of a semiconductor laser diode to achieve the required modulation rate and sufficient launched optical power to provide reliable reception after the transmission losses. Lasers require a more complex driver circuit to ensure that the drive current compensates for temperature and aging of the source, and is correctly

prebiased during transmission to avoid data distortion resulting from lasing turnon delay. Effective compensation of the drive current requires feedback control of the launched signal. Feedback control commonly operates by stabilizing the average transmitted power in continuous transmission point-to-point systems. With the burst nature of transmission in a bus system, averaging is not as convenient and requires a long preamble for the laser power to initially stabilize.

Any data bus transmitter design must include an override control, which provides a positive curtailment of transmission in the event of a nonself-correcting fault. An external timeout circuit or protocol function controls this override function.

## 5.15. Optical T/R Power Margin

A key element in the design and optimization of any fiber optic link including a data bus is the system power budget analysis. Such an analysis is important not only to ensure that there is adequate optical power at any given receiver under all conditions, but also to ensure, particularly in a data bus, that there is not too much optical power at any given receiver.

There are three basic elements to a power budget analysis: system losses, optical source output power, and optical receiver sensitivity. The maximum allowable system loss can be derived for a transmitter combined with a realizable receiver. Output powers of -6 Decibels per meter (dBm) can be achieved with high radiance LEDs coupled to 100 $\mu$m core fiber with an Numerical Acceptance (NA) of 0.3.

## 5.16. Topology Analysis

Using the practical technical limits as discussed in the previous sections, an analysis of various fiber optic data bus topologies or configurations was performed to evaluate the number of terminals possible at various data rates.

The topologies examined included:

- Linear.

  Inbound-outbound (loop or ring).
  Bidirectional (open-ended).
  Active.

- Star.

  Transmissive.
  Reflective.
  Star-star.
  Active star-star.

- Hybrids.

  Star-loop.
  Loop-star.

Since active stars and active rings are essentially point-to-point links, bus losses are not the limiting factor on the number of terminals, nor is dynamic range a factor in receiver design.

For this initial, first-order analysis, the best case performance for splices, connectors, and couplers was assumed. This approach "brackets" the problem by defining the best possible performance of a particular topology implemented with currently available (or nearly available) technology.

A passive transmissive star bus is the most efficient topology, because the power from any transmitter is distributed evenly among all receivers. In addition, there is only one coupler-insertion loss between any given T/R pair.

The principal disadvantage of a bus with a single star is that the cables from all T/R modules must be run to the star. In an aircraft, this increases the initial installation cost due to the increased number of bulkhead penetrations required. In addition, there is little flexibility for adding new terminals at arbitrary locations. One solution to this is to provide a distributed bus topology such as a star-star or a star-linear topology. The performance of the star-star topology can be easily improved by adding a single repeater (or two for redundancy) at the central star.

Two hybrid topologies combining stars with a linear bus concept were investigated because they provided four separate nodes with the potential for improved performance over a simple linear bus. The first is a star-loop; the second a loop-star.

Initial analysis of these revealed very little reduction in bus loss over a simple linear loop, and therefore a detailed analysis was not performed. The loop-star or distributed star topology can be effective, however, with active repeaters between the stars.

The only viable passive topology for 128 terminals is a star; however, an active linear bus, active star, or active star-star are viable implementations for 128 terminals at 300 Mbps. The active star-star appears optimal because it:

- Minimizes cabling and bulkhead penetrations with four (or more) nodes for concentrated location of terminals while enhancing flexibility.
- Minimizes number of repeaters and therefore cost and maintenance.
- Prevents a single point failure that will disable the entire bus.
- Allows use of star couplers with 6-32 ports, reducing the cost and increasing the performance and reliability of the couplers.

## 5.17. Fiber Optic Network Based Losses

A typical set of requirements for an avionics multiplexed bus of a commercial transport could include anywhere from 32 to 128 terminals. Data rates could be in the 10-100 MHz (or Mbps) range. The bus probably would be bidirectional, using a broadcast type mode in which any terminal might transmit data to any other terminal in the network. Various topologies for such a bus have been discussed earlier; however, the most probable topology for such an architecture would be a star-coupled topology due to the fact that it can be implemented without the use of active repeaters. This results in higher reliability, lower maintenance, and reduced losses in the optical path.

Table 5.17-1 presents a typical loss budget calculated for an approximately 60-terminal, star-coupled transmission network. From this table, it can be seen that the bus network will require high optical output from the transmitter and high receiver sensitivity to assure that the integrity of the data is maximized. To insure the high integrity, the bus optical components will have to be selected to be consistent with simple straightforward system design at both the transmitter and receiver ends.

TABLE 5.17-1. STAR-COUPLED NETWORK LOSSES

| COMPONENT | MINIMUM LOSS | MAXIMUM LOSS | COMMENT |
|---|---|---|---|
| Fiber | 0.0 dB | 1.0 dB | 50 m. terminal to star maximum, 5 dB/km |
| Connectors | 0.4 dB | 8.0 dB | .1 dB to 1.0 dB each, 4 to 8 total terminal to terminal |
| Star Coupler | 17.1 dB | 21.1 dB | Typical |
| TOTAL | 17.5 dB | 30.1 dB | |
| Optical Dynamic Range: 12.6 dB | | | |

## APPENDIX A - MODULATION AND SIGNALING METHODS

Modulation changes the carrier signal's characteristics. An unmodulated carrier is a periodic signal containing no information. Modulation imposes information on the carrier. Analog carriers can be represented by the sine wave:

$$a = A \sin (2\pi f t + \phi)$$

where

a = voltage at time t
A = maximum amplitude
$f$ = frequency
$\phi$ = phase

The carrier's amplitude, frequency, and phase can all be modified. Amplitude Modulation (AM) refers to modification of the carrier's amplitude for information transfer. Frequency Modulation (FM) refers to modification of the carrier's frequency. Phase modulation ($\phi$M) refers to modification of the carrier's phase.

- Frequency Modulation.

  FM is not commonly applied to information transfer in digital avionics systems.

- Amplitude Modulation.

  The simplest form of AM varies the magnitude of the signal (A) from zero to a fixed peak-to-peak voltage. The zero level represents a binary zero and the maximum voltage represents a binary one. Figure A-1 (Held, 1986) shows the correspondence between the AM signal and the digital data it represents. AM by itself is normally used for very low data rates. For higher transmission rates, AM may be combined with $\phi$M.

- Phase Modulation.

  $\phi$M varies the phase of the carrier signal. Figure A-2 shows an example of $\phi$M.

In terms of bus transmission of digital data, modulation techniques and signaling methods refer to the type of coding used to transmit information through the bus. All of the subsystems connected to the bus must use the same coding method. There are a number of codes that can be used, which differ with respect to advantages and disadvantages. This brief section of the tutorial looks at the coding methods likely to be used by aviation-oriented data buses.

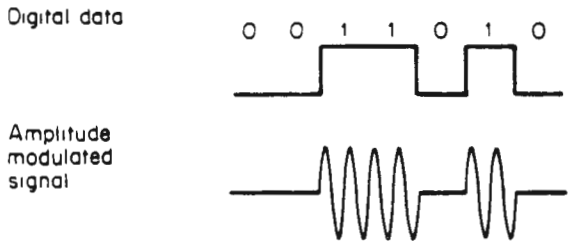



FIGURE A-1. AMPLITUDE MODULATION
(Held, 1986)

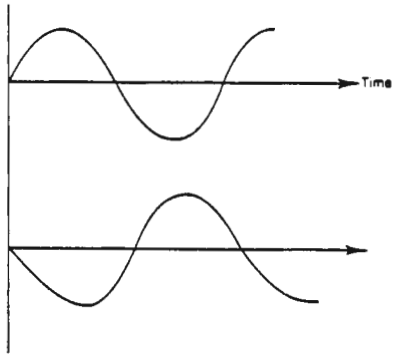



FIGURE A-2. PHASE MODULATION

Code is categorized as either single-density or double-density. Double-density codes include Delay Modulation (DM), Modified-Frequency Modulation (MFM), Group Code Recording (GCR), Zero Modulation (ZM), Enhanced Non-return to Zero (ENRZ), and Randomized Non-return to Zero (RNRZ), most of which are not appropriate for aircraft data buses. An exception is DM coding.

DM coding is complex both in hardware and functional characteristics. Its functional complexity stems from the requirement that there must be at least one signal transition for every two-bit interval and no more than one transition per bit. Because of this, some synchronization capability is still provided, but at a lower modulation and bandwidth requirement than required for the single-density codes. Consequently, it cannot be used effectively with buses requiring high data transfer rates.

The most common single-density codes are Non-return to Zero (NRZ); NRZ-Inverted (NRZ-I), which is sometimes referred to as NRZ-M; NRZ-Dual Level (NRZ-L) ratio; and biphase. Biphase covers several subcategories: Manchester II, FM, and Phase Encoding (PE). Since these single-density codes are self-clocking, the clock is represented by level transitions, which take place even if data transitions do not. NRZ, Return to Zero (RZ), and biphase are categorized by the suffixes L (level), M (mark), and S (space). An -L suffix indicates that data are represented by different levels; -M and -S suffixes indicate that data are represented by the presence or absence of transitions. In codes designated -M, a "1" (defined as a mark) occurs with a level transition; "0" is no transition. The converse is true for codes designated -S.

NRZ codes remain constant throughout a bit interval and either use absolute values of the signal elements or differential encoding where the polarity of adjacent elements is compared to determine the bit value. This method lacks independent synchronization and error-detection capabilities but provides efficient usage of the bandwidth.

RZ codes return to a binary "0" level at one-half the bit interval for binary "1" signals, requiring a higher bandwidth for an equivalent NRZ data rate.

Biphase codes include the Manchester and Differential Manchester techniques. At least one signal transition is required every bit interval, providing a self-clocking mechanism. The absence of the expected transitions may also be used for error detection. With two possible transitions per bit time, there is a corresponding increase in the bandwidth required.

Multilevel binary encoding schemes use more than two signal levels. One method is bipolar, which has no synchronization capability but does provide some error detection by requiring successive binary "1"s to be of opposite polarity.

Most of the aircraft data buses use biphase codes like Manchester II, which is self-clocking since the data and clock are included in a single serial data stream. In clocked systems, the clock defines the size of the data-bit cell; however, in nonself-clocking systems, speed fluctuations cause the data track to vary relative to the speed of the clock. Over a period of time, the clock will appear to speed up or slow down and will improperly define a data bit cell.

With self-clocking, everything stays synchronized. The mid-bit transitions of Manchester code help detect transmission errors.

Table A-1 summarizes the major features for some of the popular single and double-density codes. The encoded waveforms in figure A-3 illustrate patterns for an identical binary input produced by each form of encoding.

TABLE A-1. IMPORTANT PARAMETERS OF ENCODING TECHNIQUES

| Code | Bandwidth $f_l$ | Bandwidth $f_h$ | Storage Efficiency | Self-Clocking | dc Presence | Band Speed Ratio | Preamble for Synch-ronization |
|---|---|---|---|---|---|---|---|
| NRZ | 0 | 0.5f* | N.A. | No | Yes | N.A. | No |
| RZ | 0.25f | 1.0f | 50% | No | Yes | 4 | Yes |
| NRZ-S | 0 | 0.5f | 80% | No | No | 9 | No |
| Ratio | 0.75f | 1.5f | 33% | Yes | No | 2 | No |
| Biphase | 0.5f | 1.0f | 100% | No | Yes | 2 | Yes |
| Double-density | N.A. | 0.5f | 100% | No | Yes | 2 | Yes |

*Bandwidth in terms of the fundamental frequency of the data rate.
N.A. = Information Not Available

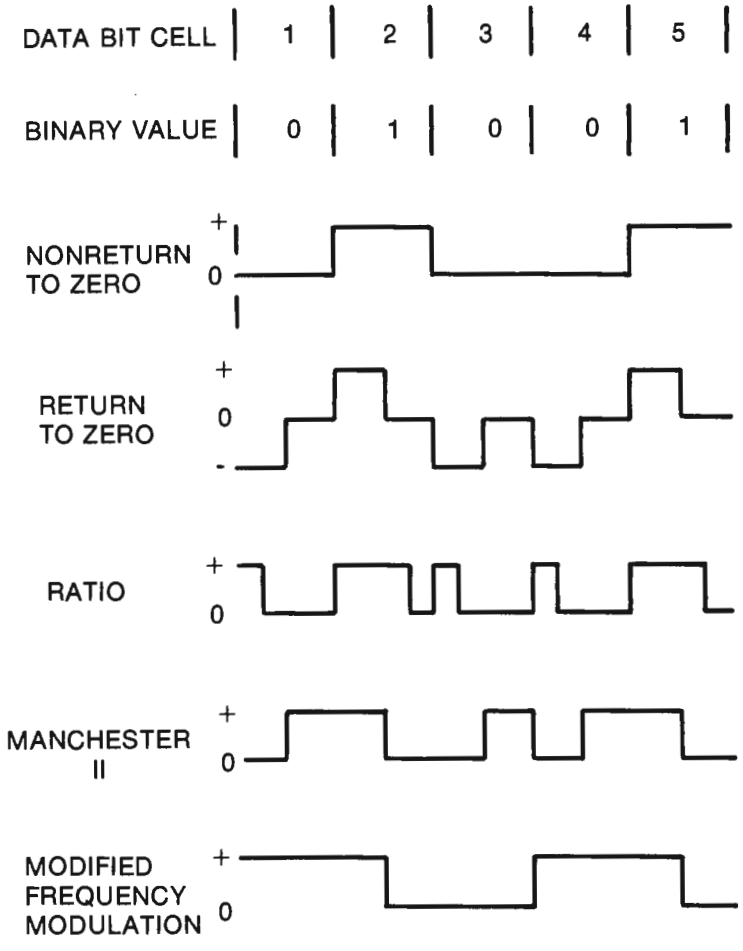



FIGURE A-3. WAVEFORMS GENERATED BY FIVE DIFFERENT ENCODING SCHEMES FOR A FIXED BINARY SIGNAL

BIBLIOGRAPHY

ARINC Specification 429-5, Mark 33 Digital Information Transfer System, April, 1981.

ARINC Specification 629 (Multi-Transmitter Data Bus), Part 1 - Protocol Description and Part 2 - Label and Word String Definitions, Drafts dated September 1987.

Aydin Vector. MIL-STD-1553: Theory and Application, 2/84-2M, Newtown, PA, 1984.

Eldredge, D., and E. F. Hitt, Digital System Bus Integrity, FAA Technical Report, DOT/FAA/CT-86/44, Atlantic City Airport, New Jersey, March 1987.

General Aviation Manufacturers Association Specification, Washington, DC, February 1986.

Held, G. Data Communications Networking Devices. John Wiley, New York, 1986.

ILC Data Device Corporation. MIL-STD-1553 Designer's Guide, New York, 1983.

Jennings, R. G. "Avionics Standard Communications Bus: Its Implementation and Usage," Proceedings of the AIAA/IEEE Seventh Digital Avionics Systems Conference, Fort Worth, Texas, October 1986.

Kiernan, J. J., and J. R. Sims, "Standardization in Military System Architecture," Standardization in Military Avionics Systems Architecture, Proceedings of an IEEE/AES-S Seminar, Dayton, Ohio, November 28, 1979.

MIL-STD-1553B, Aircraft Internal Time Division Command/Response Multiplex Data Bus, Specification, 1978.

McSharry, M. E. Architecture Tradeoffs for Bus-Integrated Control Systems, NADC-82041-60, Aircraft and Crew Systems Technology Directorate, Naval Air Development Center, Warminster, PA, October, 1983.

Rich, B. A., et al., Multibus Avionic Architecture Design Study (MAADS), AFWAL-TR-83-1141, AFWAL, Wright-Patterson Air Force Base, OH, October 1983.

Society of Automotive Engineers, SAE High-Speed Ring Bus Standard, (Draft) March, 1987.

SAE Linear Token Passing Bus Standard, (Draft) April, 1987.

Schmitter, E. J., and P. Baues, "The Basic Fault-Tolerant System," IEEE Micro, 1984.

SCI Systems, Inc. The MIL-STD-1553 Multiplex Applications Handbook, Huntsville, AL, February 1982.

Shaw, J. L., H. K. Herzog, and K. Okubo, "Digital Autonomous Terminal Access Communication," Proceedings of the AIAA/IEEE Seventh Digital Avionics Systems Conference, Fort Worth, Texas, October 1986.

Integrated Application of Active Controls (IAAC) Technology to an Advanced Subsonic Transport Project - ACT/CONTROL/GUIDANCE SYSTEM STUDY - VOLUME I. Final Report, NASA Contractor Report 165963, NASA Langley Research Center, Hampton, VA, December 1982.

ADDRESSING CAPACITY. The number of components addressable by the protocol used on a given data bus.

ASYNCHRONOUS MESSAGES. Electronic signals with transmission times that are not known a priori. These may include priority signals requiring immediate access to the bus.

BIT TIME. The time it would take to transmit one bit. Usually this is "blank" time when nothing is being transmitted. One nth of the bus speed (i.e., on a 1 kHz bus, the bit time is $10^{-3}$ seconds).

BLOCK TRANSFER. A data transfer mode allowing the transfer of variable length data blocks.

BROADCAST CAPABILITY. The capacity to transmit messages to all terminals simultaneously.

CAT IIIa LANDING. One of several landing categories defined in FAR 91. CAT IIIa implies the need for an instrument landing approach.

CENTRAL CONTROL. Control from one master, whether stationary or non-stationary.

COMMAND/RESPONSE. "Operation of a data bus system such that remote terminals receive and transmit data only when commanded to do so by the controller." (MIL-STD-1553 Designer's Guide, 1983, p. II-3).

CONTENT ADDRESSING. The system of identifying message recipients based on information embedded in the message. This is in contrast to destination terminal addresses.

DATA BUS. A system for transferring data between discrete pieces of equipment in the same complex.

DATA LATENCY. The delay from the time when a piece of information becomes available at a source terminal to the time it is received at the destination.

DATA LINK ASSURANCE OF RECEIPT. The guarantee of good data through the data link level.

DETERMINISTIC. A system where all parameters are known, as opposed to a statistical system where the outcome is subject to the laws of probability.

DISTRIBUTED CONTROL. Concurrent control from multiple points in the data bus system.

FAULT CONTAINMENT. The capacity of a system to prohibit errors and/or failures from propagating from the source throughout the system.

FAULT DETECTION. The capacity of a system to determine the occurrence of erroneous operation.

FAULT ISOLATION. The capacity of a system to isolate a failure to the required level so it can reconfigure.

FAULT TOLERANCE. The capability to endure errors and/or failures without causing total system failure.

INITIALIZATION. Setting the beginning parameters and values on system power-up. For redundant systems this includes setting the initial configuration of the system.

LABELED ADDRESSING. The system of identifying message recipients based on labels. This is in contrast to destination terminal addresses.

MESSAGE STRUCTURE. The organization of both protocol and data information in a message.

MONITORABILITY. The capacity of the protocol to be viewed passively to allow observation of the dynamics of the protocol.

NETWORK CONTROL STRATEGY. The solution proposed by the designer in addressing his specific problem (design flexibility).

NUMERICAL APERTURE. The angle of acceptance of light from a light source for a given fiber optic cable.

PARAMETERIZATION CAPABILITY. A measure of how well the attributes of the protocol can be described by parameters.

RECONFIGURATION. The capacity of a system to rearrange or reconnect the system elements or functions.

SYNCHRONOUS MESSAGES. Messages transmitted at a known a priori sequence and time or time interval.

SYSTEM INTEGRITY. The degree to which a system is dependable.

TESTABILITY. A measure of how well the protocol supports completeness of testing and the protocol's ability to produce repeatable or predictable results.

THROUGHPUT. The productivity of a data processing system as expressed in computing work per minute or hour.

## ACRONYMS AND ABBREVIATIONS

| | |
|---|---|
| ac | Alternating Current |
| AE | Avionics Equipment |
| AES-S | Aerospace and Electronic Systems Society |
| AFWAL | Air Force Wright Aeronautical Laboratory |
| AHRS | Attitude Heading Reference System |
| AIAA | American Institute for Aeronautics and Astronautics |
| AM | Amplitude Modulation |
| ARINC | Aeronautical Radio, Incorporated |
| ASCB | Avionics Standard Communications Bus |
| BIR | Benchmark Information Rate |
| BIT | Built-In Test |
| BITE | Built-In Test Equipment |
| BIU | Bus Interface Unit |
| bps | Bits Per Second |
| CCITT | Consultative Committee for International Telephone and Telegraph |
| CD | Collision Detection |
| CR | Command Response |
| CRC | Cyclic Redundancy Check |
| CSMA | Carrier Sense Multiple Access |
| CSMA/CD | Carrier Sense Multiple Access/Collision Detection |
| CT | Technical Center (designation used in FAA report numbering scheme) |
| DADC | Digital Air Data Computer |
| DATAC | Digital Autonomous Terminal Access Communication |
| dB | Decibel |
| dBm | Decibels per meter |
| dc | Direct Current |
| DITS | Digital Information Transfer System |
| DM | Delay Modulation |
| DMA | Direct Memory Access |
| DOT | Department of Transportation |
| EMC | Electromagnetic Compatibility |
| EMI | Electromagnetic Interference |
| EMUX | Electrical Multiplex |
| ENRZ | Enhanced Non-return to Zero |
| FAA | Federal Aviation Administration |
| FADEC | Full Authority Digital Engine Controller |
| FAR | Federal Acquisition Regulation |
| FCC | Flight Control Computer |
| FM | Frequency Modulation |
| GaAs | Gallium Arsenide |
| GAMA | General Aviation Manufacturers' Association |
| GCR | Group Code Recording |
| HDLC | High-Level Data Link Control |
| HSRB | High-Speed Ring Bus |

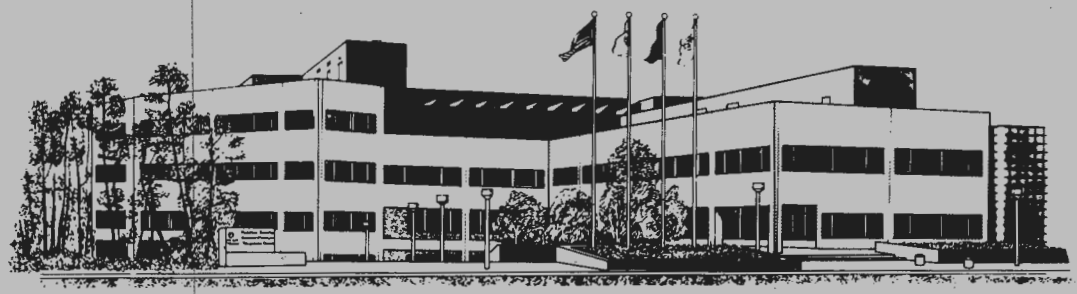
| | |
|---|---|
| IEEE | The Institute for Electrical and Electronics Engineers, Incorporated |
| INS | Inertial Navigation System |
| K | Thousand |
| KHz | Kilohertz (kilocycles per second) |
| km | kilometer |
| LED | Light Emitting Diode |
| LRC | Longitudinal Redundancy Check |
| LRU | Line Replaceable Unit |
| LSB | Least Significant Bit |
| LTPB | Linear Token Passing Bus |
| m | meter |
| $\mu$m | Micrometer |
| M | Million |
| MAADS | Multibus Avionic Architecture Design Study |
| Mbps | Million bits per second |
| MDT | Mean Down Time |
| MFM | Modified-Frequency Modulation |
| MHz | Megahertz |
| MIL-STD | Military Standard |
| ms | Millisecond |
| MSB | Most Significant Bit |
| MTBCF | Mean Time Between Critical Failures |
| MTTR | Mean Time To Repair |
| NA | Numerical Acceptance |
| NADC | Naval Air Development Center |
| NAECON | National Aerospace & Electronics Conference |
| NRZ | Non-return to Zero |
| NRZ-I | Non-return to Zero Inverted |
| NRZ-L | Non-return to Zero Dual Level |
| nsec | Nanosecond |
| PAS | Pilot Assist System |
| PE | Phase Encoding |
| $\phi$M | Phase Modulation |
| RAT | Ring Admittance Timer |
| RNRZ | Randomized Non-return to Zero |
| RZ | Return to Zero |
| SAE | Society of Automotive Engineers |
| str | string |
| TDM | Time Division Multiplex |
| THT | Token-Holding Timer |
| T/R | Transmitter/Receiver |
| VHSIC | Very High-Speed Integrated Circuits |
| VLSI | Very Large Scale Integration |
| VLSIC | Very Large Scale Integrated Circuits |
| VRC | Vertical Redundancy Check |
| ZM | Zero Modulation |

U.S. Department of Transportation
Federal Aviation Administration

DOT/FAA/CT-88/10

# HANDBOOK-VOLUME II
# DIGITAL SYSTEMS VALIDATION

## CHAPTER 7
## ANALYTICAL SENSOR REDUNDANCY



**PREPARED BY:**

**COMPUTER RESOURCE MANAGEMENT, INC.**
**950 HERNDON PARKWAY, SUITE 360**
**HERNDON, VIRGINIA 22070**

**PREPARED FOR:**

**FEDERAL AVIATION ADMINISTRATION**
**TECHNICAL CENTER**
**ATLANTIC CITY INTERNATIONAL AIRPORT, NEW JERSEY 08405**

NOTICE

This document is disseminated under the sponsorship of the U.S. Department of Transportation in the interest of information exchange. The United States Government assumes no liability for the contents or use thereof.

The United States Government does not endorse products or manufacturers. Trade or manufacturers' names appear herein solely because they are considered essential to the objective of this report.

TABLE OF CONTENTS


| Section | | Page |
|---|---|---|
| 1. | INTRODUCTION | 7-1 |
| | 1.1. Control Law Modification | 7-2 |
| | 1.2. Fault Tolerant Design | 7-2 |
| | 1.3. Scope | 7-4 |
| 2. | OVERVIEW | 7-5 |
| | 2.1. Analytical Redundancy Building Blocks | 7-5 |
| 3. | ANALYTICAL REDUNDANCY - DESIGN APPROACH | 7-11 |
| | 3.1. Observer/Signal Blender Filter | 7-16 |
| | 3.2. Diagnostic Kalman Filters | 7-16 |
| | 3.3. Super-Diagnostic Kalman Filter Design | 7-16 |
| | 3.3.1. Multiple Hypothesis Tests | 7-17 |
| | 3.3.2. Parameter Identification | 7-17 |
| | 3.3.3. Generalized Likelihood Ratio Methods | 7-17 |
| | 3.3.4. Modified Filter Design | 7-18 |
| | 3.3.5. Jump Process Estimation | 7-18 |
| | 3.4. Redundancy Management | 7-18 |
| 4. | ANALYTICAL REDUNDANCY - RELIABILITY | 7-21 |
| 5. | ANALYTICAL REDUNDANCY - FAULT DETECTION AND ISOLATION | 7-23 |
| | 5.1. Multiple Trip Monitor | 7-23 |
| | 5.2. Sequential Likelihood Ratio Tests | 7-25 |
| | 5.3. Fault Modeling | 7-27 |
| 6. | SUPPLEMENTAL WORKED EXAMPLES | 7-29 |
| | 6.1. Altitude Kinematics | 7-29 |
| | 6.2. Translational Kinematics | 7-32 |
| | 6.3. Stability Augmentation System | 7-33 |
| BIBLIOGRAPHY | | 7-37 |
| GLOSSARY | | 7-40 |
| ACRONYMS | | 7-42 |

## LIST OF ILLUSTRATIONS


| Figure | | Page |
|---|---|---|
| 2-2 | ROTATIONAL RELATIONSHIPS USED FOR ANALYTICAL REDUNDANCY | 7-7 |
| 2.1-1 | ANALYTICAL REDUNDANCY CONCEPT | 7-8 |
| 2.1-2 | ANALYTICAL SENSOR REDUNDANCY FAULT DETECTION LOGIC | 7-9 |
| 3-1 | EXAMPLES OF FAULT DETECTION USING DIAGNOSTIC FILTERS AND INTERCHANGEABLE SENSORS | 7-12 |
| 3-2 | EXAMPLES OF FAULT DETECTION USING DIAGNOSTIC FILTERS AND NONINTERCHANGEABLE SENSORS | 7-13 |
| 3-3 | EXAMPLE OF FAULT DETECTION WITH REVERSION MODES | 7-15 |
| 5.1-1 | MULTIPLE TRIP FAULT DETECTION MONITOR | 7-24 |
| 5.1-2 | SEQUENTIAL LIKELIHOOD FAULT DETECTION MONITOR | 7-27 |
| 6.1-1 | THIRD ORDER VERTICAL COMPLEMENTARY FILTER FOR ALTITUDE KINEMATICS | 7-30 |
| 6.1-2 | POSSIBLE REDUNDANCY MANAGEMENT FOR ALTITUDE KINEMATICS FAULT ISOLATION | 7-31 |
| 6.2-1 | EXAMPLE FILTER USED FOR TRANSLATIONAL KINEMATICS REDUNDANCY | 7-34 |
| 6.3-1 | EXAMPLE FILTERS USED FOR STABILITY AUGMENTATION SYSTEM | 7-36 |


## LIST OF TABLES


| Table | | Page |
|---|---|---|
| 3-1 | SENSOR REDUCTION COMPARISONS FOR ANALYTICAL REDUNDANCY BUILDING BLOCKS | 7-14 |
| 4-1 | SYSTEM RELIABILITY | 7-22 |

# 1. INTRODUCTION

The advent of modern digital controls has greatly changed the control technology and hardware implementation of aircraft flight control. These Digital Flight Control Systems (DFCS) provide enhanced performance for stability and control augmentation over an extremely wide range of flight envelope parameters. They also offer improved capabilities to operate in differing aircraft configurations from takeoff with wheels and flaps down to a high speed clean configuration with low drag, thrust, and low g forces. In addition to the performance enhancements in Digital Flight Control (DFC), the DFC computer has the capability to perform other computations to effect improvements in reliability and fault tolerance. Elaboration of these later features is the major thrust of this portion of the handbook.

The DFC improves reliability due to two factors. The first is the increased reliability of digital computer hardware relative to older technology; this enhancement leads to tens of thousands of hours between failures. The second factor is the increased reliability of the sensors and actuators which leads to several hundreds of hours between failures. The principal factor affecting DFC system reliability is reliability of the sensors and actuators.

The overall reliability required in today's flight control systems dictates the need for $10^{-9}$ probability of failure in cases where stability augmentation is needed for safe flight. This requires an increase in the reliability of the DFC components. The reliability of DFC computers may be achieved by utilizing dual processors, error detection, and fault tolerant designs. An increase in reliability of sensors and actuators may be achieved by means of redundant elements (dual or triple).

Assessment of this situation leads to a conclusion that redundant actuators and linkages are an adequate solution for actuation, and that DFC computers are adequate. The term Analytical Redundancy (AR) refers to the scheme for increasing the reliability of sensors by using the DFC computer for Fault Detection (FD) and isolation and for analytically determining estimates of sensor values.

Reliable DFC systems are burdened with large numbers of sensors which are required for both quality and redundancy. The demand for control quality has expanded the types of sensors used in basic augmentation functions. The yaw damper currently requires sensor data concerning attitude, angle of attack, and airspeed. The pitch damper requires a complex multifunction control law along with data from several sensors to drive multiple surfaces. The complexity of the sensors has improved quality; however, designs that meet overall flight control system reliability requirements lead to triple and quadruple replication of system components.

High levels of fault tolerance are mandatory for flight-critical control systems. Current Fly-By-Wire (FBW) systems utilize redundant computers,

actuators, and sensors to achieve the necessary fault tolerance. Fault isolation (FI) is determined by some type of comparison or voting between like sensors. A one out of two sensor failure cannot be isolated by comparison techniques. In order to improve the reliability of a redundant set of sensors, one or more of the following must be implemented: increase the reliability of the existing sensors; add additional like sensors; synthesize the needed information; or isolate the remaining sensor if a failure occurs at the duplex level.

A large amount of sensor hardware is costly from a system standpoint. Sensors may act as antennas to increase electromagnetic susceptibility. The overall aircraft design may be constrained due to a large number of sensors. Therefore, there is a strong motivation to reduce the number of sensors. Significant reductions appear to be possible through a variety of methods. Recognized techniques for sensor reduction are control law modification and fault tolerant design. These methods may be used in combinations to optimize the number of sensors necessary for a particular application.

## 1.1. Control Law Modification

Control law modification ensures that operational requirements can be met with the minimum number of sensors and a better control system design. The control laws are modified to optimize the use of sensed data utilizing the analytical relationships which exist between sensors. The flight controller uses fewer sensed values in the basic single channel control laws, yet maintains the same performance as the controller which utilized all the sensor data. These modifications are application dependent.

Signal synthesis is one example of control law modification. The number of sensors required is reduced by synthesizing a signal or group of signals from a reduced set. The elements used to perform the synthesis are Kalman filters or Luenberger observers. Reduction in the number of sensors is a trade-off against increased complexity of the control laws; this is a critical factor for analog systems. The trade-off in digital systems occurs in system performance. The bandwidth and noise content of the synthesized signal must be equivalent to the sensor eliminated. These techniques must be applied with care because of potential physical limitations which cannot be practically overcome.

Integrated flight management is another example of control law modification. This method works on the principle that subsystems which use common sensor types are combined. An example is the combination of the navigator and the flight control. The signal quality derived from the navigator is high and the data could easily be used by the flight control. Issues here are cost (since the navigator is normally not flight critical) and multiple failures.

## 1.2. Fault Tolerant Design

The following fault tolerant design techniques exist to meet reliability requirements using a minimum number of sensors. These techniques use fewer sensors to achieve the same level of redundancy as a quad-redundant system.

- Skewed and special sensors.

- Sensor integration.
- In-line sensor monitoring.
- Analytical sensor redundancy.

A skewed sensor arrangement significantly reduces the number of sensors required for Redundancy Management (RM). Skewed sensors do not have to be placed on orthogonal axes. The x, y, and z components of the sensed quantity can be determined using an appropriate coordinate transformation matrix. If one of the skewed sensors fails, the three orthogonal components can still be determined with another sensor and its transformation matrix.

A quad-redundant dual fail-operative system with orthogonal gyros in three axes requires twelve gyros. The same properly configured system with skewed gyros requires only six to maintain the same redundancy requirements. Skewing accelerometers can similarly reduce the number of sensors from twelve to six. Potential limitations are differences in scale and resolution requirements for each axis. For the skewed configuration, each gyro must have the same specifications. Depending upon the gyro chosen, either the signal quality may be limited or the instrumentation cost may be greater. Weinstein (1978) and Kaniuka, et al., 1982 show adequate performance for this technique.

Special sensors measure linear combinations of variables. The measurements are skewed in measurement space rather than geometrically, i.e., the sensor measures both acceleration and body rate. A conventional quad-redundant system could be replaced by five special sensors. These sensors are typically unique to the application and have a higher cost. A multipurpose air data sensor which measured pitot pressure, static pressure, angle of attack, and angle of sideslip was implemented in the system described by Weinstein (1978) and Kaniuka, et al., 1982.

Subsystem sensor integration can also be used to reduce the number of sensors. This technique uses data from subsystems which are not functionally related for monitoring and tie-breaking. An example is on the space shuttle where derived rates from the navigation system are used in the primary flight control system for voting.

Fault tolerant design can be achieved through in-line monitoring. The state-of-the-art for servos and computers is near 99 percent self-test confidence. In-line self-test feasibility for sensors is limited for several reasons. The sensor input is unknown except for special test signals. There is no way to test for sensor installation errors. Any sensor self-test adds considerable complexity which may be undesirable. Typically, a self-test can only detect very large faults, particularly in a dynamic environment.

Analytical Redundancy provides a means for substituting software algorithms for hardware sensors. Sensors are eliminated by analysis using known relationships between sensed values. Various simulation studies have shown the effectiveness of utilizing AR to increase system reliability and to reduce the amount of sensor hardware.

## 1.3. Scope

This document details the uses and limitations of analytical sensor redundancy. Section 2 gives an overview of AR. The physical relationships which can be utilized are written in equation form. The basic concept and building blocks are discussed. Section 3 describes various algorithms and design tools used to implement the filters used in AR processing. Its feasibility and performance are discussed in section 4. Algorithms and methods used for FD and isolation are detailed in section 5. Examples of the filters, RM, and sensor relationships for three cases where analytical sensor redundancy is easily implemented are given in section 6.

# 2. OVERVIEW

The basic idea of AR is to use known geometric and dynamic relationships between different sensors in order to detect and isolate sensor failures. These relationships can be defined in translational and rotational equations as shown in figures 2-1 and 2-2. Figure 2-2 also shows a diagram of the Euler angle relationships.

Cunningham and Hartmann (1977) describe the application of these mathematical relationships to eleven different measured quantities, normal acceleration, lateral acceleration, angle of attack, true airspeed, altitude, P, Q, R, $\emptyset$, $\theta$, and $\psi$.

Four categories of AR were utilized by Deckert, et al., 1978. Translational kinematics redundancy exists between the integrated accelerometer outputs, vertical and rate gyros, and the air data sensor outputs. Translational dynamics redundancy uses the aerodynamic forces on the aircraft measured by the accelerometers and stored coefficients to relate to aerodynamic forces calculated from air-data sensor data. Rotational kinematics redundancy relates the integrated outputs of the rate gyros and the vertical and directional gyro outputs. Altitude Kinematics (AK) redundancy is defined as the relationship between the altitude measured by the altimeter and the computed altitude from a double integration of the accelerometer and vertical gyro output data. These relationships are defined in the equations in figures 2-1 and 2-2. Examples of altitude and translational kinematics redundancy processing are given in sections 6.1 and 6.2.

## 2.1. Analytical Redundancy Building Blocks

Many specific approaches to AR have been developed, ranging from simple signal blenders to complex combinations of Kalman filters.

In general, AR is approached using one of two basic building blocks. A Diagnostic Filter (DF) is an algorithm which processes data from a family of N functionally related sensors in order to estimate signals and to assess the functioning of the sensors. It outputs signal estimates and an error flag which indicates that all sensors are functioning properly or that one of them has failed. A Super-Diagnostic Filter (SDF) performs the functions of a DF and can additionally isolate the specific faulted sensor. One can use assemblies of DFs to construct an SDF through truth table logic.

The concept of AR and associated FD are shown in figures 2.1-1 and 2.1-2.

Translational Equations

Inertial Velocity

$$\dot{X}_e = \cos\theta \cos\tau\, U + (\sin\phi \sin\theta \cos\tau - \cos\phi \sin\tau)V + (\cos\phi\sin\theta \cos\tau + \sin\phi \sin\tau)W$$

$$\dot{Y}_e = \cos\theta \sin\tau\, U + (\sin\phi \sin\theta \sin\tau + \cos\phi \cos\tau)V + (\cos\phi \sin\theta \cos\tau - \sin\phi \cos\tau)W$$

$$\dot{Z}_e = -\sin\theta\, U + \sin\phi \cos\theta\, V + \cos\phi \cos\theta\, W$$

Acceleration

$$\dot{U} = A_x - g \sin\theta - QW + RV$$

$$\dot{V} = A_y + g \cos\theta \sin\phi - RU + PW$$

$$\dot{W} = A_z + g \cos\theta \cos\phi - PV + QU$$

where $X_e$, $Y_e$, $Z_e$ = earth referenced forward, lateral, and vertical positions
U, V, W = forward, lateral, and vertical body velocities, respectively
$A_x$, $A_y$, $A_z$ = forward, lateral, and vertical body accelerations
g = acceleration due to gravity
$\theta$, $\phi$, $\tau$ = Euler pitch, roll, and yaw angles
P, Q, R = body roll, pitch, and yaw rates

FIGURE 2-1. TRANSLATIONAL RELATIONSHIPS USED IN ANALYTICAL REDUNDANCY

**Rotational Equations**

Euler Rates

$$\dot{\phi} = P + (Q \sin\phi + R \cos\phi) \tan\phi$$

$$\dot{\theta} = Q \cos\phi - R \sin\phi$$

$$\dot{\tau} = (Q \sin\phi + R \cos\phi) \sec\theta$$

Body Rates

$$P = \dot{\phi} - \dot{\tau} \sin\theta$$

$$Q = \dot{\theta} \cos\phi - \dot{\tau} \sin\phi \cos\theta$$

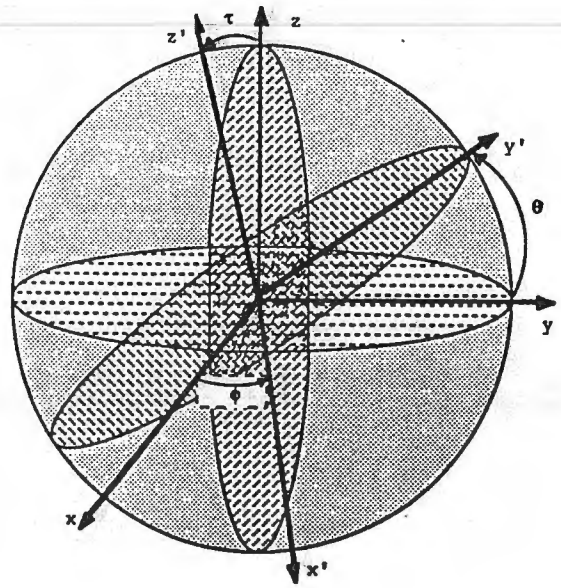
$$R = \dot{\tau} \cos\phi \cos\theta - \dot{\theta} \sin\phi$$

Moments

$$L = \dot{P} I_{xx} - \dot{R} I_{xz} + QR (I_{zz} - I_{yy}) - PQ I_{xz}$$

$$M = \dot{Q} I_{yy} + PR (I_{xx} - I_{zz}) + (P^2 - R^2) I_{xz}$$

$$N = \dot{R} I_{zz} - \dot{P} I_{xz} + PQ (I_{yy} - I_{xx}) + QR I_{xz}$$




EULER ANGLE DIAGRAM

where $\theta$, $\phi$, $\tau$ = Euler pitch, roll, and yaw angles
P, Q, R = body roll, pitch, and yaw rates
L, M, N = moment about the body x, y, and z axes
$I_{xx}$, $I_{yy}$, $I_{zz}$ = moment of inertia about the x, y, and z axes
$I_{xz}$ = cross moment of inertia (xz axis)

FIGURE 2-2. ROTATIONAL RELATIONSHIPS USED FOR ANALYTICAL REDUNDANCY

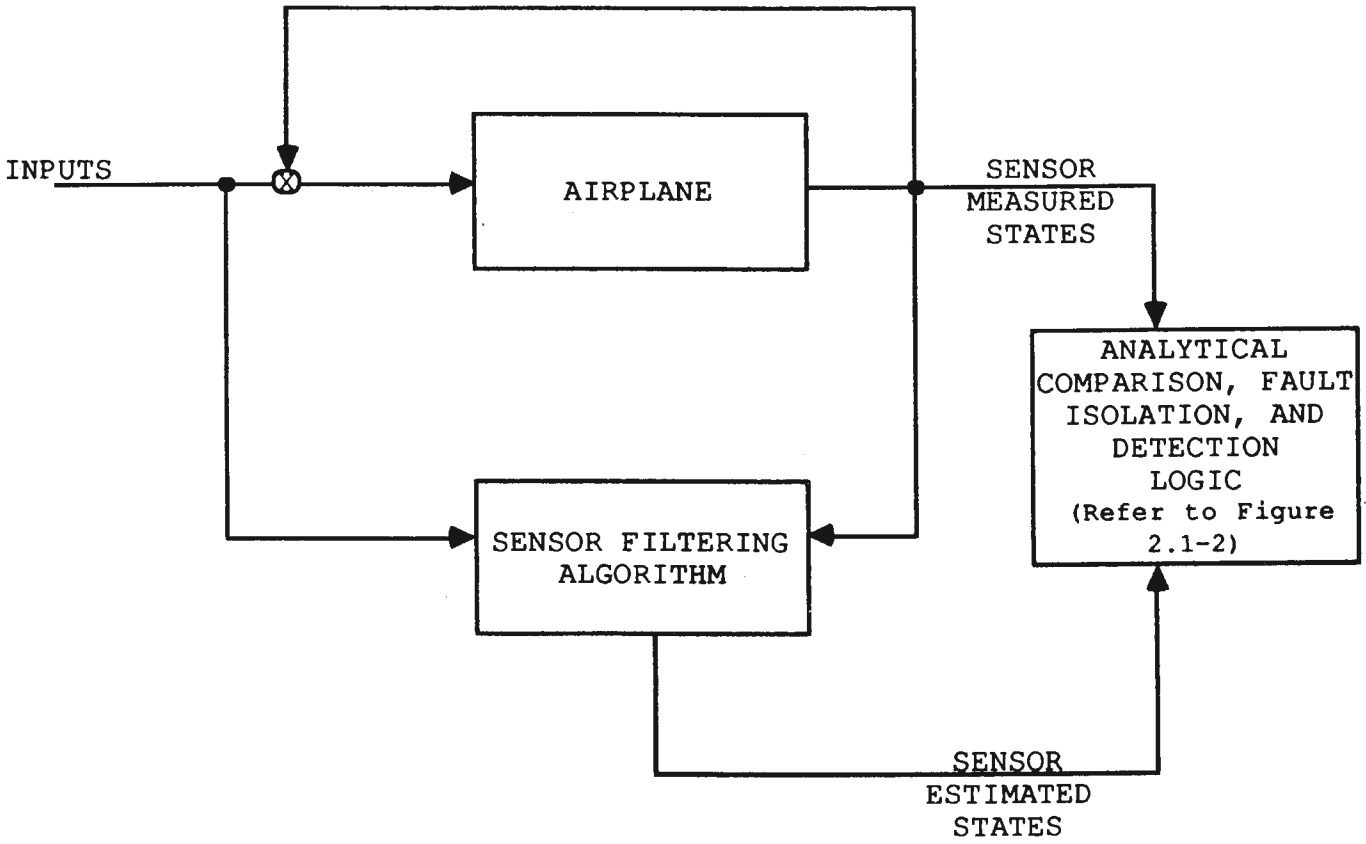



FIGURE 2.1-1. ANALYTICAL REDUNDANCY CONCEPT

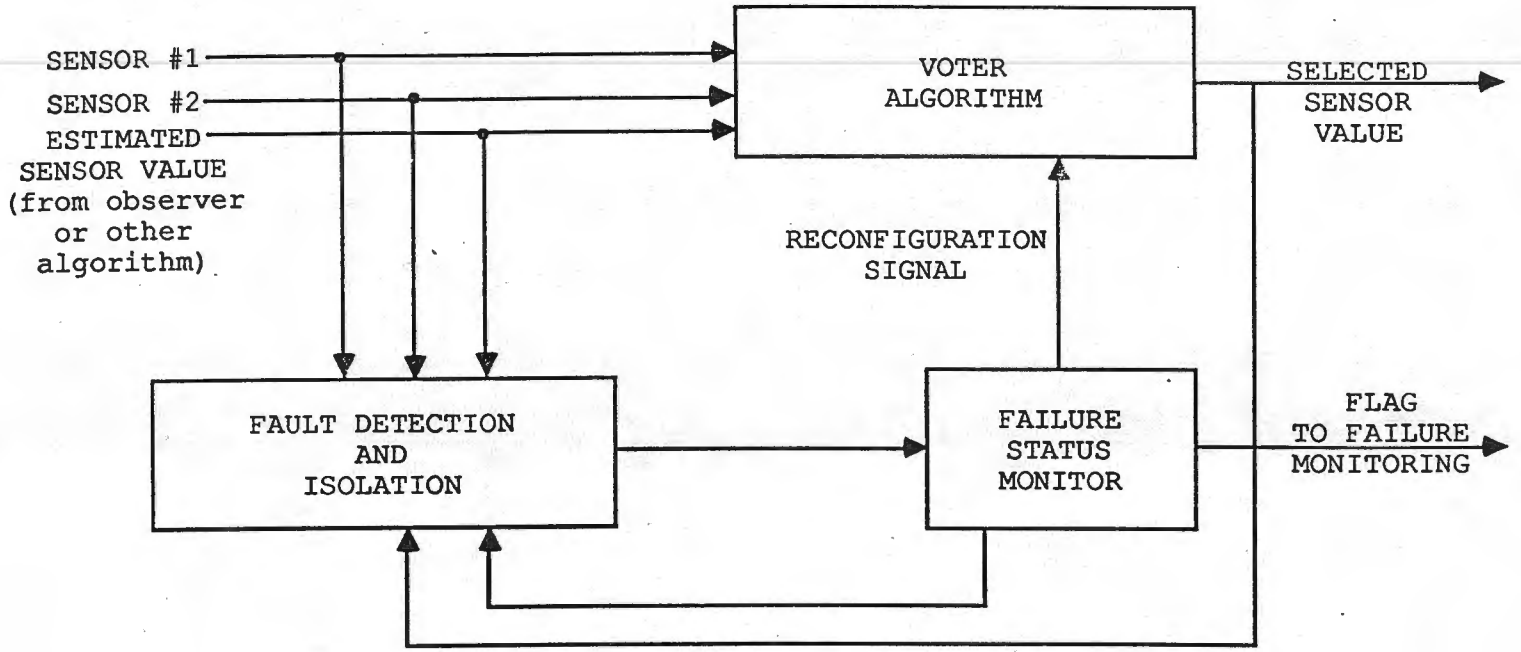



FIGURE 2.1-2. ANALYTICAL SENSOR REDUNDANCY FAULT DETECTION LOGIC

## 3. ANALYTICAL REDUNDANCY - DESIGN APPROACH

A literature survey was performed by Cunningham and Hartmann (1977) to examine currently available techniques that were able to detect and isolate faults in aircraft sensors and meet current onboard flight computer allocations. All such methods for failure detection were found to lie in three categories.

- Assemblies of DFs (see Meier, Ross, and Glaser, 1971; Maybeck, 1974; Hartmann, et al., 1975; and Clark, et al., 1975).
- Specific DF design techniques (see Kerr, 1974 and Mehra and Peschon, 1971).
- Explicit SDF design techniques.

The first two categories also include observer/blender techniques. Mulcare, Downing, and Smith (1987) and Deckert, et al., 1978 utilize these techniques.

The three filter types listed above can be combined in many ways to obtain useable AR results. The end product may vary considerably in accuracy and complexity. The structure of the filter depends on the overall RM approach, whether uniform redundancy or different levels of redundancy are required throughout the system.

Uniform redundancy schemes categorize sensors as interchangeable or noninterchangeable. Interchangeable sensors can substitute for each other when a failure occurs. Appropriate coordinate transformations must be accounted for, if necessary. If a noninterchangeable sensor fails, an identical one must replace it in order to maintain performance.

If N fully operational sensors are required to maintain aircraft operations, then a minimum of N+2 interchangeable sensors and 3N noninterchangeable sensors are required for dual fail-operational performance when using AR. In order to resolve conflicts using traditional voting techniques, N+3 and 4N sensors are needed to maintain the same performance.

Fault Detection using DFs with interchangeable sensors is illustrated in figure 3-1. The figure shows FD for N, the number of sensors required for safe flight, equal to 2 for fail-operational performance. The truth table shows that each column of the error flags is a unique binary word and that the sensor inputs to each filter are uniquely determined by each row of the table. The generalized way to assemble DFs for interchangeable sensors is given below.

- Determine the number of sensors required to maintain safe flight and meet operational requirements (N is the number of sensors necessary for safe flight).

N + 1 for fail-operational performance
N + 2 for dual fail-operational performance

- Determine the number of DFs (the brackets indicate to use the next largest integer of the enclosed expression).

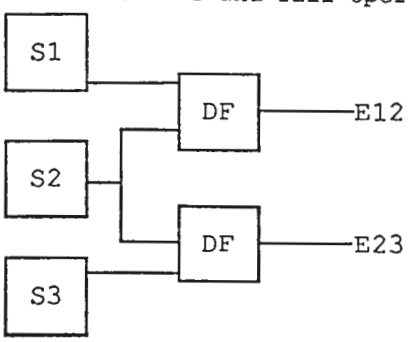
$[\log_2 (N + 2)]$ for fail-operational performance
$[\log_2 (N + 3)]$ for dual fail-operational performance

- Determine sensor inputs to each filter from the truth table.

The FD capability of a DF combined with comparators for noninterchangeable sensors can also be verified with a truth table. See figure 3-2 for an illustration of fail-operational performance with N, the number of sensors necessary to maintain safe flight, equal to 2. The generalized arrangement uses one filter and N comparators for fail-operational and one filter and 2N comparators for dual fail-operational. The truth table here contains FD for single sensor failures only. It is generally assumed that failures occur singly when designing FD algorithms. A single SDF is all that is needed in all cases to provide FD and isolation. This type of filter may be quite complex in order to achieve the same results as DFs.

INTERCHANGEABLE SENSORS

for N=2 and fail-operational performance



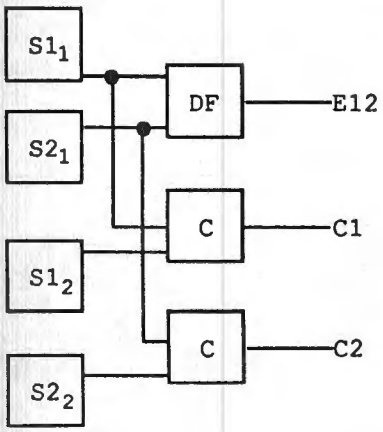

TRUTH TABLE

| ERROR FLAGS | SINGLE SENSOR FAILURE | | | |
|---|---|---|---|---|
| | NONE | S1 | S2 | S3 |
| E12 | 0 | 1 | 1 | 0 |
| E23 | 0 | 0 | 1 | 1 |

FIGURE 3-1. EXAMPLES OF FAULT DETECTION USING DIAGNOSTIC FILTERS AND INTERCHANGEABLE SENSORS

NONINTERCHANGEABLE SENSORS

for N=2 and fail-operational performance




TRUTH TABLE

| ERROR FLAGS | SINGLE SENSOR FAILURE | | | | |
|---|---|---|---|---|---|
| | NONE | $S1_1$ | $S2_1$ | $S1_2$ | $S2_2$ |
| E12 | 0 | 1 | 1 | 0 | 0 |
| C1 | 0 | 1 | 0 | 1 | 0 |
| C2 | 0 | 0 | 1 | 0 | 1 |

C = Comparator
DF = Diagnostic Filter
N = # of sensors needed to maintain safe flight

FIGURE 3-2. EXAMPLES OF FAULT DETECTION USING DIAGNOSTIC FILTERS AND NONINTERCHANGEABLE SENSORS

The largest amount of sensor reduction is achieved for noninterchangeable sensors when using AR. In this case, AR implemented with one DF in combination with 2N comparators has the same benefit as one SDF. An SDF which is more complex will not be feasible. A comparison of the sensor reductions possible with these analytical techniques and traditional voting is given in table 3-1. The number of comparators and DFs needed for AR are also given. N is defined as the minimum number of sensors required to maintain safe flight.

There is a sensor reduction benefit for interchangeable sensors; however, it is a small one. AR provides little or no benefit to save one sensor. More DFs are needed to match the capabilities of a single SDF than for noninterchangeable sensors.

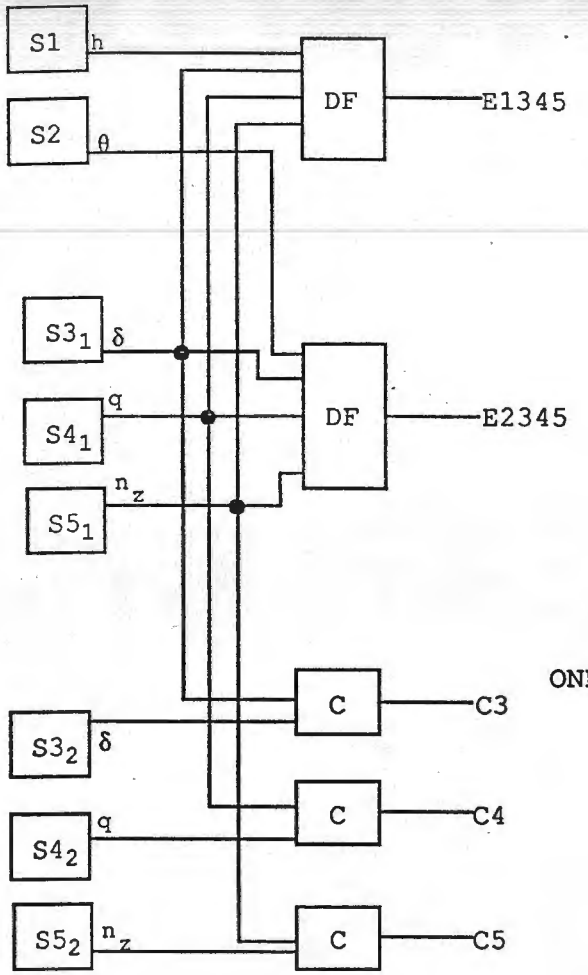
Current flight control systems normally do not have the same redundancy requirements for all sensors. A more realistic setup may have high levels of redundancy for some critical sensors and lower levels for others. The high levels of redundancy are known as reversion modes. While this may be application dependent, the basic principles can be used to obtain a failure detection scheme as shown in figure 3-3. In order to implement the reversion modes, sufficient filters are required to distinguish between four outer loop logic conditions: no failure; S1 failed; S2 failed; and $S3_1$, $S4_1$, or $S5_1$ failed. The error flags, E1345 and E2345, are used to obtain this information. The error flags, C3, C4, and C5, must be used to determine which of the inner loop sensors have failed.

TABLE 3-1. SENSOR REDUCTION COMPARISONS FOR ANALYTICAL REDUNDANCY BUILDING BLOCKS

| **FAIL-OPERATIONAL** | NUMBER OF SDF'S | NUMBER OF DIAGNOSTIC FILTERS | NUMBER OF COMPARATORS | NUMBER OF SENSORS |
|---|---|---|---|---|
| **INTERCHANGEABLE SENSORS** | | | | |
| DETECTION WITH SDF'S | 1 | 0 | 0 | N+1 |
| WITH DF'S | 0 | $[\text{LOG}_2(N+2)]$ | 0 | N+1 |
| WITH VOTING | 0 | 0 | N+3 | N+2 |
| **NONINTERCHANGEABLE SENSORS** | | | | |
| DETECTION WITH SDF'S | 1 | 0 | 0 | 2N |
| WITH DF'S | 0 | 1 | N | 2N |
| WITH VOTING | 0 | 0 | 2N | 3N |
| **DUAL FAIL-OPERATIONAL** | NUMBER OF SDF'S | NUMBER OF DIAGNOSTIC FILTERS | NUMBER OF COMPARATORS | NUMBER OF SENSORS |
| **INTERCHANGEABLE SENSORS** | | | | |
| DETECTION WITH SDF'S | 1 | 0 | 0 | N+2 |
| WITH DF'S | 0 | $[\text{LOG}_2(N+3)]$ | 0 | N+2 |
| WITH VOTING | 0 | 0 | N+4 | N+3 |
| **NONINTERCHANGEABLE SENSORS** | | | | |
| DETECTION WITH SDF'S | 1 | 0 | 0 | 3N |
| WITH DF'S | 0 | 1 | 2N | 3N |
| WITH VOTING | 0 | 0 | 3N | 4N |

N = MINIMUM NUMBER OF SENSORS REQUIRED FOR SAFE FLIGHT

[ ] IMPLIES USE THE NEXT LARGEST INTEGER OF THE EXPRESSION WITHIN THE BRACKETS




TRUTH TABLE

| ERROR FLAGS | SENSORS FAILED | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | NONE | S1 | S2 | $S3_1$ | $S4_1$ | $S5_1$ | $S3_2$ | $S4_2$ | $S5_2$ |
| E1345 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| E2345 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| C3 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| C4 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| C5 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |

TWO FAIL-SAFE OUTER LOOPS (h), (θ)

ONE FAIL-OPERATIONAL INNER LOOP ($\delta, q, n_z$)
(NON-INTERCHANGEABLE SENSORS)

- h = altitude
- θ = Euler pitch angle
- δ = control surface deflection
- q = body pitch rate
- $n_z$ = normal acceleration
- C = Comparator
- DF = Diagnostic Filter

FIGURE 3-3. EXAMPLE OF FAULT DETECTION WITH REVERSION MODES

Three concepts encompass the categories of failure detection methods. These are examined as a result of various combinations of DFs. These concepts in order of increasing complexity are:

- Observer/signal blender filter.
- Diagnostic Kalman filters.
- Super-diagnostic Kalman filter design.

## 3.1. Observer/Signal Blender Filter

An observer/blender will model physical relationships using sensor outputs to provide FD for several sensors. It uses a specific DF design and has the least amount of complexity and computer requirements. Monitors may use a three trip level exceedance criterion to detect failures. Trip thresholds can be set to account for unmodeled dynamics and sensor anomalies.

The basic operation of the observer/blender is to define a consistent error function between a measured parameter and its derivative. Both values are measured for all but a few parameters, e.g., lateral acceleration. A relationship such as,

$$V_x = \dot{X}$$

is exact. For perfect measurements, there is no error. Realistically, the measured values of $V_x$ and X are not true representations of $V_x$ and X. There is an error associated with digital integration and sampling. Either $V_x$ or X may contain faults. This implementation of AR must distinguish between the errors associated with measurements and digital sampling and errors associated with the sensors.

## 3.2. Diagnostic Kalman Filters

Diagnostic Kalman filters incorporate modeling of sensor anomalies (such as, bias, scale factors, and wind gust estimation) in the filter. FD is provided for more sensors than available with the observer/blender. This is achieved by using an assembly of DFs. This concept places moderate requirements on a computer.

## 3.3. Super-Diagnostic Kalman Filter Design

The super-diagnostic design utilizes Kalman filters as well as linear equations for pitch and lateral directional dynamic equations of motion. Both detection and isolation can be determined. An error signal is created for each sensor involved. The concept is built with an assembly of DFs. Some applications may be very complex.

The largest number of surveyed references are devoted to this topic. These filters are the most difficult to construct. They are constrained more than other techniques due to the amount of complexity which is feasible. Five methods have been examined for SDFs.

- Multiple hypothesis test.
- Parameter identification.
- Generalized likelihood ratio methods.
- Modified filter designs.
- Jump process estimation.

### 3.3.1. Multiple Hypothesis Tests

Each failure condition of a particular sensor is used to define a hypothesis. The entire range of failure conditions forms a set of hypotheses which are tested against observed data. One Kalman filter is required to test each hypothesis. The filter generates residuals, assuming that the hypothesis is true, which are summed into likelihood functions. The maximum likelihood then identifies the current valid hypothesis. This results in a large number of filters for even a few sensors and at present is too complex for advanced flight computers to utilize. Montgomery and Caglayan (1974), Montgomery and Price (1974), Athans and Willner (1973), Lainiotis (1971), and Buxbaum and Haddad (1969) contain more information.

### 3.3.2. Parameter Identification

Parameters of each sensor, such as, gain and bias, are defined as unknowns and estimated from sensor data. If the estimates deviate a great deal from nominal values, then a failure is declared. For any procedure used, a separate calculation similar to a Kalman filter is required to compare the nominal values with the estimated sensor parameters. This becomes as complex and inefficient as the previous technique. Mehra and Peschon (1971) and Stein and Hartmann (1976) have investigated this technique.

### 3.3.3. Generalized Likelihood Ratio Methods

In an attempt to reduce the complexity of this type of failure detection method, initial constraints were placed on the design of an algorithm. These constraints are that only one Kalman filter is available and that sensor failure detection must be decided through the monitoring of residuals from the single filter. This can be done for those failures which have recognizable signatures for correlation processing. The residuals are correlated with known signals, normalized by their expected no-failure value, and compared to predetermined threshold levels. The number of filters has been greatly reduced, however, the correlation functions can be fairly complex. See Willsky, Deyst, and Crawford (1974); Willsky and Jones (1974), Deyst and Deckert (1975), McAulay and Denlinger (1973), and Sanyal and Shen (1974).

### 3.3.4. Modified Filter Design

This method starts with a constraint of one Kalman filter. It does not allow the filter to remain a fixed element. An attempt is made to alter the filter gains to produce strong and easily recognized failure signatures. This concept is largely theoretical at this time. Details are given in Jones (1973) and Beard (1973).

### 3.3.5. Jump Process Estimation

This procedure is also theoretical. The basic idea is to represent failures as random jump processes in an otherwise known stochastic system. The time of occurrence and magnitude of the jumps are then estimated using optimal stochastic filtering theory. (See Sworder, 1972; Sworder and Robinson, 1973; Ratner and Luenberger, 1969; Pierce and Sworder, 1971; Davis, 1975; McGarty, 1974; and Chien, 1972.)

## 3.4. Redundancy Management

Redundancy Management will be optimized when included in a system design, but it can be incorporated into an existing system. Mulcare, Downing, and Smith (1987) modified an existing quadruplex DFCS to incorporate AR. Section 6.3 contains example filters which could be used to add AR to one of the functions, a Stability Augmentation System (SAS), implemented in this modification.

Issues concerned when designing and structuring the RM are:

- The number of success paths.
- The amount of dependency placed on monitoring quality.
- The tradeoff between fail safety and reliability.
- The cost effectiveness for increased reliability with added FD and isolation.
- The throughput constraints imposed by the flight computer. RM can be 60 to 80 percent of the required processing.

Performance qualities may vary for different RM structuring.

In order to maintain fail-safe sensor operations, the following three types of logic are needed.

- Comparison monitoring of dual like signals without an associated DF. Examples where this may be implemented are after a computer failure or a failure of an input to a DF.
- Monitoring the diagnostic error function after failure of one-out-of-two sensors in a dual set. If a pitch rate gyro fails, the relevant DF then monitors the remaining pitch rate gyro.

- Monitoring of a single sensor by its associated DF. This logic requires a failure declaration from both computers to avoid fault indication due to a dual sensor failure.

## 4. ANALYTICAL REDUNDANCY - RELIABILITY

The concept of AR, in order to be implemented in practice, must satisfy performance and reliability requirements of typical flights and must not overload the flight computer.

The traditional RM for a quadruplex set of sensors would isolate the first and second failed sensors by a majority vote. On the failure of the third sensor the set would be shut down. For a perfect monitoring process, the probability of total failure of the set is $4Q_s^3$, where $Q_s$ is the probability of failure of one of the sensors within a certain time. The probability of any two sensors failing is $6Q_s^2$.

A rate gyro has a failure rate of about $10^{-4}$ per hour. This is one of the worst sensor failure rates. The probability of a quadruplex set of gyros failing would be $4 \times 10^{-12}$. This exceeds the typical flight control system requirements per sensor set of $10^{-9}$ by more than two orders of magnitude. For a triplex set of gyros, failure is defined as $3Q_s^2$ with a value of $3 \times 10^{-8}$ which is less than the requirements by a factor of thirty.

The assumption of perfect monitoring is not valid. An additional factor, $Q_m$, should be included in the failure rate. $Q_m$ defines the probability that the first failure is not detected and that both failures are alike. This issue of imperfect monitoring or coverage may cause failure of a quadruplex sensor set after only two failures, if it cannot distinguish between the two good and two failed sensors. The probability of failure here is $6Q_s^2\ Q_m$. No data exists for the term $Q_m$, but numbers on the order of $10^{-2}$ have been suggested. With imperfect monitoring, the probability of failure for a quadruplex set of rate gyros is $6 \times 10^{-10}$ in one hour which is just within the requirements.

The diagnostics provided in an AR system must provide similar quality in monitoring. For triplex sensors configured for dual-fail-operational systems, the total failure rate is given in the following equation.

$$Q_s^3 + 3Q_s^2\ Q_d$$

The first term represents the probability for all three sensors failing. The second term is the combination of the probabilities of a dual sensor failure and the diagnostics failing to recognize the fault, $Q_d$. This is the dominant term. If it is set equal to the reliability requirement,

$$3Q_s^2\ Q_d = 10^{-9}$$

$$Q_d = 3.3 \times 10^{-2}$$

The value of $Q_d$ indicates that the diagnostics must be 97 percent reliable. If this performance is feasible, then AR techniques are at least equal to those of quadruplex channel voting.

This example was devoted to just one set of sensors. Total system reliability includes all hardware: computers, servos, and sensors. A computer code, Computer Aided Redundant System Reliability Analysis (CARSRA), computes system reliabilities based on a specific configuration and sensor failure rates. Mulcare, Downing, and Smith (1987) used a revised version of this code to perform a system reliability assessment.

Table 4-1, utilizing parameters of an existing system (Cunningham and Hartmann, 1977), shows that servos, not sensors, tend to dominate reliability. By adding sensor redundancy, only a factor of two less than baseline system failure rate is obtained. With both sensor and servo redundancy, system failure rate decreased by more than an order of magnitude. Failure rates are dependent on system architecture. A system should be designed to have matched reliability.

TABLE 4-1. SYSTEM RELIABILITY

| Probability (failures per flight hour) | | |
|---|---|---|
| Failure Type | 95% Sensor Redundancy | 95% Sensor/Servo Redundancy |
| Either computer fails (95% effective self-test) | $0.2 \times 10^{-4}$ | $0.2 \times 10^{-4}$ |
| Servo failure in any axis | $6.0 \times 10^{-4}$ | $0.3 \times 10^{-4}$ |
| Gyro failure in any axis | $0.3 \times 10^{-4}$ | $0.3 \times 10^{-4}$ |
| Normal accelerometer failure | $0.02 \times 10^{-4}$ | $0.02 \times 10^{-4}$ |
| Total | $6.52 \times 10^{-4}$ | $0.82 \times 10^{-4}$ |

# 5. ANALYTICAL REDUNDANCY - FAULT DETECTION AND ISOLATION

The type of FD monitor used to test various error signals is critical for performance. Important parameters are speed of response and an acceptable false alarm rate, one per 1000 hours of flight. Observers or DFs can operate with comparison monitors of dual sensors to isolate faults for a fail-operational capability. A fail-safe capability exists for all single sensors and for dual sensors after one failure.

Fault Detection monitors can be multiple trip monitors or a Sequential Likelihood Ratio Test (SLRT) of a defined parameter, error signals, or likelihood functions. The design goal of an FD monitor is FD within a minimum time after the occurrence of the fault while maintaining an acceptable false alarm rate. Another important goal is the number of missed alarms. This is addressed in the literature (Cunningham and Hartmann, 1977 and Deckert, et al., 1978) where data exists from flight simulation tests.

## 5.1. Multiple Trip Monitor

The FD logic in a multiple trip monitor typically requires three consecutive trips for a fault to be declared. The trip thresholds must be based on the performance of each sensor and known sensor and modeling inaccuracies.

The simplest test to check for an error signal is to compare its magnitude with predefined limits. This is illustrated in figure 5.1-1. The trip magnitude is typically placed at some multiple, m, of $\sigma$, the standard deviation of the error signal of an unfailed sensor. The mean of this error signal is zero. The value of m is chosen to yield an acceptable false alarm rate. This is shown in the following sequence of equations. The probability that there will be no false alarms during one hour of flight is given by a Poisson distribution. See Ross (1972) for more information on probability density functions, likelihood functions, and reliability.

$$P(\text{no false alarms}) = e^{-\delta t}$$

where $\delta$ = the false alarm rate (1 per 1000 flight hours)
$t$ = the flight time (1 hour)

so that P(no false alarms) = 0.999. This is also equal to the probability that a sequence of random variables, the sampled error signals, will remain within the trip boundary of $m\sigma$.

$$P\left(\bigcap_{i=1}^{n} X_i\right) = e^{-\delta t}$$

For a one hour flight, sampling at 16 Hz, n will equal 57,600.

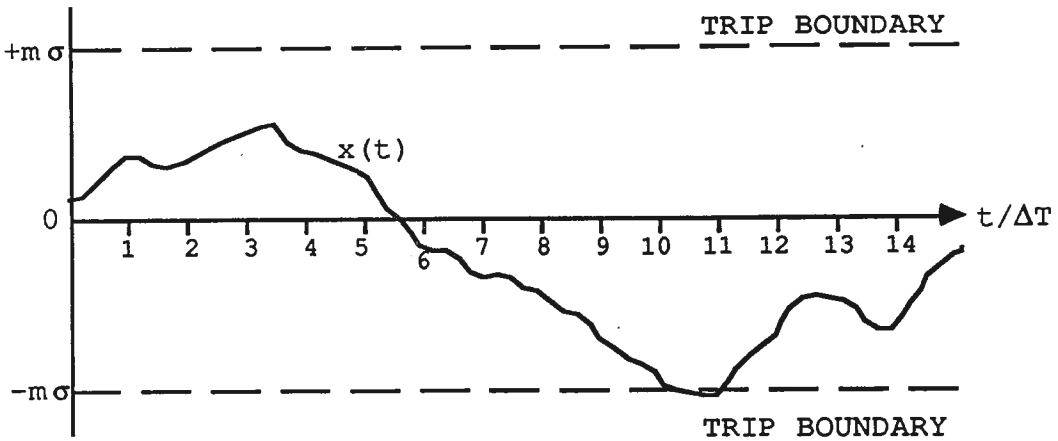



FIGURE 5.1-1. MULTIPLE TRIP FAULT DETECTION MONITOR

If the sequence of random numbers is independent, then

$$P(\bigcap_{i=1}^{n} X_i) = \prod_{i=1}^{n} P(X_i) = nP(X_i)$$

$$P(|X_i| > m\sigma) = (1 - e^{-\delta t}) / n$$

$$= 1.74 \times 10^{-8}$$

This would require m to be much larger than 4. Multiples of $\sigma$ are not given in normal probability tables for numbers in this range. For three consecutive trips,

$$P(|X_i| > m\sigma; \text{ for three consecutive times}) = (1.74 \times 10^{-8})^{1/3}$$

$$= 2.6 \times 10^{-3}$$

This corresponds to $m \approx 3.2$. This multiple of $\sigma$ was determined using a table containing probability values for a standard normal distribution.

For a completely dependent sequence,

$$P(|X_i| > m\sigma;\ \text{for three consecutive times}) \approx P(|X_i| > m\sigma;\ \text{one time})$$

$$\approx 1 - e^{-\delta t}$$

$$= 1.0 \times 10^{-3}$$

This corresponds to $m \approx 3.4$. This value was obtained from the same table used above.

The results of this analysis imply that the triple trip boundary needs to be set at $3.4\sigma$ to meet the false alarm rate for a given error signal.

The trip threshold can remain constant, adjusted based on the output of the sensor, or adjusted based on inputs from the pilot. An adjustable threshold can increase sensitivity to environmental disturbances, such as turbulence, and to aircraft in-flight maneuvering.

A multiple trip criterion (i.e., a fault is declared when a limit is exceeded n consecutive times) gives a much lower false alarm rate than a first trip monitor. The number of consecutive trips chosen for a monitor is a tradeoff between error insensitivity and the speed of FD. Choosing a multiple number of trips applies a built-in delay to FD.

## 5.2. Sequential Likelihood Ratio Tests

Likelihood ratio tests have been demonstrated to be a better FD scheme than the multiple trip monitor in Cunningham and Hartmann (1977). This algorithm provides quicker response to faults such as hardover, and more sensitivity to other smaller faults (bias and scale factor changes). However, there is more complexity to this monitoring scheme.

These tests are based on two hypothesized density functions of a random variable, $f_0(x)$ and $f_1(x)$. The likelihood ratio is expressed as a function, $\Gamma_n$, of error signal observations of that random variable.

$$\Gamma_n = \frac{f_0(e_1, e_2, \ldots, e_n)}{f_1(e_1, e_2, \ldots, e_n)}$$

The two hypothesized outcomes, $H_0$ and $H_1$, are no failure or sensor failure. These outcomes are accepted based on the following rules.

- Accept $H_0$ if $\Gamma_n \leq A$.
- Accept $H_1$ if $\Gamma_n \geq B$.
- No decision if $A < \Gamma_n < B$.

where A = 1/B. A typical value for B is 20,000.

$H_0$ implies that the expected value of a given error signal is 0 and there is no fault. $H_1$ implies the mean of the error signal has shifted unacceptably to $\mu_1$ and a sensor fault should be declared. For actual monitoring, a combined test is used where $\mu_1$ may equal $\pm$ m$\sigma$. Figure 5.2-1 illustrates the bounds for $H_0$ and $H_1$.

This can be defined mathematically using the following equations. The functions, $f_0$ and $f_1$, are assumed to have a normal distribution with a standard deviation of $\sigma$.

$$f_0(e_i) = c \exp[-\sum_{i=1}^{n} (e_i - \mu_0)^2 / 2\sigma^2]$$

$$f_1(e_i) = c \exp[-\sum_{i=1}^{n} (e_i - \mu_1)^2 / 2\sigma^2]$$

Using the no decision criteria, taking natural logarithms, and substituting 0 for $\mu_0$ and $\pm m\sigma$ for $\mu_1$,

$$\ln A < \sum_{i=1}^{n} [(e_i \mp m\sigma)^2 - (e_i)^2] / 2\sigma^2 < \ln B$$

which is equivalent to

$$nm\sigma/2 - \sigma \ln B/m < \sum_{i=1}^{n} e_i < nm\sigma/2 + \sigma \ln B/m, \text{ for } \mu_1 = +m\sigma$$

and

$$-nm\sigma/2 - \sigma \ln B/m < \sum_{i=1}^{n} e_i < \sigma \ln B/m - nm\sigma/2, \text{ for } \mu_1 = -m\sigma$$

A fault is determined when $\left| \sum_{i=1}^{n} e_i \right| \leq \sigma \ln B/m + nm\sigma/2$.

No fault is determined when $\left| \sum_{i=1}^{n} e_i \right| \geq nm\sigma/2 - \sigma \ln B/m$.

Otherwise, no decision is made.

The test sequence begins by initializing n to 1 and updates the summation each cycle. The sequence restarts when $H_0$ is accepted or there is no decision for a fixed period of time. Frequent restarts are desirable because summations over long intervals have less sensitivity to new data.

Likelihood functions of like sensors, i.e., with identical statistical error characteristics, can be compared and used to isolate the faulted sensor. The likelihood functions are $L_{n1}$ and $L_{n2}$ for sensors 1 and 2, respectively. The hypotheses here are that $H_0$ implies that sensor 1 has failed, and $H_1$ implies that sensor 2 has failed. The mean of $H_0$ is $\mu_0$ which is equal to $+m\sigma$, and the mean of $H_1$ is $\mu_1$ which is equal to $-m\sigma$.

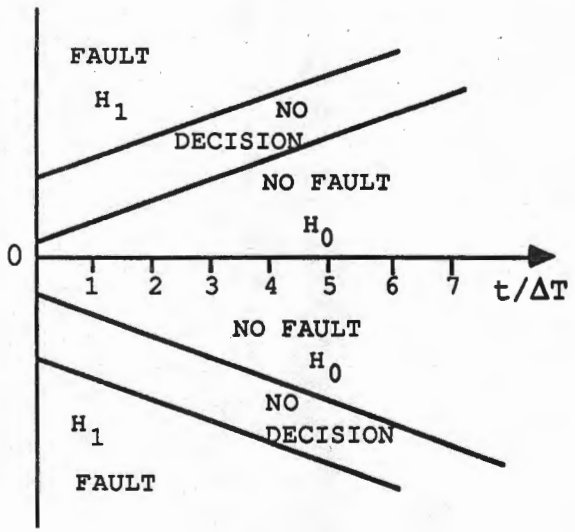



FIGURE 5.2-1. SEQUENTIAL LIKELIHOOD FAULT DETECTION MONITOR

Decisions are made using the difference of the likelihood functions based on the following rules.

$$\Delta L_n = L_{n1} - L_{n2}$$

- Accept $H_0$ if $\Delta L_n \geq \sigma \ln B/2m$.
- Accept $H_1$ if $\Delta L_n \geq \sigma \ln A/2m$.
- No decision if $\sigma \ln A/2m < \Delta L_n < \sigma \ln B/2m$.

Because the no fail hypothesis is not included here, this particular test can proceed only when a fault has already been declared. This test can be initiated after one trip using comparison monitoring. If a fault is declared (three consecutive trips), then the likelihood comparison test is used to determine which sensor is faulted. Because the likelihood functions can only increase in a fault situation, the plus or minus direction of the delta likelihood function provides the information necessary to determine which sensor has failed.

## 5.3. Fault Modeling

Sensor error and fault modeling require careful treatment for FD because typical sensor models are not sufficient and fault characteristics may not be

well-defined. A key issue is modeling sensor characteristics, including noise in an operational environment. Possible errors superimposed on the sensor output include bias, resolution, scale factor, null output, alignment error, hysteresis, environmental effects, response dynamics, and unmodeled sensor inputs. Errors may occur if the model does not include pilot inputs. Mode specific error signals may occur for selected modes, e.g., altitude or heading hold. A determination of the importance of each type of error is dependent on the type and percentage of occurrence of failure modes for each sensor. Modeling these errors may be a difficult task, because data is not easily obtained or does not exist on some of these parameters.

Faults can be classed as open-loop or closed-loop faults. An open-loop fault is one in which the sensor output is completely independent of its input. Examples of this category are hardover (at maximum output), zero output, output stuck at an intermediate value, and a randomly varying output. Closed-loop faults consist of the correct sensor output with errors superimposed. These faults are misalignment, scale factor changes, noise, or others listed in the previous paragraph.

Simulations should subject the FD algorithms to the highest probability faults. Sensor hardover, dead sensor, dynamic response reduction, scale factor changes, and bias shifts represent 90 to 95 percent of expected failures in typical rate gyros and accelerometers. These two sensor types have the highest failure rates compared to other sensors. Other faults should be defined dependent on the sensor.

Some techniques are limited in the type of failures they look for. Deckert, et al., 1978 limited their technique to looking for bias failures. They felt that most other failures could be found with the same technique. As yet, no one has addressed oscillatory faults which may or may not be important.

As defined by Cunningham and Hartmann (1977), the observer/blender and diagnostic Kalman filter worked well for FD for most sensors included in the study. Super-diagnostic techniques were found to have performance concerns and developmental needs. Problems arose in detection of accelerometer soft failure, dynamic response faults, and FI based on more than one filter detecting a given fault. (The difficulty arose when both filters did not detect the failure at the same time.)

A comparison of monitors shows that the SLRT of residual mean values performed well relative to the multiple trip monitor. SLRT caught hardover failures sooner. Frequently scale factor changes would escape the multiple trip monitor, but not the SLRT.

One way to enhance a particular FD algorithm is to utilize the fact that there are physical limitations as to how much the aircraft state can change from one sample to the next. This can minimize the effects of extremely large sensor failures. Desai, Deckert, and Deyst (1979) added a self-test in conjunction with an FD algorithm which provisionally failed a sensor when an abrupt change in output was maintained for three sample intervals.

# 6. SUPPLEMENTAL WORKED EXAMPLES

## 6.1. Altitude Kinematics

Altitude Kinematics refers to the redundancy between changes in altitude measured by the barostatic pressure sensors and changes in altitude computed from the vertical acceleration measured by the accelerometers. The vertical direction is determined via Euler attitude angles: roll ($\phi$) and pitch ($\theta$). Since the accelerometers sense gravity, the gravitational component must be subtracted in computing vertical acceleration. Vertical acceleration is computed as:

$$A_v = A_x \sin\theta - (A_y \sin\phi + A_z \cos\phi) \cos\theta - g$$

where $A_x$, $A_y$, $A_z$ are measured accelerations compensated to the center of mass.

A third order complementary filter (illustrated in figure 6.1-1) is used to store $\dot{h}$ (vertical velocity) prior to a failure detection and possible corruption. The value of $\dot{h}$ can then be used to compare altitude deviation from slightly prior to the failure detection by keeping a moving window of data.

The altitude residues for both sensors would be:

$$\text{Residue \#1} = \Sigma(\text{current } h_1 - \text{previous } h_1 - \dot{h}\, dt)$$

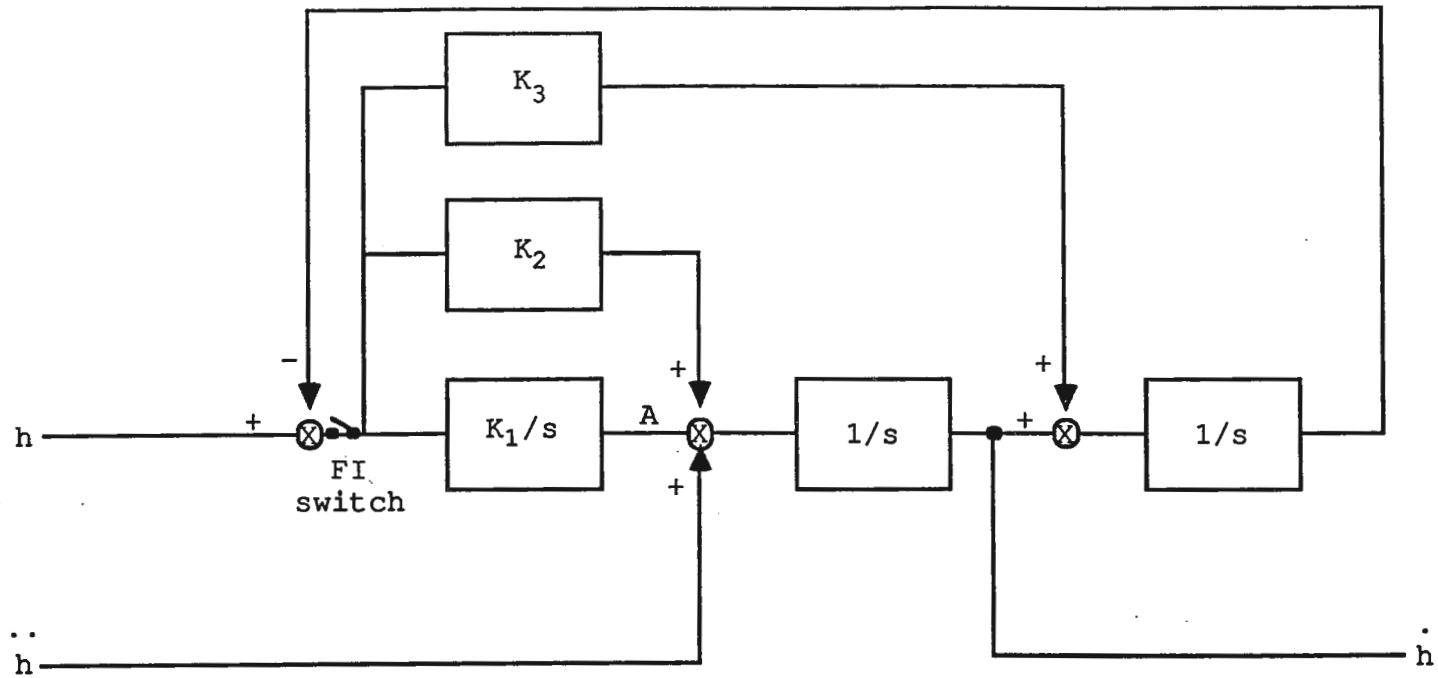
$$\text{Residue \#2} = \Sigma(\text{current } h_2 - \text{previous } h_2 - \dot{h}\, dt)$$

using h measurements from sensors 1 and 2, respectively, where dt is the integration interval.

Both residues are then evaluated by an FI algorithm. The algorithm may be a multiple trip threshold exceedance or a statistical hypothesis algorithm (e.g., SLRT).

During the FI period, the $\dot{h}$ used is only inertially updated via the switch shown in figure 6.1-1, i.e., $\dot{h} = \int \ddot{h}$. The switch opens (FI begins) when the absolute magnitude of $(h_1 - h_2)$ exceeds a threshold N times. The filter is not updated if any sample exceeds the threshold to avoid $\dot{h}$ corruption. Figure 6.1-2 demonstrates the RM processing.

Error sources must be evaluated to avoid a false FI decision. The third order filter has a state to estimate steady-state errors on $\ddot{h}$ to avoid a parabolic error growth when calculating both residues. An error analysis must be made to determine the maximum time that can be allowed for FI.




h is the average of two baro-static pressure altitudes

$\ddot{h}$ is an estimate of error of $\ddot{h}$, subsequently added to each $\ddot{h}$ calculation in the fault isolation residue equations to effectively cancel the error.

FIGURE 6.1-1 THIRD ORDER VERTICAL COMPLEMENTARY FILTER FOR ALTITUDE KINEMATICS

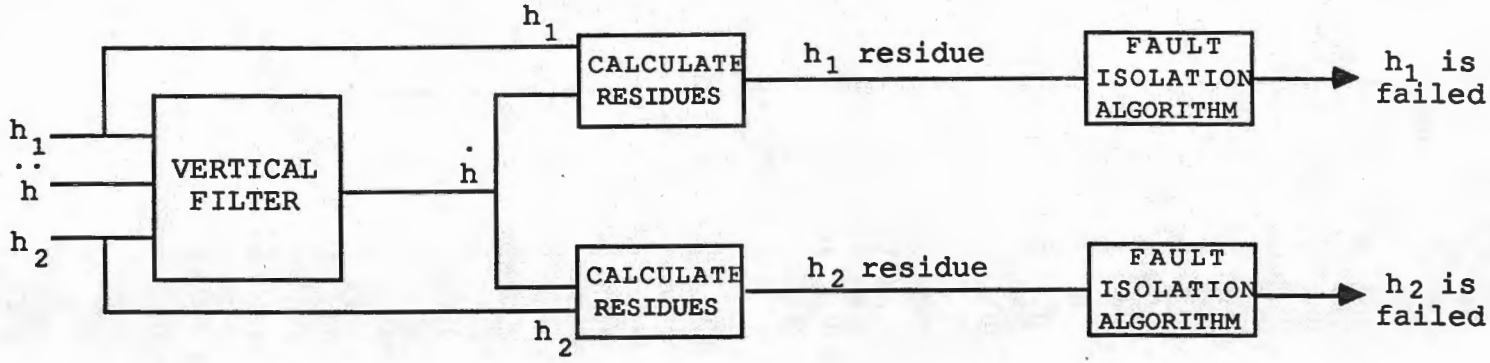



FIGURE 6.1-2. POSSIBLE REDUNDANCY MANAGEMENT FOR ALTITUDE KINEMATICS FAULT ISOLATION

If the time expires without an FI decision, then the following logic can be used:

Is there still a fault?

The answer is yes, if the absolute magnitude of $(h_1 - h_2)$ is greater than the threshold. Then isolate the sensor with the worst error history, i.e., the largest residue magnitude.

Otherwise, the answer is no. Reset the RM logic and restart the vertical filter, closing the FI switch and clearing the FD flag, if one is present.

6.2. Translational Kinematics

A translational kinematic relationship can be used to determine an airspeed sensor fault by means of the related inertial acceleration along the x-axis, such that

$$V_x = \int \dot{U}\, dt \qquad \text{or} \qquad \Delta V_x = \dot{U}\, dt$$

$$V_x = V_t \cos\beta \cos\alpha \approx V_t \qquad \text{for small angles of } \alpha \text{ and } \beta$$

where $V_x$ = x-axis airspeed velocity

$V_t$ = true airspeed

$\alpha$ = angle of attack

$\beta$ = angle of sideslip

$U$ = inertial velocity along the x-axis

An estimate of $\dot{V}_x$ can be obtained using the following equation.

$$\dot{V}_x = A_x - g \sin\theta - QW + RV + \delta_x$$

where $Q$ = pitch rate

$W$ = z-axis velocity

$R$ = yaw rate

$V$ = y-axis velocity

$\delta_x$ = wind acceleration along the x-axis

$A_x$ = acceleration measured by axial accelerometer

Q and R are measured gyro values. V and W can be calculated as follows.

$V = V_t \sin\beta$ = y-axis velocity

$W = V_t \cos\beta \sin\alpha$ = z-axis velocity

The only term unaccounted for is $\delta_x$, wind acceleration. It must be carefully analyzed as an error source in the FI decision. As long as the FI period is not too long relative to this error source, the decision should not be wrong.

Another way to alleviate this error source is to use a moving window to calculate the standard deviation of the airspeed residues (prior to FD) to determine the threshold size for FI. Wind turbulence associated with wind acceleration increases the threshold, thereby protecting from a false FI.

A filter is used to get a good estimate of any steady-state error on $\dot{V}_x$ as shown in figure 6.2-1. The term a is an estimate of error on $\dot{V}_x$, subsequently added to each $V_x$ calculation in the FI residue equations to effectively cancel the error.

$V_{tavg}$ is the average of the two true airspeed sensors. When an FD occurs the filter is no longer updated. FD is accomplished by the means discussed in section 6.1.

The airspeed residues are calculated as:

$$V \text{ Residue \#1} = \Sigma(\text{current } V_{t1} \text{ measured} - \text{previous } V_{t1} \text{ measured} - \dot{V}_x \, dt)$$

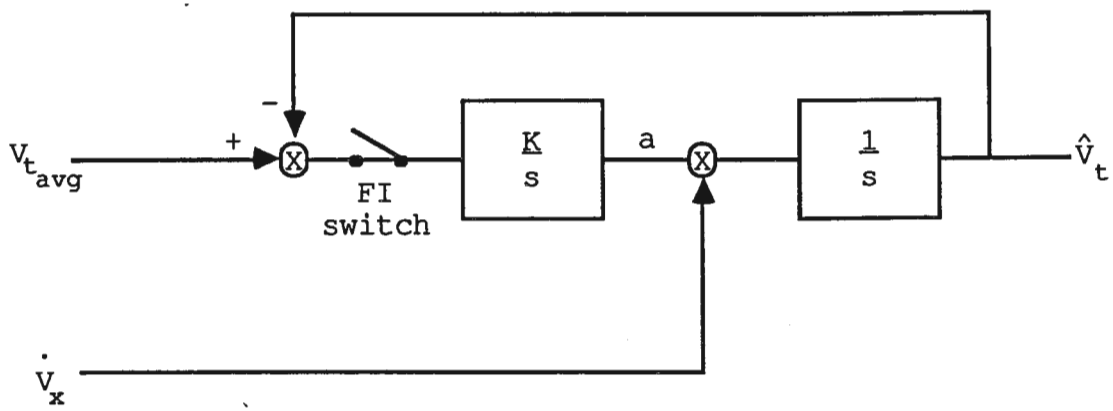
where dt is the update rate. This equation is duplicated for each airspeed sensor measurement. The two residues are then evaluated by an FI algorithm as discussed in section 6.1.

Similar comments concerning error analysis and adaptive threshold solutions apply for this example also. The standard deviation of the average of the sensors could be calculated with a moving window prior to an FD.

6.3. Stability Augmentation System

A large number of advanced aircraft are not statically stable. This is a concern because the airframe is no longer inherently stable. As an aircraft's center of gravity moves aft, the wing has a decreasing capacity to generate a nose down moment in response to an increase in angle of attack. At the neutral point, there is no pitch-axis response due to a change in angle of attack. When the center of gravity is farther aft than the neutral point, a pitch-up moment is generated from an increase in angle of attack which may make the aircraft very difficult to fly.

A Relaxed Static Stability (RSS) aircraft requires an SAS to restore stability to reasonable levels. If the SAS function is critical, it makes sense to implement AR rather than to increase the number of sensors. This function is dependent on feedback from inertial sensors, pitch angle, pitch rate, and vertical acceleration, or air data sensors, true airspeed, and angle of attack, to prevent a rapid pitch-axis divergence.




a is an estimate of error on $\dot{V}_x$, subsequently added to each $\dot{V}_x$ calculation in the fault isolation residue equations to effectively cancel the error.

FIGURE 6.2-1. EXAMPLE FILTER USED FOR TRANSLATIONAL KINEMATICS REDUNDANCY

An estimate of pitch rate and vertical acceleration can be determined with the following equations.

$$Q_E = Q_m - \dot{\theta}\cos\phi_m + \dot{\tau}\sin\phi_m\cos\theta_m$$

$$A_{zE} = A_{zm} - U_m(Q_m - \dot{\alpha}_m) - g\cos\theta_m\cos\phi_m \text{ assuming that } V = 0$$

where $Q$ = pitch rate

$A_z$ = vertical acceleration

$U$ = x-axis velocity

$g$ = acceleration due to gravity

$\phi$ = roll angle

$\theta$ = pitch angle

$\tau$ = yaw angle

$\alpha$ = angle of attack $\approx W/U$ where $W$ is the vertical velocity.

The subscript m denotes measured quantities and the subscript E denotes an estimated value.

Figure 6.3-1 shows one way to implement filters for these two estimated values. FD and isolation is as described in section 6.1.

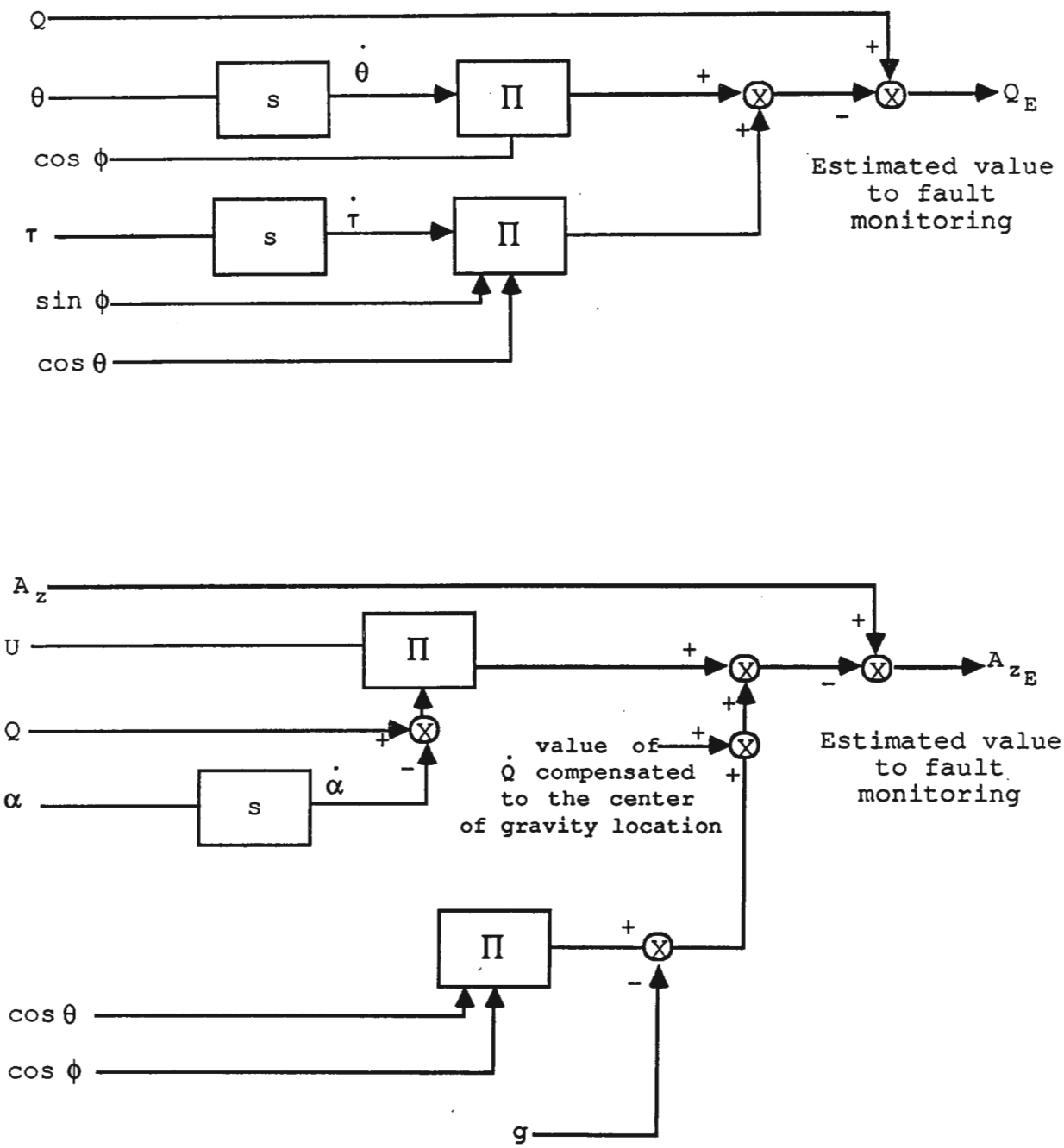



FIGURE 6.3-1. EXAMPLE FILTERS USED FOR STABILITY AUGMENTATION SYSTEM

## BIBLIOGRAPHY


Athans, M. and D. Willner, "A Practical Scheme for Adaptive Aircraft Flight Control Systems," Symposium on Parameter Estimation Techniques and Applications in Aircraft Flight Testing, NASA Flight Research Center, Edwards Air Force Base, April 24-25, 1973.

Beard, R. V., Failure Accommodation in Linear Systems Through Self-Reorganization, Report MVT-71-1, Man-Vehicle Laboratory, Cambridge, MA, February 1973.

Buxbaum, P. J. and R. A. Haddad, "Recursive Optimal Estimation for a Class of Nongaussian Processes," Proceedings of Symposium on Computer Processing in Communications, Polytech Institute of Brooklyn, April 8-10, 1969.

Chien, T. T., An Adaptive Technique for a Redundant-Sensor Navigation System, Report T-560, Draper Labs, Cambridge, MA, February 1972.

Clark, R. N., et al., "Detecting Instrument Malfunctions in Control Systems," IEEE Transactions on Aerospace and Electronic Systems, Vol. AES-11, No. 4, July 1975.

Cunningham, T., and G. Hartmann, Fault Tolerant Digital Flight Control with Analytical Redundancy, Technical Report AFFDL-TR-77-25, May 1977.

Davis, M. H. A., "The Application of Nonlinear Filtering to Fault Detection in Linear Systems," IEEE Transactions on Automatic Control, Vol. AC-20, No. 2, April 1975.

Deckert, J. C., et al., Reliable Dual- Redundant Sensor Failure Detection and Identification for the NASA F-8 DFBW Aircraft, NASA Contractor Report 2944, February 1978.

Desai, M. N., J. C. Deckert, and J.J. Deyst, "Dual Sensor Failure Identification Using Analytical Redundancy," Journal of Guidance and Control, Vol. 2, No. 3, May-June 1979.

Deyst, J. J. and J. C. Deckert, "RCS Jet Failure Identification for the Space Shuttle," Proceedings of IFAC 1975, Cambridge, MA, August 1975.

Hartmann, G. L., et al., Digital Adaptive F-8C Control Laws, Final Report NAS 1-13358, July 1975.

Jones, H. L., Failure Detection in Linear Systems, PhD Thesis, Department of Aeronautics and Astronautics, MIT, Cambridge, MA, September 1973.

Kaniuka, W. W., et al., New Flight Control Technologies for Future Naval Aircraft, Report No. NADC-82240-60, September 1982.

Kerr, T. H., "A Two Ellipsoid Overlap Test for Real-Time Failure Detection and Isolation by Confidence Regions," IEEE Conference on Decision and Control, Phoenix, AZ, November 1974.

Lainiotis, D. G., "Joint Detection, Estimation, and System Identification," Information and Control, Vol. 19, No. 1, August 1971.

Maybeck, P. S., "Failure Detection Through Functional Redundancy," AFFDL-TR-74-3, January 1974.

McAulay, R. J. and E. Denlinger, "A Decision-Directed Adaptive Tracker," IEEE Transactions on Aerospace and Electronic Systems, Vol. AES-9, March 1973.

McGarty, T. P., "State Estimation with Faulty Measurements: An Application of Bayesian Outlier Rejection," Proceedings of the Fifth Symposium on Nonlinear Estimation and Its Applications, San Diego, CA, September 1974.

Mehra, R. K. and J. Peschon, "An Innovations Approach to Fault Detection and Diagnosis in Dynamic Systems," Automatica, Vol. 7, 1971.

Meier, L., D. W. Ross, and M. B. Glaser, "Evaluation of the Feasibility of Using Internal Redundancy to Detect and Isolate On-Board Control Data Instrumentation Failures," AFFDL-TR-70-172, January 1971.

Montgomery, R. C. and A. K. Caglayan, "A Self-Reorganizing Digital Flight Control System for Aircraft," AIAA 12th Aerospace Sciences Meeting, Washington, D.C., January 30-February 1, 1974.

Montgomery, R. C. and D. B. Price, "Management of Analytical Redundancy in Digital Flight Control Systems for Aircraft," AIAA Mechanics and Control of Flight Conference, Anaheim, CA, August 5-9, 1974.

Mulcare, D. B., L. E. Downing, and M. K. Smith, Analytical Sensor Redundancy Assessment, DOT/FAA/CT-86/32, Draft Report, June 1987.

Pierce, B. D. and D. D. Sworder, "Bayes and Minimax Controllers for a Linear System with Stochastic Jump Parameters," IEEE Transactions on Automatic Control, Vol. AC-16, No. 4, August 1971.

Ratner, R. S. and D. G. Luenberger, "Performance-Adaptive Renewal Policies for Linear Systems," IEEE Transactions on Automatic Control, Vol. AC-14, No. 4, August 1969.

Ross, S. M., "Introduction to Probability Models," Academic Press, 1972.

Sanyal, P. and C. N. Shen, "Bayes' Decision Rule for Rapid Detection and Adaptive Estimation Scheme with Space Applications," IEEE Transactions on Automatic Control, Vol. AC-19, June 1974.

Stein, G. and G. L. Hartmann, F-8C Adaptive Control Extensions, NASA Contract NAS 1-13383, June 1976.

Sworder, D. D. and V. G. Robinson, "Feedback Regulators for Jump Parameter Systems with State and Control Dependent Transition Rates," IEEE Transactions on Automatic Control, Vol. AC-18, No. 4, August 1973.

Sworder, D. D., "Bayes' Controllers with Memory for a Linear System with Jump Parameters," IEEE Transactions on Automatic Control, Vol. AC-17, February 1972.

Weinstein, W., Feasibility and Design Studies of an Integrated Sensory Subsystem (ISS) for Advanced V/STOL Aircraft, Report No. NADC-76259-30, March 1978.

Willsky, A. S., J. J. Deyst, and B. S. Crawford, "Adaptive Filtering and Self-Test Methods for Failure Detection and Compensation," Proceedings of the 1974 JACC, Austin, Texas, June 19-21, 1974.

Willsky, A. S. and H. L. Jones, "A Generalized Likelihood Ratio Approach to State Estimation in Linear Systems Subject to Abrupt Changes," Proceedings of the 1974 Conference on Decision and Control, Phoenix, AZ, November 1974.

## GLOSSARY

ANALYTICAL REDUNDANCY. The use of software algorithms which use known mathematical relationships between different sensors for sensor failure detection and replace most of additional redundant sensor hardware.

CONTROL LAW. The physical relationship between various sensors and control surfaces.

COVERAGE. The percent confidence level of a given analytical redundancy fault detection and isolation algorithm for all types of faults.

DIAGNOSTIC FILTER. An analytical algorithm which processes data from N functionally related sensors. The data are used to estimate some sensor outputs and assess the correct functioning of the sensors.

DUAL FAIL-OPERATIONAL. A reliability requirement placed on a system which requires the system to be operational after two failures have occurred.

FAIL-OPERATIONAL. A reliability requirement placed on a system which requires the system to be operational after a single failure has occurred.

FAIL-SAFE. A reliability requirement placed on a system which requires that safe flight not be hindered even after a failure.

FALSE ALARM. The declaration of a fault by a fault detection monitor or algorithm when there is no fault.

FAULT DETECTION. The determination that a sensor is faulted by using a software algorithm.

FAULT ISOLATION. The determination that a particular sensor is faulted by using a software algorithm.

FAULT TOLERANCE. Accommodation of sensor hardware faults based on some type of comparator scheme.

FLIGHT-CRITICAL. A description of functions whose failure would contribute to or cause a failure condition preventing the continued safe flight and landing of the aircraft.

MISSED ALARM. The failure of a fault detection monitor or algorithm to detect a fault when there is a sensor fault.

MULTIPLE TRIP MONITOR. A fault detection algorithm which declares a fault after the sensor output has exceeded a predefined threshold N times.

OBSERVER. An algorithm which models physical relationships between sensor data and uses the data to provide fault detection for one or more sensors. This is also known as a Luenberger observer or a signal blender.

REDUNDANCY MANAGEMENT. The computer processing which is needed to implement fault detection and isolation algorithms.

REVERSION MODE. The high level of redundancy in a system having different redundancies requirements for some sensors. Critical sensors may have a high level of redundancy while other sensors have low levels.

SENSOR. An instrument which measures a particular physical parameter. The data output may be digital or analog and is utilized by the flight computer.

SEQUENTIAL LIKELIHOOD RATIO TEST. A fault detection algorithm which is based on two hypothesized density functions of no fault or sensor fault.

SEQUENTIAL PROBABILITY RATIO TEST. See sequential likelihood ratio test.

SUPER-DIAGNOSTIC FILTER. An algorithm which provides all the capabilities of a diagnostic filter. Additionally, it can isolate a specific faulted sensor. At the current time, this is the most complex technique used to implement analytical redundancy.

## ACRONYMS

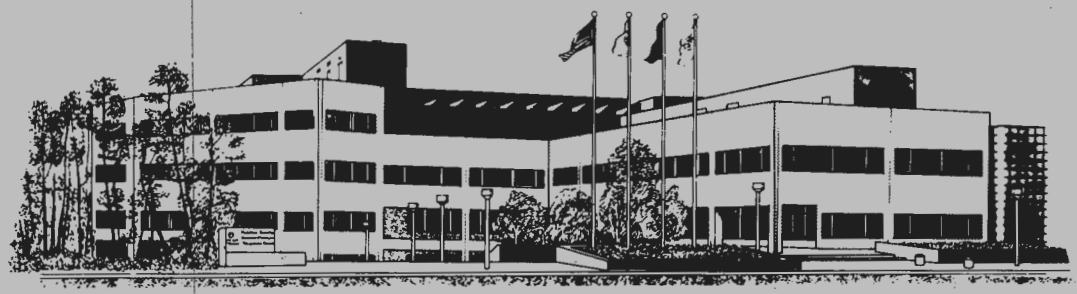
| | |
|---|---|
| AFFDL | Air Force Flight Dynamics Laboratory |
| AK | Altitude Kinematics |
| AR | Analytical Redundancy |
| C | Comparator |
| CARSRA | Computer Aided Redundant System Reliability Analysis |
| DF | Diagnostic Filter |
| DFC | Digital Flight Control |
| DFCS | Digital Flight Control System |
| DOT | Department of Transportation |
| FAA | Federal Aviation Administration |
| FBW | Fly-By-Wire |
| FD | Fault Detection |
| FI | Fault Isolation |
| NADC | Naval Air Development Center |
| NASA | National Aeronautics and Space Administration |
| RM | Redundancy Management |
| RSS | Relaxed Static Stability |
| SAS | Stability Augmentation System |
| SDF | Super-Diagnostic Filter |
| SLRT | Sequential Likelihood Ratio Test |
| SPRT | Sequential Probability Ratio Test |

U.S. Department of Transportation
Federal Aviation Administration

DOT/FAA/CT-88/10

# HANDBOOK-VOLUME II
# DIGITAL SYSTEMS VALIDATION

## CHAPTER 7
## ANALYTICAL SENSOR REDUNDANCY



**PREPARED BY:**

**COMPUTER RESOURCE MANAGEMENT, INC.**
**950 HERNDON PARKWAY, SUITE 360**
**HERNDON, VIRGINIA 22070**

**PREPARED FOR:**

**FEDERAL AVIATION ADMINISTRATION**
**TECHNICAL CENTER**
**ATLANTIC CITY INTERNATIONAL AIRPORT, NEW JERSEY 08405**

NOTICE

This document is disseminated under the sponsorship of the U.S. Department of Transportation in the interest of information exchange. The United States Government assumes no liability for the contents or use thereof.

The United States Government does not endorse products or manufacturers. Trade or manufacturers' names appear herein solely because they are considered essential to the objective of this report.

TABLE OF CONTENTS


| Section | | Page |
|---|---|---|
| 1. | INTRODUCTION | 7-1 |
| | 1.1. Control Law Modification | 7-2 |
| | 1.2. Fault Tolerant Design | 7-2 |
| | 1.3. Scope | 7-4 |
| 2. | OVERVIEW | 7-5 |
| | 2.1. Analytical Redundancy Building Blocks | 7-5 |
| 3. | ANALYTICAL REDUNDANCY - DESIGN APPROACH | 7-11 |
| | 3.1. Observer/Signal Blender Filter | 7-16 |
| | 3.2. Diagnostic Kalman Filters | 7-16 |
| | 3.3. Super-Diagnostic Kalman Filter Design | 7-16 |
| | 3.3.1. Multiple Hypothesis Tests | 7-17 |
| | 3.3.2. Parameter Identification | 7-17 |
| | 3.3.3. Generalized Likelihood Ratio Methods | 7-17 |
| | 3.3.4. Modified Filter Design | 7-18 |
| | 3.3.5. Jump Process Estimation | 7-18 |
| | 3.4. Redundancy Management | 7-18 |
| 4. | ANALYTICAL REDUNDANCY - RELIABILITY | 7-21 |
| 5. | ANALYTICAL REDUNDANCY - FAULT DETECTION AND ISOLATION | 7-23 |
| | 5.1. Multiple Trip Monitor | 7-23 |
| | 5.2. Sequential Likelihood Ratio Tests | 7-25 |
| | 5.3. Fault Modeling | 7-27 |
| 6. | SUPPLEMENTAL WORKED EXAMPLES | 7-29 |
| | 6.1. Altitude Kinematics | 7-29 |
| | 6.2. Translational Kinematics | 7-32 |
| | 6.3. Stability Augmentation System | 7-33 |
| BIBLIOGRAPHY | | 7-37 |
| GLOSSARY | | 7-40 |
| ACRONYMS | | 7-42 |

## LIST OF ILLUSTRATIONS


| Figure | | Page |
|---|---|---|
| 2-2 | ROTATIONAL RELATIONSHIPS USED FOR ANALYTICAL REDUNDANCY | 7-7 |
| 2.1-1 | ANALYTICAL REDUNDANCY CONCEPT | 7-8 |
| 2.1-2 | ANALYTICAL SENSOR REDUNDANCY FAULT DETECTION LOGIC | 7-9 |
| 3-1 | EXAMPLES OF FAULT DETECTION USING DIAGNOSTIC FILTERS AND INTERCHANGEABLE SENSORS | 7-12 |
| 3-2 | EXAMPLES OF FAULT DETECTION USING DIAGNOSTIC FILTERS AND NONINTERCHANGEABLE SENSORS | 7-13 |
| 3-3 | EXAMPLE OF FAULT DETECTION WITH REVERSION MODES | 7-15 |
| 5.1-1 | MULTIPLE TRIP FAULT DETECTION MONITOR | 7-24 |
| 5.1-2 | SEQUENTIAL LIKELIHOOD FAULT DETECTION MONITOR | 7-27 |
| 6.1-1 | THIRD ORDER VERTICAL COMPLEMENTARY FILTER FOR ALTITUDE KINEMATICS | 7-30 |
| 6.1-2 | POSSIBLE REDUNDANCY MANAGEMENT FOR ALTITUDE KINEMATICS FAULT ISOLATION | 7-31 |
| 6.2-1 | EXAMPLE FILTER USED FOR TRANSLATIONAL KINEMATICS REDUNDANCY | 7-34 |
| 6.3-1 | EXAMPLE FILTERS USED FOR STABILITY AUGMENTATION SYSTEM | 7-36 |


## LIST OF TABLES


| Table | | Page |
|---|---|---|
| 3-1 | SENSOR REDUCTION COMPARISONS FOR ANALYTICAL REDUNDANCY BUILDING BLOCKS | 7-14 |
| 4-1 | SYSTEM RELIABILITY | 7-22 |

# 1. INTRODUCTION

The advent of modern digital controls has greatly changed the control technology and hardware implementation of aircraft flight control. These Digital Flight Control Systems (DFCS) provide enhanced performance for stability and control augmentation over an extremely wide range of flight envelope parameters. They also offer improved capabilities to operate in differing aircraft configurations from takeoff with wheels and flaps down to a high speed clean configuration with low drag, thrust, and low g forces. In addition to the performance enhancements in Digital Flight Control (DFC), the DFC computer has the capability to perform other computations to effect improvements in reliability and fault tolerance. Elaboration of these later features is the major thrust of this portion of the handbook.

The DFC improves reliability due to two factors. The first is the increased reliability of digital computer hardware relative to older technology; this enhancement leads to tens of thousands of hours between failures. The second factor is the increased reliability of the sensors and actuators which leads to several hundreds of hours between failures. The principal factor affecting DFC system reliability is reliability of the sensors and actuators.

The overall reliability required in today's flight control systems dictates the need for $10^{-9}$ probability of failure in cases where stability augmentation is needed for safe flight. This requires an increase in the reliability of the DFC components. The reliability of DFC computers may be achieved by utilizing dual processors, error detection, and fault tolerant designs. An increase in reliability of sensors and actuators may be achieved by means of redundant elements (dual or triple).

Assessment of this situation leads to a conclusion that redundant actuators and linkages are an adequate solution for actuation, and that DFC computers are adequate. The term Analytical Redundancy (AR) refers to the scheme for increasing the reliability of sensors by using the DFC computer for Fault Detection (FD) and isolation and for analytically determining estimates of sensor values.

Reliable DFC systems are burdened with large numbers of sensors which are required for both quality and redundancy. The demand for control quality has expanded the types of sensors used in basic augmentation functions. The yaw damper currently requires sensor data concerning attitude, angle of attack, and airspeed. The pitch damper requires a complex multifunction control law along with data from several sensors to drive multiple surfaces. The complexity of the sensors has improved quality; however, designs that meet overall flight control system reliability requirements lead to triple and quadruple replication of system components.

High levels of fault tolerance are mandatory for flight-critical control systems. Current Fly-By-Wire (FBW) systems utilize redundant computers,

actuators, and sensors to achieve the necessary fault tolerance. Fault isolation (FI) is determined by some type of comparison or voting between like sensors. A one out of two sensor failure cannot be isolated by comparison techniques. In order to improve the reliability of a redundant set of sensors, one or more of the following must be implemented: increase the reliability of the existing sensors; add additional like sensors; synthesize the needed information; or isolate the remaining sensor if a failure occurs at the duplex level.

A large amount of sensor hardware is costly from a system standpoint. Sensors may act as antennas to increase electromagnetic susceptibility. The overall aircraft design may be constrained due to a large number of sensors. Therefore, there is a strong motivation to reduce the number of sensors. Significant reductions appear to be possible through a variety of methods. Recognized techniques for sensor reduction are control law modification and fault tolerant design. These methods may be used in combinations to optimize the number of sensors necessary for a particular application.

## 1.1. Control Law Modification

Control law modification ensures that operational requirements can be met with the minimum number of sensors and a better control system design. The control laws are modified to optimize the use of sensed data utilizing the analytical relationships which exist between sensors. The flight controller uses fewer sensed values in the basic single channel control laws, yet maintains the same performance as the controller which utilized all the sensor data. These modifications are application dependent.

Signal synthesis is one example of control law modification. The number of sensors required is reduced by synthesizing a signal or group of signals from a reduced set. The elements used to perform the synthesis are Kalman filters or Luenberger observers. Reduction in the number of sensors is a trade-off against increased complexity of the control laws; this is a critical factor for analog systems. The trade-off in digital systems occurs in system performance. The bandwidth and noise content of the synthesized signal must be equivalent to the sensor eliminated. These techniques must be applied with care because of potential physical limitations which cannot be practically overcome.

Integrated flight management is another example of control law modification. This method works on the principle that subsystems which use common sensor types are combined. An example is the combination of the navigator and the flight control. The signal quality derived from the navigator is high and the data could easily be used by the flight control. Issues here are cost (since the navigator is normally not flight critical) and multiple failures.

## 1.2. Fault Tolerant Design

The following fault tolerant design techniques exist to meet reliability requirements using a minimum number of sensors. These techniques use fewer sensors to achieve the same level of redundancy as a quad-redundant system.

- Skewed and special sensors.

- Sensor integration.
- In-line sensor monitoring.
- Analytical sensor redundancy.

A skewed sensor arrangement significantly reduces the number of sensors required for Redundancy Management (RM). Skewed sensors do not have to be placed on orthogonal axes. The x, y, and z components of the sensed quantity can be determined using an appropriate coordinate transformation matrix. If one of the skewed sensors fails, the three orthogonal components can still be determined with another sensor and its transformation matrix.

A quad-redundant dual fail-operative system with orthogonal gyros in three axes requires twelve gyros. The same properly configured system with skewed gyros requires only six to maintain the same redundancy requirements. Skewing accelerometers can similarly reduce the number of sensors from twelve to six. Potential limitations are differences in scale and resolution requirements for each axis. For the skewed configuration, each gyro must have the same specifications. Depending upon the gyro chosen, either the signal quality may be limited or the instrumentation cost may be greater. Weinstein (1978) and Kaniuka, et al., 1982 show adequate performance for this technique.

Special sensors measure linear combinations of variables. The measurements are skewed in measurement space rather than geometrically, i.e., the sensor measures both acceleration and body rate. A conventional quad-redundant system could be replaced by five special sensors. These sensors are typically unique to the application and have a higher cost. A multipurpose air data sensor which measured pitot pressure, static pressure, angle of attack, and angle of sideslip was implemented in the system described by Weinstein (1978) and Kaniuka, et al., 1982.

Subsystem sensor integration can also be used to reduce the number of sensors. This technique uses data from subsystems which are not functionally related for monitoring and tie-breaking. An example is on the space shuttle where derived rates from the navigation system are used in the primary flight control system for voting.

Fault tolerant design can be achieved through in-line monitoring. The state-of-the-art for servos and computers is near 99 percent self-test confidence. In-line self-test feasibility for sensors is limited for several reasons. The sensor input is unknown except for special test signals. There is no way to test for sensor installation errors. Any sensor self-test adds considerable complexity which may be undesirable. Typically, a self-test can only detect very large faults, particularly in a dynamic environment.

Analytical Redundancy provides a means for substituting software algorithms for hardware sensors. Sensors are eliminated by analysis using known relationships between sensed values. Various simulation studies have shown the effectiveness of utilizing AR to increase system reliability and to reduce the amount of sensor hardware.

## 1.3. Scope

This document details the uses and limitations of analytical sensor redundancy. Section 2 gives an overview of AR. The physical relationships which can be utilized are written in equation form. The basic concept and building blocks are discussed. Section 3 describes various algorithms and design tools used to implement the filters used in AR processing. Its feasibility and performance are discussed in section 4. Algorithms and methods used for FD and isolation are detailed in section 5. Examples of the filters, RM, and sensor relationships for three cases where analytical sensor redundancy is easily implemented are given in section 6.

# 2. OVERVIEW

The basic idea of AR is to use known geometric and dynamic relationships between different sensors in order to detect and isolate sensor failures. These relationships can be defined in translational and rotational equations as shown in figures 2-1 and 2-2. Figure 2-2 also shows a diagram of the Euler angle relationships.

Cunningham and Hartmann (1977) describe the application of these mathematical relationships to eleven different measured quantities, normal acceleration, lateral acceleration, angle of attack, true airspeed, altitude, P, Q, R, $\emptyset$, $\theta$, and $\psi$.

Four categories of AR were utilized by Deckert, et al., 1978. Translational kinematics redundancy exists between the integrated accelerometer outputs, vertical and rate gyros, and the air data sensor outputs. Translational dynamics redundancy uses the aerodynamic forces on the aircraft measured by the accelerometers and stored coefficients to relate to aerodynamic forces calculated from air-data sensor data. Rotational kinematics redundancy relates the integrated outputs of the rate gyros and the vertical and directional gyro outputs. Altitude Kinematics (AK) redundancy is defined as the relationship between the altitude measured by the altimeter and the computed altitude from a double integration of the accelerometer and vertical gyro output data. These relationships are defined in the equations in figures 2-1 and 2-2. Examples of altitude and translational kinematics redundancy processing are given in sections 6.1 and 6.2.

## 2.1. Analytical Redundancy Building Blocks

Many specific approaches to AR have been developed, ranging from simple signal blenders to complex combinations of Kalman filters.

In general, AR is approached using one of two basic building blocks. A Diagnostic Filter (DF) is an algorithm which processes data from a family of N functionally related sensors in order to estimate signals and to assess the functioning of the sensors. It outputs signal estimates and an error flag which indicates that all sensors are functioning properly or that one of them has failed. A Super-Diagnostic Filter (SDF) performs the functions of a DF and can additionally isolate the specific faulted sensor. One can use assemblies of DFs to construct an SDF through truth table logic.

The concept of AR and associated FD are shown in figures 2.1-1 and 2.1-2.

Translational Equations

Inertial Velocity

$$\dot{X}_e = \cos\theta \cos\tau\, U + (\sin\phi \sin\theta \cos\tau - \cos\phi \sin\tau)V + (\cos\phi\sin\theta \cos\tau + \sin\phi \sin\tau)W$$

$$\dot{Y}_e = \cos\theta \sin\tau\, U + (\sin\phi \sin\theta \sin\tau + \cos\phi \cos\tau)V + (\cos\phi \sin\theta \cos\tau - \sin\phi \cos\tau)W$$

$$\dot{Z}_e = -\sin\theta\, U + \sin\phi \cos\theta\, V + \cos\phi \cos\theta\, W$$

Acceleration

$$\dot{U} = A_x - g \sin\theta - QW + RV$$

$$\dot{V} = A_y + g \cos\theta \sin\phi - RU + PW$$

$$\dot{W} = A_z + g \cos\theta \cos\phi - PV + QU$$

where $X_e$, $Y_e$, $Z_e$ = earth referenced forward, lateral, and vertical positions
U, V, W = forward, lateral, and vertical body velocities, respectively
$A_x$, $A_y$, $A_z$ = forward, lateral, and vertical body accelerations
g = acceleration due to gravity
$\theta$, $\phi$, $\tau$ = Euler pitch, roll, and yaw angles
P, Q, R = body roll, pitch, and yaw rates

FIGURE 2-1. TRANSLATIONAL RELATIONSHIPS USED IN ANALYTICAL REDUNDANCY

**Rotational Equations**

Euler Rates

$$\dot{\phi} = P + (Q \sin\phi + R \cos\phi) \tan\phi$$

$$\dot{\theta} = Q \cos\phi - R \sin\phi$$

$$\dot{\tau} = (Q \sin\phi + R \cos\phi) \sec\theta$$

Body Rates

$$P = \dot{\phi} - \dot{\tau} \sin\theta$$

$$Q = \dot{\theta} \cos\phi - \dot{\tau} \sin\phi \cos\theta$$

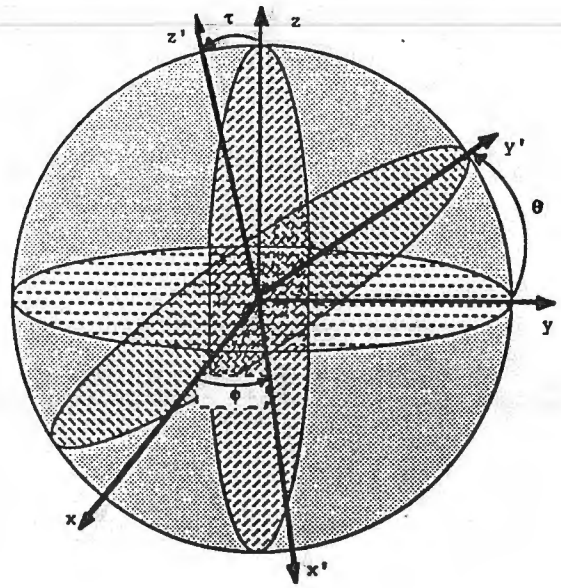
$$R = \dot{\tau} \cos\phi \cos\theta - \dot{\theta} \sin\phi$$

Moments

$$L = \dot{P} I_{xx} - \dot{R} I_{xz} + QR (I_{zz} - I_{yy}) - PQ I_{xz}$$

$$M = \dot{Q} I_{yy} + PR (I_{xx} - I_{zz}) + (P^2 - R^2) I_{xz}$$

$$N = \dot{R} I_{zz} - \dot{P} I_{xz} + PQ (I_{yy} - I_{xx}) + QR I_{xz}$$




EULER ANGLE DIAGRAM

where $\theta$, $\phi$, $\tau$ = Euler pitch, roll, and yaw angles
P, Q, R = body roll, pitch, and yaw rates
L, M, N = moment about the body x, y, and z axes
$I_{xx}$, $I_{yy}$, $I_{zz}$ = moment of inertia about the x, y, and z axes
$I_{xz}$ = cross moment of inertia (xz axis)

FIGURE 2-2. ROTATIONAL RELATIONSHIPS USED FOR ANALYTICAL REDUNDANCY

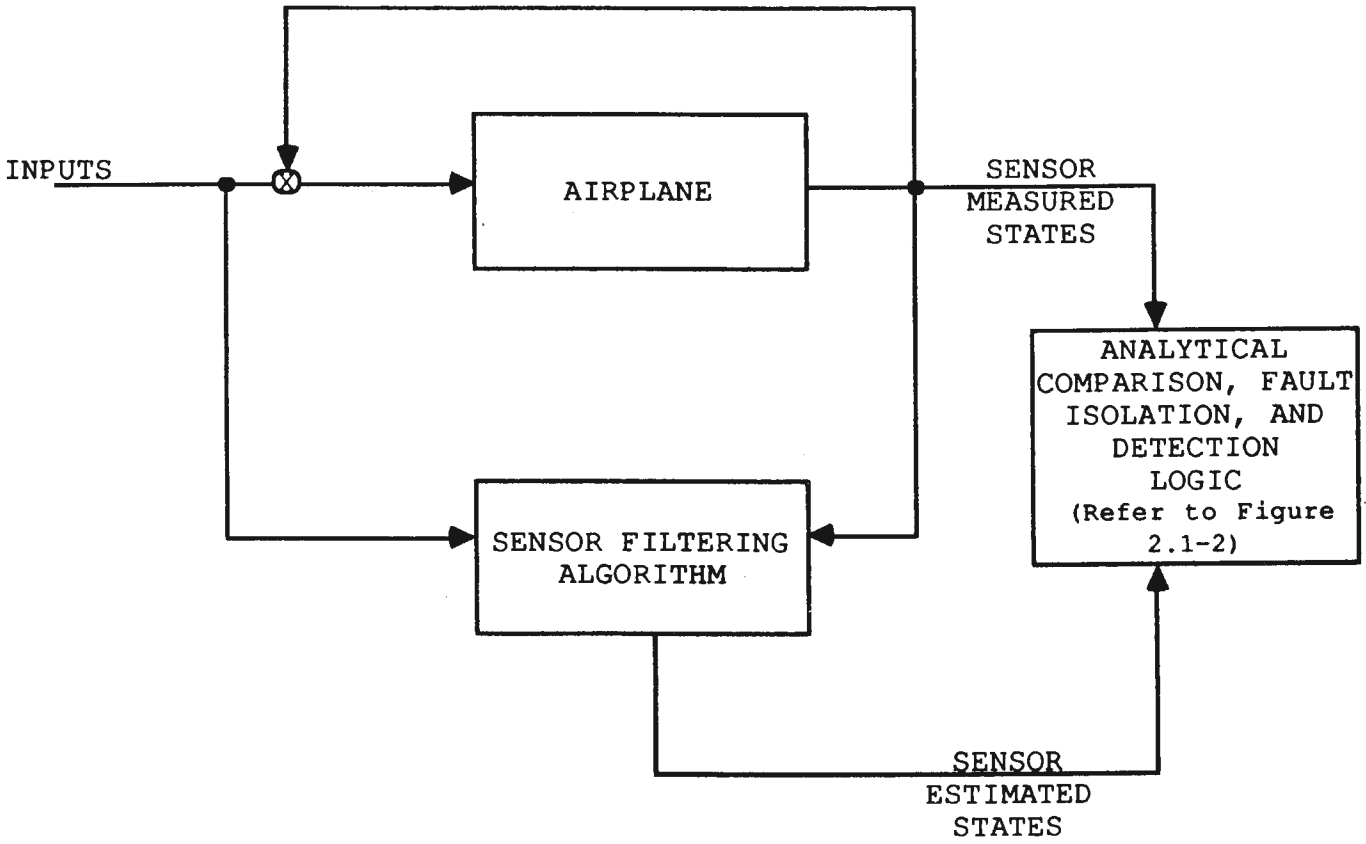



FIGURE 2.1-1. ANALYTICAL REDUNDANCY CONCEPT

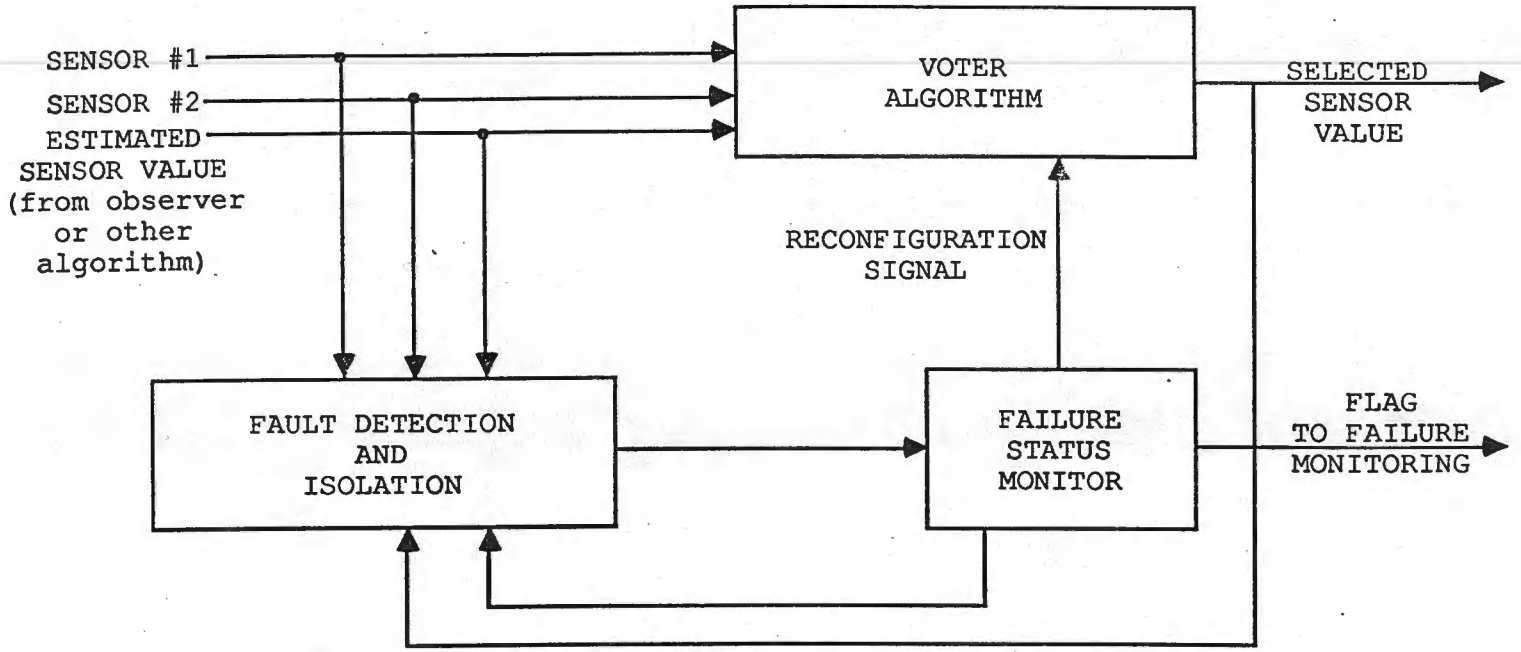



FIGURE 2.1-2. ANALYTICAL SENSOR REDUNDANCY FAULT DETECTION LOGIC

## 3. ANALYTICAL REDUNDANCY - DESIGN APPROACH

A literature survey was performed by Cunningham and Hartmann (1977) to examine currently available techniques that were able to detect and isolate faults in aircraft sensors and meet current onboard flight computer allocations. All such methods for failure detection were found to lie in three categories.

- Assemblies of DFs (see Meier, Ross, and Glaser, 1971; Maybeck, 1974; Hartmann, et al., 1975; and Clark, et al., 1975).
- Specific DF design techniques (see Kerr, 1974 and Mehra and Peschon, 1971).
- Explicit SDF design techniques.

The first two categories also include observer/blender techniques. Mulcare, Downing, and Smith (1987) and Deckert, et al., 1978 utilize these techniques.

The three filter types listed above can be combined in many ways to obtain useable AR results. The end product may vary considerably in accuracy and complexity. The structure of the filter depends on the overall RM approach, whether uniform redundancy or different levels of redundancy are required throughout the system.

Uniform redundancy schemes categorize sensors as interchangeable or noninterchangeable. Interchangeable sensors can substitute for each other when a failure occurs. Appropriate coordinate transformations must be accounted for, if necessary. If a noninterchangeable sensor fails, an identical one must replace it in order to maintain performance.

If N fully operational sensors are required to maintain aircraft operations, then a minimum of N+2 interchangeable sensors and 3N noninterchangeable sensors are required for dual fail-operational performance when using AR. In order to resolve conflicts using traditional voting techniques, N+3 and 4N sensors are needed to maintain the same performance.

Fault Detection using DFs with interchangeable sensors is illustrated in figure 3-1. The figure shows FD for N, the number of sensors required for safe flight, equal to 2 for fail-operational performance. The truth table shows that each column of the error flags is a unique binary word and that the sensor inputs to each filter are uniquely determined by each row of the table. The generalized way to assemble DFs for interchangeable sensors is given below.

- Determine the number of sensors required to maintain safe flight and meet operational requirements (N is the number of sensors necessary for safe flight).

N + 1 for fail-operational performance
N + 2 for dual fail-operational performance

- Determine the number of DFs (the brackets indicate to use the next largest integer of the enclosed expression).

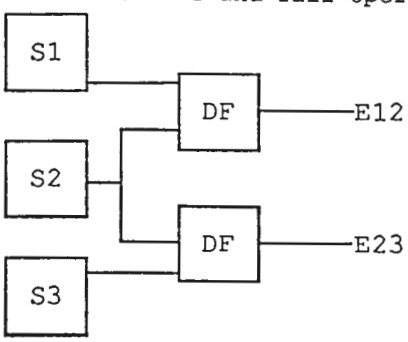
$[\log_2 (N + 2)]$ for fail-operational performance
$[\log_2 (N + 3)]$ for dual fail-operational performance

- Determine sensor inputs to each filter from the truth table.

The FD capability of a DF combined with comparators for noninterchangeable sensors can also be verified with a truth table. See figure 3-2 for an illustration of fail-operational performance with N, the number of sensors necessary to maintain safe flight, equal to 2. The generalized arrangement uses one filter and N comparators for fail-operational and one filter and 2N comparators for dual fail-operational. The truth table here contains FD for single sensor failures only. It is generally assumed that failures occur singly when designing FD algorithms. A single SDF is all that is needed in all cases to provide FD and isolation. This type of filter may be quite complex in order to achieve the same results as DFs.

INTERCHANGEABLE SENSORS

for N=2 and fail-operational performance



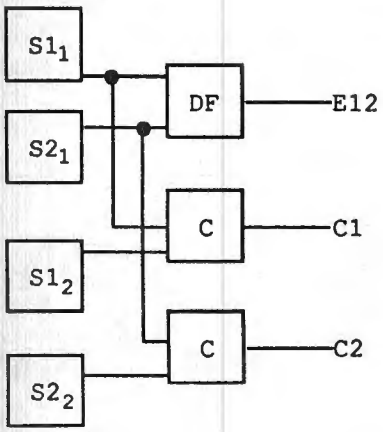

TRUTH TABLE

| ERROR FLAGS | SINGLE SENSOR FAILURE | | | |
|---|---|---|---|---|
| | NONE | S1 | S2 | S3 |
| E12 | 0 | 1 | 1 | 0 |
| E23 | 0 | 0 | 1 | 1 |

FIGURE 3-1. EXAMPLES OF FAULT DETECTION USING DIAGNOSTIC FILTERS AND INTERCHANGEABLE SENSORS

NONINTERCHANGEABLE SENSORS

for N=2 and fail-operational performance




TRUTH TABLE

| ERROR FLAGS | SINGLE SENSOR FAILURE | | | | |
|---|---|---|---|---|---|
| | NONE | $S1_1$ | $S2_1$ | $S1_2$ | $S2_2$ |
| E12 | 0 | 1 | 1 | 0 | 0 |
| C1 | 0 | 1 | 0 | 1 | 0 |
| C2 | 0 | 0 | 1 | 0 | 1 |

C = Comparator
DF = Diagnostic Filter
N = # of sensors needed to maintain safe flight

FIGURE 3-2. EXAMPLES OF FAULT DETECTION USING DIAGNOSTIC FILTERS AND NONINTERCHANGEABLE SENSORS

The largest amount of sensor reduction is achieved for noninterchangeable sensors when using AR. In this case, AR implemented with one DF in combination with 2N comparators has the same benefit as one SDF. An SDF which is more complex will not be feasible. A comparison of the sensor reductions possible with these analytical techniques and traditional voting is given in table 3-1. The number of comparators and DFs needed for AR are also given. N is defined as the minimum number of sensors required to maintain safe flight.

There is a sensor reduction benefit for interchangeable sensors; however, it is a small one. AR provides little or no benefit to save one sensor. More DFs are needed to match the capabilities of a single SDF than for noninterchangeable sensors.

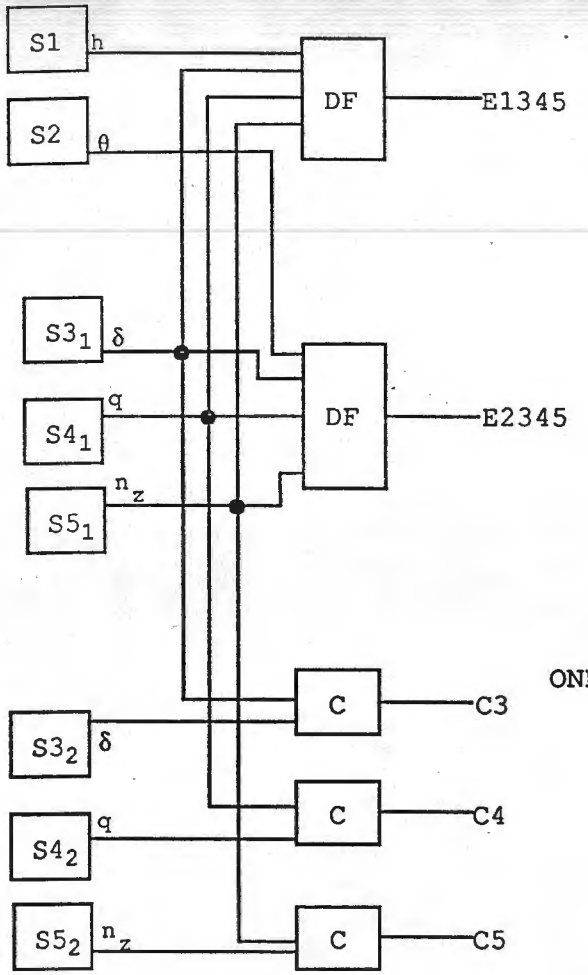
Current flight control systems normally do not have the same redundancy requirements for all sensors. A more realistic setup may have high levels of redundancy for some critical sensors and lower levels for others. The high levels of redundancy are known as reversion modes. While this may be application dependent, the basic principles can be used to obtain a failure detection scheme as shown in figure 3-3. In order to implement the reversion modes, sufficient filters are required to distinguish between four outer loop logic conditions: no failure; S1 failed; S2 failed; and $S3_1$, $S4_1$, or $S5_1$ failed. The error flags, E1345 and E2345, are used to obtain this information. The error flags, C3, C4, and C5, must be used to determine which of the inner loop sensors have failed.

TABLE 3-1. SENSOR REDUCTION COMPARISONS FOR ANALYTICAL REDUNDANCY BUILDING BLOCKS

| **FAIL-OPERATIONAL** | NUMBER OF SDF'S | NUMBER OF DIAGNOSTIC FILTERS | NUMBER OF COMPARATORS | NUMBER OF SENSORS |
|---|---|---|---|---|
| **INTERCHANGEABLE SENSORS** | | | | |
| DETECTION WITH SDF'S | 1 | 0 | 0 | N+1 |
| WITH DF'S | 0 | $[\text{LOG}_2(N+2)]$ | 0 | N+1 |
| WITH VOTING | 0 | 0 | N+3 | N+2 |
| **NONINTERCHANGEABLE SENSORS** | | | | |
| DETECTION WITH SDF'S | 1 | 0 | 0 | 2N |
| WITH DF'S | 0 | 1 | N | 2N |
| WITH VOTING | 0 | 0 | 2N | 3N |
| **DUAL FAIL-OPERATIONAL** | NUMBER OF SDF'S | NUMBER OF DIAGNOSTIC FILTERS | NUMBER OF COMPARATORS | NUMBER OF SENSORS |
| **INTERCHANGEABLE SENSORS** | | | | |
| DETECTION WITH SDF'S | 1 | 0 | 0 | N+2 |
| WITH DF'S | 0 | $[\text{LOG}_2(N+3)]$ | 0 | N+2 |
| WITH VOTING | 0 | 0 | N+4 | N+3 |
| **NONINTERCHANGEABLE SENSORS** | | | | |
| DETECTION WITH SDF'S | 1 | 0 | 0 | 3N |
| WITH DF'S | 0 | 1 | 2N | 3N |
| WITH VOTING | 0 | 0 | 3N | 4N |

N = MINIMUM NUMBER OF SENSORS REQUIRED FOR SAFE FLIGHT

[ ] IMPLIES USE THE NEXT LARGEST INTEGER OF THE EXPRESSION WITHIN THE BRACKETS




TRUTH TABLE

| ERROR FLAGS | SENSORS FAILED | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | NONE | S1 | S2 | $S3_1$ | $S4_1$ | $S5_1$ | $S3_2$ | $S4_2$ | $S5_2$ |
| E1345 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| E2345 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| C3 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| C4 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| C5 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |

TWO FAIL-SAFE OUTER LOOPS (h), (θ)

ONE FAIL-OPERATIONAL INNER LOOP ($\delta, q, n_z$)
(NON-INTERCHANGEABLE SENSORS)

- h = altitude
- θ = Euler pitch angle
- δ = control surface deflection
- q = body pitch rate
- $n_z$ = normal acceleration
- C = Comparator
- DF = Diagnostic Filter

FIGURE 3-3. EXAMPLE OF FAULT DETECTION WITH REVERSION MODES

Three concepts encompass the categories of failure detection methods. These are examined as a result of various combinations of DFs. These concepts in order of increasing complexity are:

- Observer/signal blender filter.
- Diagnostic Kalman filters.
- Super-diagnostic Kalman filter design.

## 3.1. Observer/Signal Blender Filter

An observer/blender will model physical relationships using sensor outputs to provide FD for several sensors. It uses a specific DF design and has the least amount of complexity and computer requirements. Monitors may use a three trip level exceedance criterion to detect failures. Trip thresholds can be set to account for unmodeled dynamics and sensor anomalies.

The basic operation of the observer/blender is to define a consistent error function between a measured parameter and its derivative. Both values are measured for all but a few parameters, e.g., lateral acceleration. A relationship such as,

$$V_x = \dot{X}$$

is exact. For perfect measurements, there is no error. Realistically, the measured values of $V_x$ and X are not true representations of $V_x$ and X. There is an error associated with digital integration and sampling. Either $V_x$ or X may contain faults. This implementation of AR must distinguish between the errors associated with measurements and digital sampling and errors associated with the sensors.

## 3.2. Diagnostic Kalman Filters

Diagnostic Kalman filters incorporate modeling of sensor anomalies (such as, bias, scale factors, and wind gust estimation) in the filter. FD is provided for more sensors than available with the observer/blender. This is achieved by using an assembly of DFs. This concept places moderate requirements on a computer.

## 3.3. Super-Diagnostic Kalman Filter Design

The super-diagnostic design utilizes Kalman filters as well as linear equations for pitch and lateral directional dynamic equations of motion. Both detection and isolation can be determined. An error signal is created for each sensor involved. The concept is built with an assembly of DFs. Some applications may be very complex.

The largest number of surveyed references are devoted to this topic. These filters are the most difficult to construct. They are constrained more than other techniques due to the amount of complexity which is feasible. Five methods have been examined for SDFs.

- Multiple hypothesis test.
- Parameter identification.
- Generalized likelihood ratio methods.
- Modified filter designs.
- Jump process estimation.

### 3.3.1. Multiple Hypothesis Tests

Each failure condition of a particular sensor is used to define a hypothesis. The entire range of failure conditions forms a set of hypotheses which are tested against observed data. One Kalman filter is required to test each hypothesis. The filter generates residuals, assuming that the hypothesis is true, which are summed into likelihood functions. The maximum likelihood then identifies the current valid hypothesis. This results in a large number of filters for even a few sensors and at present is too complex for advanced flight computers to utilize. Montgomery and Caglayan (1974), Montgomery and Price (1974), Athans and Willner (1973), Lainiotis (1971), and Buxbaum and Haddad (1969) contain more information.

### 3.3.2. Parameter Identification

Parameters of each sensor, such as, gain and bias, are defined as unknowns and estimated from sensor data. If the estimates deviate a great deal from nominal values, then a failure is declared. For any procedure used, a separate calculation similar to a Kalman filter is required to compare the nominal values with the estimated sensor parameters. This becomes as complex and inefficient as the previous technique. Mehra and Peschon (1971) and Stein and Hartmann (1976) have investigated this technique.

### 3.3.3. Generalized Likelihood Ratio Methods

In an attempt to reduce the complexity of this type of failure detection method, initial constraints were placed on the design of an algorithm. These constraints are that only one Kalman filter is available and that sensor failure detection must be decided through the monitoring of residuals from the single filter. This can be done for those failures which have recognizable signatures for correlation processing. The residuals are correlated with known signals, normalized by their expected no-failure value, and compared to predetermined threshold levels. The number of filters has been greatly reduced, however, the correlation functions can be fairly complex. See Willsky, Deyst, and Crawford (1974); Willsky and Jones (1974), Deyst and Deckert (1975), McAulay and Denlinger (1973), and Sanyal and Shen (1974).

### 3.3.4. Modified Filter Design

This method starts with a constraint of one Kalman filter. It does not allow the filter to remain a fixed element. An attempt is made to alter the filter gains to produce strong and easily recognized failure signatures. This concept is largely theoretical at this time. Details are given in Jones (1973) and Beard (1973).

### 3.3.5. Jump Process Estimation

This procedure is also theoretical. The basic idea is to represent failures as random jump processes in an otherwise known stochastic system. The time of occurrence and magnitude of the jumps are then estimated using optimal stochastic filtering theory. (See Sworder, 1972; Sworder and Robinson, 1973; Ratner and Luenberger, 1969; Pierce and Sworder, 1971; Davis, 1975; McGarty, 1974; and Chien, 1972.)

## 3.4. Redundancy Management

Redundancy Management will be optimized when included in a system design, but it can be incorporated into an existing system. Mulcare, Downing, and Smith (1987) modified an existing quadruplex DFCS to incorporate AR. Section 6.3 contains example filters which could be used to add AR to one of the functions, a Stability Augmentation System (SAS), implemented in this modification.

Issues concerned when designing and structuring the RM are:

- The number of success paths.
- The amount of dependency placed on monitoring quality.
- The tradeoff between fail safety and reliability.
- The cost effectiveness for increased reliability with added FD and isolation.
- The throughput constraints imposed by the flight computer. RM can be 60 to 80 percent of the required processing.

Performance qualities may vary for different RM structuring.

In order to maintain fail-safe sensor operations, the following three types of logic are needed.

- Comparison monitoring of dual like signals without an associated DF. Examples where this may be implemented are after a computer failure or a failure of an input to a DF.
- Monitoring the diagnostic error function after failure of one-out-of-two sensors in a dual set. If a pitch rate gyro fails, the relevant DF then monitors the remaining pitch rate gyro.

- Monitoring of a single sensor by its associated DF. This logic requires a failure declaration from both computers to avoid fault indication due to a dual sensor failure.

## 4. ANALYTICAL REDUNDANCY - RELIABILITY

The concept of AR, in order to be implemented in practice, must satisfy performance and reliability requirements of typical flights and must not overload the flight computer.

The traditional RM for a quadruplex set of sensors would isolate the first and second failed sensors by a majority vote. On the failure of the third sensor the set would be shut down. For a perfect monitoring process, the probability of total failure of the set is $4Q_s^3$, where $Q_s$ is the probability of failure of one of the sensors within a certain time. The probability of any two sensors failing is $6Q_s^2$.

A rate gyro has a failure rate of about $10^{-4}$ per hour. This is one of the worst sensor failure rates. The probability of a quadruplex set of gyros failing would be $4 \times 10^{-12}$. This exceeds the typical flight control system requirements per sensor set of $10^{-9}$ by more than two orders of magnitude. For a triplex set of gyros, failure is defined as $3Q_s^2$ with a value of $3 \times 10^{-8}$ which is less than the requirements by a factor of thirty.

The assumption of perfect monitoring is not valid. An additional factor, $Q_m$, should be included in the failure rate. $Q_m$ defines the probability that the first failure is not detected and that both failures are alike. This issue of imperfect monitoring or coverage may cause failure of a quadruplex sensor set after only two failures, if it cannot distinguish between the two good and two failed sensors. The probability of failure here is $6Q_s^2\ Q_m$. No data exists for the term $Q_m$, but numbers on the order of $10^{-2}$ have been suggested. With imperfect monitoring, the probability of failure for a quadruplex set of rate gyros is $6 \times 10^{-10}$ in one hour which is just within the requirements.

The diagnostics provided in an AR system must provide similar quality in monitoring. For triplex sensors configured for dual-fail-operational systems, the total failure rate is given in the following equation.

$$Q_s^3 + 3Q_s^2\ Q_d$$

The first term represents the probability for all three sensors failing. The second term is the combination of the probabilities of a dual sensor failure and the diagnostics failing to recognize the fault, $Q_d$. This is the dominant term. If it is set equal to the reliability requirement,

$$3Q_s^2\ Q_d = 10^{-9}$$

$$Q_d = 3.3 \times 10^{-2}$$

The value of $Q_d$ indicates that the diagnostics must be 97 percent reliable. If this performance is feasible, then AR techniques are at least equal to those of quadruplex channel voting.

This example was devoted to just one set of sensors. Total system reliability includes all hardware: computers, servos, and sensors. A computer code, Computer Aided Redundant System Reliability Analysis (CARSRA), computes system reliabilities based on a specific configuration and sensor failure rates. Mulcare, Downing, and Smith (1987) used a revised version of this code to perform a system reliability assessment.

Table 4-1, utilizing parameters of an existing system (Cunningham and Hartmann, 1977), shows that servos, not sensors, tend to dominate reliability. By adding sensor redundancy, only a factor of two less than baseline system failure rate is obtained. With both sensor and servo redundancy, system failure rate decreased by more than an order of magnitude. Failure rates are dependent on system architecture. A system should be designed to have matched reliability.

TABLE 4-1. SYSTEM RELIABILITY

| Probability (failures per flight hour) | | |
|---|---|---|
| Failure Type | 95% Sensor Redundancy | 95% Sensor/Servo Redundancy |
| Either computer fails (95% effective self-test) | $0.2 \times 10^{-4}$ | $0.2 \times 10^{-4}$ |
| Servo failure in any axis | $6.0 \times 10^{-4}$ | $0.3 \times 10^{-4}$ |
| Gyro failure in any axis | $0.3 \times 10^{-4}$ | $0.3 \times 10^{-4}$ |
| Normal accelerometer failure | $0.02 \times 10^{-4}$ | $0.02 \times 10^{-4}$ |
| Total | $6.52 \times 10^{-4}$ | $0.82 \times 10^{-4}$ |

# 5. ANALYTICAL REDUNDANCY - FAULT DETECTION AND ISOLATION

The type of FD monitor used to test various error signals is critical for performance. Important parameters are speed of response and an acceptable false alarm rate, one per 1000 hours of flight. Observers or DFs can operate with comparison monitors of dual sensors to isolate faults for a fail-operational capability. A fail-safe capability exists for all single sensors and for dual sensors after one failure.

Fault Detection monitors can be multiple trip monitors or a Sequential Likelihood Ratio Test (SLRT) of a defined parameter, error signals, or likelihood functions. The design goal of an FD monitor is FD within a minimum time after the occurrence of the fault while maintaining an acceptable false alarm rate. Another important goal is the number of missed alarms. This is addressed in the literature (Cunningham and Hartmann, 1977 and Deckert, et al., 1978) where data exists from flight simulation tests.

## 5.1. Multiple Trip Monitor

The FD logic in a multiple trip monitor typically requires three consecutive trips for a fault to be declared. The trip thresholds must be based on the performance of each sensor and known sensor and modeling inaccuracies.

The simplest test to check for an error signal is to compare its magnitude with predefined limits. This is illustrated in figure 5.1-1. The trip magnitude is typically placed at some multiple, m, of $\sigma$, the standard deviation of the error signal of an unfailed sensor. The mean of this error signal is zero. The value of m is chosen to yield an acceptable false alarm rate. This is shown in the following sequence of equations. The probability that there will be no false alarms during one hour of flight is given by a Poisson distribution. See Ross (1972) for more information on probability density functions, likelihood functions, and reliability.

$$P(\text{no false alarms}) = e^{-\delta t}$$

where $\delta$ = the false alarm rate (1 per 1000 flight hours)
$t$ = the flight time (1 hour)

so that P(no false alarms) = 0.999. This is also equal to the probability that a sequence of random variables, the sampled error signals, will remain within the trip boundary of $m\sigma$.

$$P\left(\bigcap_{i=1}^{n} X_i\right) = e^{-\delta t}$$

For a one hour flight, sampling at 16 Hz, n will equal 57,600.

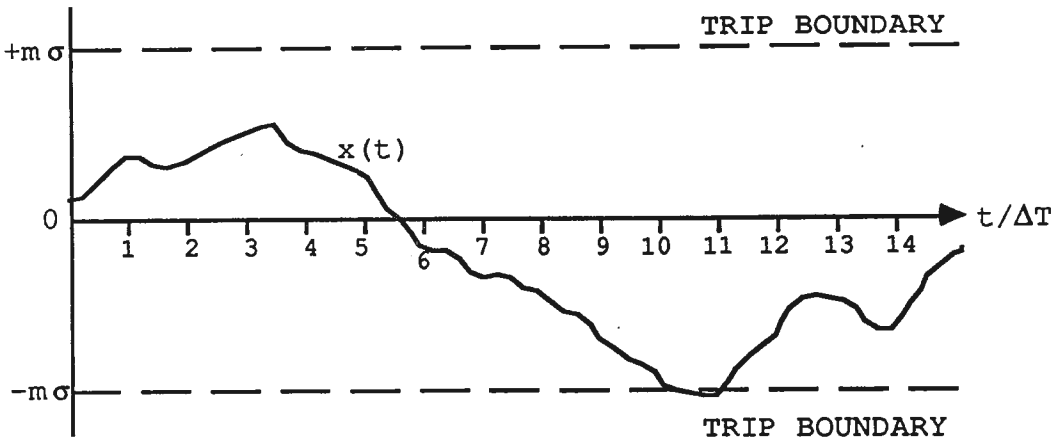



FIGURE 5.1-1. MULTIPLE TRIP FAULT DETECTION MONITOR

If the sequence of random numbers is independent, then

$$P(\bigcap_{i=1}^{n} X_i) = \prod_{i=1}^{n} P(X_i) = nP(X_i)$$

$$P(|X_i| > m\sigma) = (1 - e^{-\delta t}) / n$$

$$= 1.74 \times 10^{-8}$$

This would require m to be much larger than 4. Multiples of $\sigma$ are not given in normal probability tables for numbers in this range. For three consecutive trips,

$$P(|X_i| > m\sigma; \text{ for three consecutive times}) = (1.74 \times 10^{-8})^{1/3}$$

$$= 2.6 \times 10^{-3}$$

This corresponds to $m \approx 3.2$. This multiple of $\sigma$ was determined using a table containing probability values for a standard normal distribution.

For a completely dependent sequence,

$$P(|X_i| > m\sigma;\ \text{for three consecutive times}) \approx P(|X_i| > m\sigma;\ \text{one time})$$

$$\approx 1 - e^{-\delta t}$$

$$= 1.0 \times 10^{-3}$$

This corresponds to $m \approx 3.4$. This value was obtained from the same table used above.

The results of this analysis imply that the triple trip boundary needs to be set at $3.4\sigma$ to meet the false alarm rate for a given error signal.

The trip threshold can remain constant, adjusted based on the output of the sensor, or adjusted based on inputs from the pilot. An adjustable threshold can increase sensitivity to environmental disturbances, such as turbulence, and to aircraft in-flight maneuvering.

A multiple trip criterion (i.e., a fault is declared when a limit is exceeded n consecutive times) gives a much lower false alarm rate than a first trip monitor. The number of consecutive trips chosen for a monitor is a tradeoff between error insensitivity and the speed of FD. Choosing a multiple number of trips applies a built-in delay to FD.

## 5.2. Sequential Likelihood Ratio Tests

Likelihood ratio tests have been demonstrated to be a better FD scheme than the multiple trip monitor in Cunningham and Hartmann (1977). This algorithm provides quicker response to faults such as hardover, and more sensitivity to other smaller faults (bias and scale factor changes). However, there is more complexity to this monitoring scheme.

These tests are based on two hypothesized density functions of a random variable, $f_0(x)$ and $f_1(x)$. The likelihood ratio is expressed as a function, $\Gamma_n$, of error signal observations of that random variable.

$$\Gamma_n = \frac{f_0(e_1, e_2, \ldots, e_n)}{f_1(e_1, e_2, \ldots, e_n)}$$

The two hypothesized outcomes, $H_0$ and $H_1$, are no failure or sensor failure. These outcomes are accepted based on the following rules.

- Accept $H_0$ if $\Gamma_n \leq A$.
- Accept $H_1$ if $\Gamma_n \geq B$.
- No decision if $A < \Gamma_n < B$.

where A = 1/B. A typical value for B is 20,000.

$H_0$ implies that the expected value of a given error signal is 0 and there is no fault. $H_1$ implies the mean of the error signal has shifted unacceptably to $\mu_1$ and a sensor fault should be declared. For actual monitoring, a combined test is used where $\mu_1$ may equal $\pm$ m$\sigma$. Figure 5.2-1 illustrates the bounds for $H_0$ and $H_1$.

This can be defined mathematically using the following equations. The functions, $f_0$ and $f_1$, are assumed to have a normal distribution with a standard deviation of $\sigma$.

$$f_0(e_i) = c \exp[-\sum_{i=1}^{n} (e_i - \mu_0)^2 / 2\sigma^2]$$

$$f_1(e_i) = c \exp[-\sum_{i=1}^{n} (e_i - \mu_1)^2 / 2\sigma^2]$$

Using the no decision criteria, taking natural logarithms, and substituting 0 for $\mu_0$ and $\pm m\sigma$ for $\mu_1$,

$$\ln A < \sum_{i=1}^{n} [(e_i \mp m\sigma)^2 - (e_i)^2] / 2\sigma^2 < \ln B$$

which is equivalent to

$$nm\sigma/2 - \sigma \ln B/m < \sum_{i=1}^{n} e_i < nm\sigma/2 + \sigma \ln B/m, \text{ for } \mu_1 = +m\sigma$$

and

$$-nm\sigma/2 - \sigma \ln B/m < \sum_{i=1}^{n} e_i < \sigma \ln B/m - nm\sigma/2, \text{ for } \mu_1 = -m\sigma$$

A fault is determined when $\left| \sum_{i=1}^{n} e_i \right| \leq \sigma \ln B/m + nm\sigma/2$.

No fault is determined when $\left| \sum_{i=1}^{n} e_i \right| \geq nm\sigma/2 - \sigma \ln B/m$.

Otherwise, no decision is made.

The test sequence begins by initializing n to 1 and updates the summation each cycle. The sequence restarts when $H_0$ is accepted or there is no decision for a fixed period of time. Frequent restarts are desirable because summations over long intervals have less sensitivity to new data.

Likelihood functions of like sensors, i.e., with identical statistical error characteristics, can be compared and used to isolate the faulted sensor. The likelihood functions are $L_{n1}$ and $L_{n2}$ for sensors 1 and 2, respectively. The hypotheses here are that $H_0$ implies that sensor 1 has failed, and $H_1$ implies that sensor 2 has failed. The mean of $H_0$ is $\mu_0$ which is equal to $+m\sigma$, and the mean of $H_1$ is $\mu_1$ which is equal to $-m\sigma$.

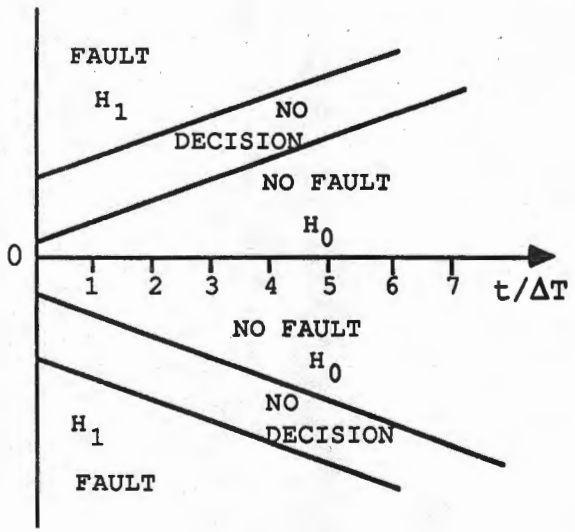



FIGURE 5.2-1. SEQUENTIAL LIKELIHOOD FAULT DETECTION MONITOR

Decisions are made using the difference of the likelihood functions based on the following rules.

$$\Delta L_n = L_{n1} - L_{n2}$$

- Accept $H_0$ if $\Delta L_n \geq \sigma \ln B/2m$.
- Accept $H_1$ if $\Delta L_n \geq \sigma \ln A/2m$.
- No decision if $\sigma \ln A/2m < \Delta L_n < \sigma \ln B/2m$.

Because the no fail hypothesis is not included here, this particular test can proceed only when a fault has already been declared. This test can be initiated after one trip using comparison monitoring. If a fault is declared (three consecutive trips), then the likelihood comparison test is used to determine which sensor is faulted. Because the likelihood functions can only increase in a fault situation, the plus or minus direction of the delta likelihood function provides the information necessary to determine which sensor has failed.

## 5.3. Fault Modeling

Sensor error and fault modeling require careful treatment for FD because typical sensor models are not sufficient and fault characteristics may not be

well-defined. A key issue is modeling sensor characteristics, including noise in an operational environment. Possible errors superimposed on the sensor output include bias, resolution, scale factor, null output, alignment error, hysteresis, environmental effects, response dynamics, and unmodeled sensor inputs. Errors may occur if the model does not include pilot inputs. Mode specific error signals may occur for selected modes, e.g., altitude or heading hold. A determination of the importance of each type of error is dependent on the type and percentage of occurrence of failure modes for each sensor. Modeling these errors may be a difficult task, because data is not easily obtained or does not exist on some of these parameters.

Faults can be classed as open-loop or closed-loop faults. An open-loop fault is one in which the sensor output is completely independent of its input. Examples of this category are hardover (at maximum output), zero output, output stuck at an intermediate value, and a randomly varying output. Closed-loop faults consist of the correct sensor output with errors superimposed. These faults are misalignment, scale factor changes, noise, or others listed in the previous paragraph.

Simulations should subject the FD algorithms to the highest probability faults. Sensor hardover, dead sensor, dynamic response reduction, scale factor changes, and bias shifts represent 90 to 95 percent of expected failures in typical rate gyros and accelerometers. These two sensor types have the highest failure rates compared to other sensors. Other faults should be defined dependent on the sensor.

Some techniques are limited in the type of failures they look for. Deckert, et al., 1978 limited their technique to looking for bias failures. They felt that most other failures could be found with the same technique. As yet, no one has addressed oscillatory faults which may or may not be important.

As defined by Cunningham and Hartmann (1977), the observer/blender and diagnostic Kalman filter worked well for FD for most sensors included in the study. Super-diagnostic techniques were found to have performance concerns and developmental needs. Problems arose in detection of accelerometer soft failure, dynamic response faults, and FI based on more than one filter detecting a given fault. (The difficulty arose when both filters did not detect the failure at the same time.)

A comparison of monitors shows that the SLRT of residual mean values performed well relative to the multiple trip monitor. SLRT caught hardover failures sooner. Frequently scale factor changes would escape the multiple trip monitor, but not the SLRT.

One way to enhance a particular FD algorithm is to utilize the fact that there are physical limitations as to how much the aircraft state can change from one sample to the next. This can minimize the effects of extremely large sensor failures. Desai, Deckert, and Deyst (1979) added a self-test in conjunction with an FD algorithm which provisionally failed a sensor when an abrupt change in output was maintained for three sample intervals.

# 6. SUPPLEMENTAL WORKED EXAMPLES

## 6.1. Altitude Kinematics

Altitude Kinematics refers to the redundancy between changes in altitude measured by the barostatic pressure sensors and changes in altitude computed from the vertical acceleration measured by the accelerometers. The vertical direction is determined via Euler attitude angles: roll ($\phi$) and pitch ($\theta$). Since the accelerometers sense gravity, the gravitational component must be subtracted in computing vertical acceleration. Vertical acceleration is computed as:

$$A_v = A_x \sin\theta - (A_y \sin\phi + A_z \cos\phi) \cos\theta - g$$

where $A_x$, $A_y$, $A_z$ are measured accelerations compensated to the center of mass.

A third order complementary filter (illustrated in figure 6.1-1) is used to store $\dot{h}$ (vertical velocity) prior to a failure detection and possible corruption. The value of $\dot{h}$ can then be used to compare altitude deviation from slightly prior to the failure detection by keeping a moving window of data.

The altitude residues for both sensors would be:

$$\text{Residue \#1} = \Sigma(\text{current } h_1 - \text{previous } h_1 - \dot{h}\, dt)$$

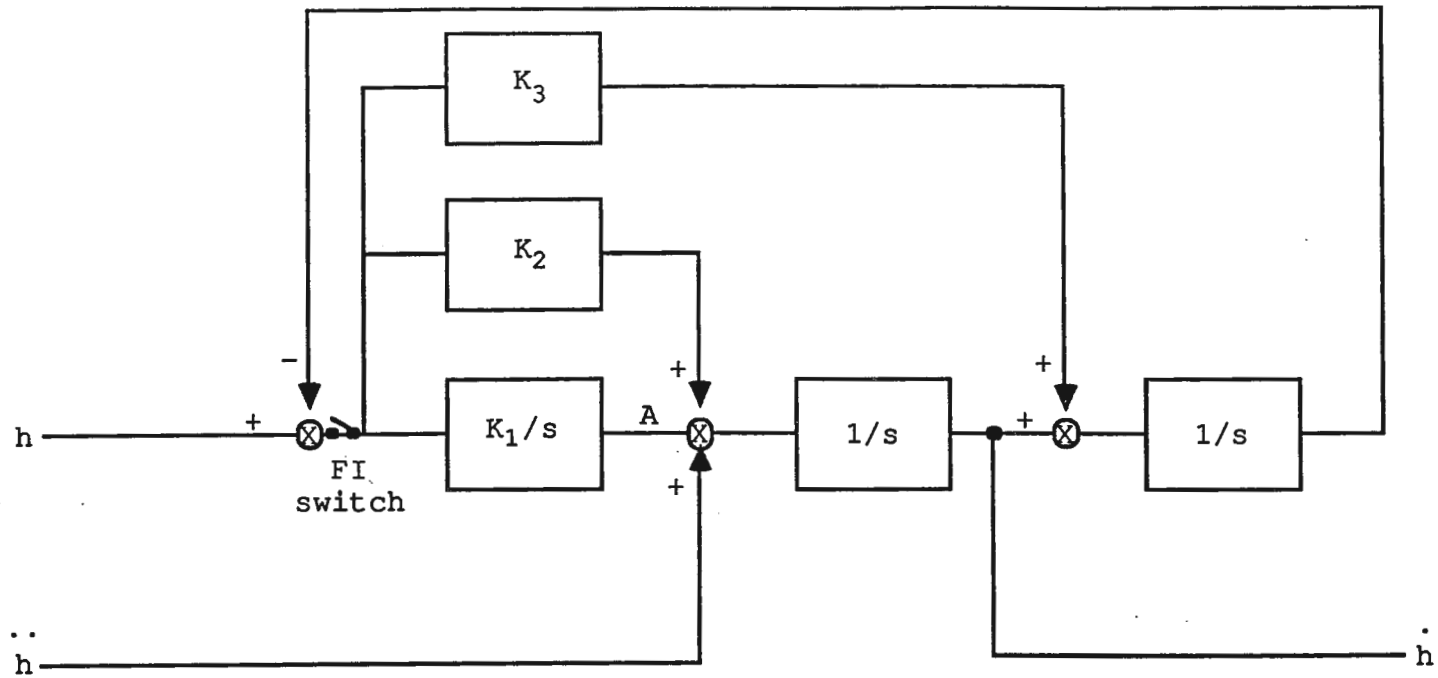
$$\text{Residue \#2} = \Sigma(\text{current } h_2 - \text{previous } h_2 - \dot{h}\, dt)$$

using h measurements from sensors 1 and 2, respectively, where dt is the integration interval.

Both residues are then evaluated by an FI algorithm. The algorithm may be a multiple trip threshold exceedance or a statistical hypothesis algorithm (e.g., SLRT).

During the FI period, the $\dot{h}$ used is only inertially updated via the switch shown in figure 6.1-1, i.e., $\dot{h} = \int \ddot{h}$. The switch opens (FI begins) when the absolute magnitude of $(h_1 - h_2)$ exceeds a threshold N times. The filter is not updated if any sample exceeds the threshold to avoid $\dot{h}$ corruption. Figure 6.1-2 demonstrates the RM processing.

Error sources must be evaluated to avoid a false FI decision. The third order filter has a state to estimate steady-state errors on $\ddot{h}$ to avoid a parabolic error growth when calculating both residues. An error analysis must be made to determine the maximum time that can be allowed for FI.




h is the average of two baro-static pressure altitudes

$\ddot{h}$ is an estimate of error of $\ddot{h}$, subsequently added to each $\ddot{h}$ calculation in the fault isolation residue equations to effectively cancel the error.

FIGURE 6.1-1 THIRD ORDER VERTICAL COMPLEMENTARY FILTER FOR ALTITUDE KINEMATICS

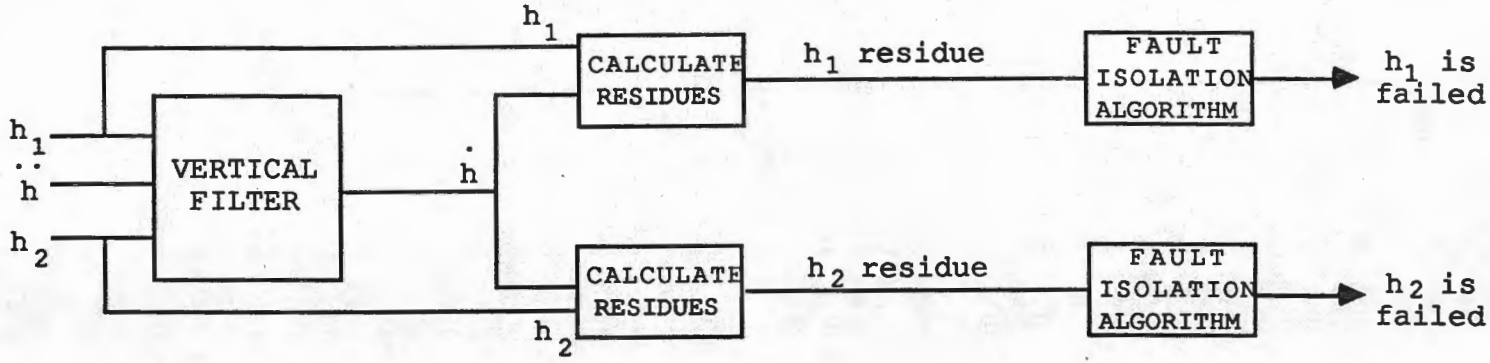



FIGURE 6.1-2. POSSIBLE REDUNDANCY MANAGEMENT FOR ALTITUDE KINEMATICS FAULT ISOLATION

If the time expires without an FI decision, then the following logic can be used:

Is there still a fault?

The answer is yes, if the absolute magnitude of $(h_1 - h_2)$ is greater than the threshold. Then isolate the sensor with the worst error history, i.e., the largest residue magnitude.

Otherwise, the answer is no. Reset the RM logic and restart the vertical filter, closing the FI switch and clearing the FD flag, if one is present.

6.2. Translational Kinematics

A translational kinematic relationship can be used to determine an airspeed sensor fault by means of the related inertial acceleration along the x-axis, such that

$$V_x = \int \dot{U}\, dt \qquad \text{or} \qquad \Delta V_x = \dot{U}\, dt$$

$$V_x = V_t \cos\beta \cos\alpha \approx V_t \qquad \text{for small angles of } \alpha \text{ and } \beta$$

where $V_x$ = x-axis airspeed velocity

$V_t$ = true airspeed

$\alpha$ = angle of attack

$\beta$ = angle of sideslip

$U$ = inertial velocity along the x-axis

An estimate of $\dot{V}_x$ can be obtained using the following equation.

$$\dot{V}_x = A_x - g \sin\theta - QW + RV + \delta_x$$

where $Q$ = pitch rate

$W$ = z-axis velocity

$R$ = yaw rate

$V$ = y-axis velocity

$\delta_x$ = wind acceleration along the x-axis

$A_x$ = acceleration measured by axial accelerometer

Q and R are measured gyro values. V and W can be calculated as follows.

$V = V_t \sin\beta$ = y-axis velocity

$W = V_t \cos\beta \sin\alpha$ = z-axis velocity

The only term unaccounted for is $\delta_x$, wind acceleration. It must be carefully analyzed as an error source in the FI decision. As long as the FI period is not too long relative to this error source, the decision should not be wrong.

Another way to alleviate this error source is to use a moving window to calculate the standard deviation of the airspeed residues (prior to FD) to determine the threshold size for FI. Wind turbulence associated with wind acceleration increases the threshold, thereby protecting from a false FI.

A filter is used to get a good estimate of any steady-state error on $\dot{V}_x$ as shown in figure 6.2-1. The term a is an estimate of error on $\dot{V}_x$, subsequently added to each $V_x$ calculation in the FI residue equations to effectively cancel the error.

$V_{tavg}$ is the average of the two true airspeed sensors. When an FD occurs the filter is no longer updated. FD is accomplished by the means discussed in section 6.1.

The airspeed residues are calculated as:

$$V \text{ Residue \#1} = \Sigma(\text{current } V_{t1} \text{ measured} - \text{previous } V_{t1} \text{ measured} - \dot{V}_x \, dt)$$

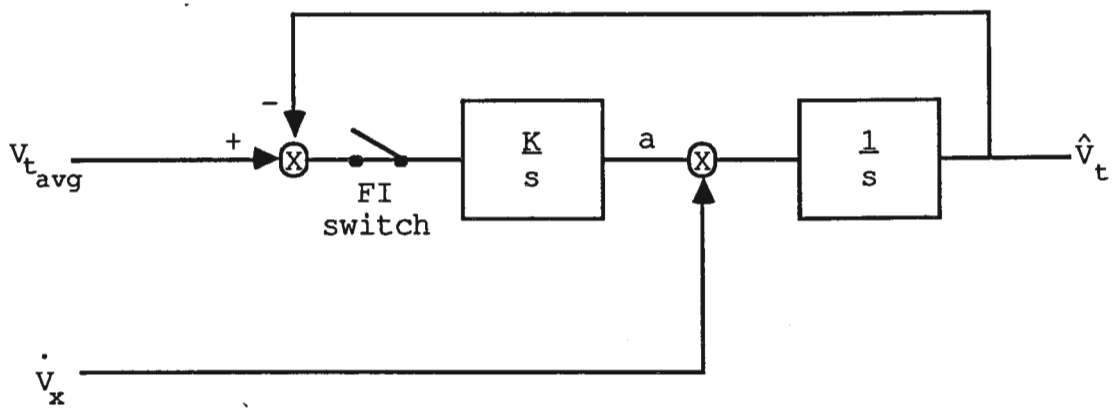
where dt is the update rate. This equation is duplicated for each airspeed sensor measurement. The two residues are then evaluated by an FI algorithm as discussed in section 6.1.

Similar comments concerning error analysis and adaptive threshold solutions apply for this example also. The standard deviation of the average of the sensors could be calculated with a moving window prior to an FD.

6.3. Stability Augmentation System

A large number of advanced aircraft are not statically stable. This is a concern because the airframe is no longer inherently stable. As an aircraft's center of gravity moves aft, the wing has a decreasing capacity to generate a nose down moment in response to an increase in angle of attack. At the neutral point, there is no pitch-axis response due to a change in angle of attack. When the center of gravity is farther aft than the neutral point, a pitch-up moment is generated from an increase in angle of attack which may make the aircraft very difficult to fly.

A Relaxed Static Stability (RSS) aircraft requires an SAS to restore stability to reasonable levels. If the SAS function is critical, it makes sense to implement AR rather than to increase the number of sensors. This function is dependent on feedback from inertial sensors, pitch angle, pitch rate, and vertical acceleration, or air data sensors, true airspeed, and angle of attack, to prevent a rapid pitch-axis divergence.




a is an estimate of error on $\dot{V}_x$, subsequently added to each $\dot{V}_x$ calculation in the fault isolation residue equations to effectively cancel the error.

FIGURE 6.2-1. EXAMPLE FILTER USED FOR TRANSLATIONAL KINEMATICS REDUNDANCY

An estimate of pitch rate and vertical acceleration can be determined with the following equations.

$$Q_E = Q_m - \dot{\theta}\cos\phi_m + \dot{\tau}\sin\phi_m\cos\theta_m$$

$$A_{zE} = A_{zm} - U_m(Q_m - \dot{\alpha}_m) - g\cos\theta_m\cos\phi_m \text{ assuming that } V = 0$$

where $Q$ = pitch rate

$A_z$ = vertical acceleration

$U$ = x-axis velocity

$g$ = acceleration due to gravity

$\phi$ = roll angle

$\theta$ = pitch angle

$\tau$ = yaw angle

$\alpha$ = angle of attack $\approx W/U$ where $W$ is the vertical velocity.

The subscript m denotes measured quantities and the subscript E denotes an estimated value.

Figure 6.3-1 shows one way to implement filters for these two estimated values. FD and isolation is as described in section 6.1.

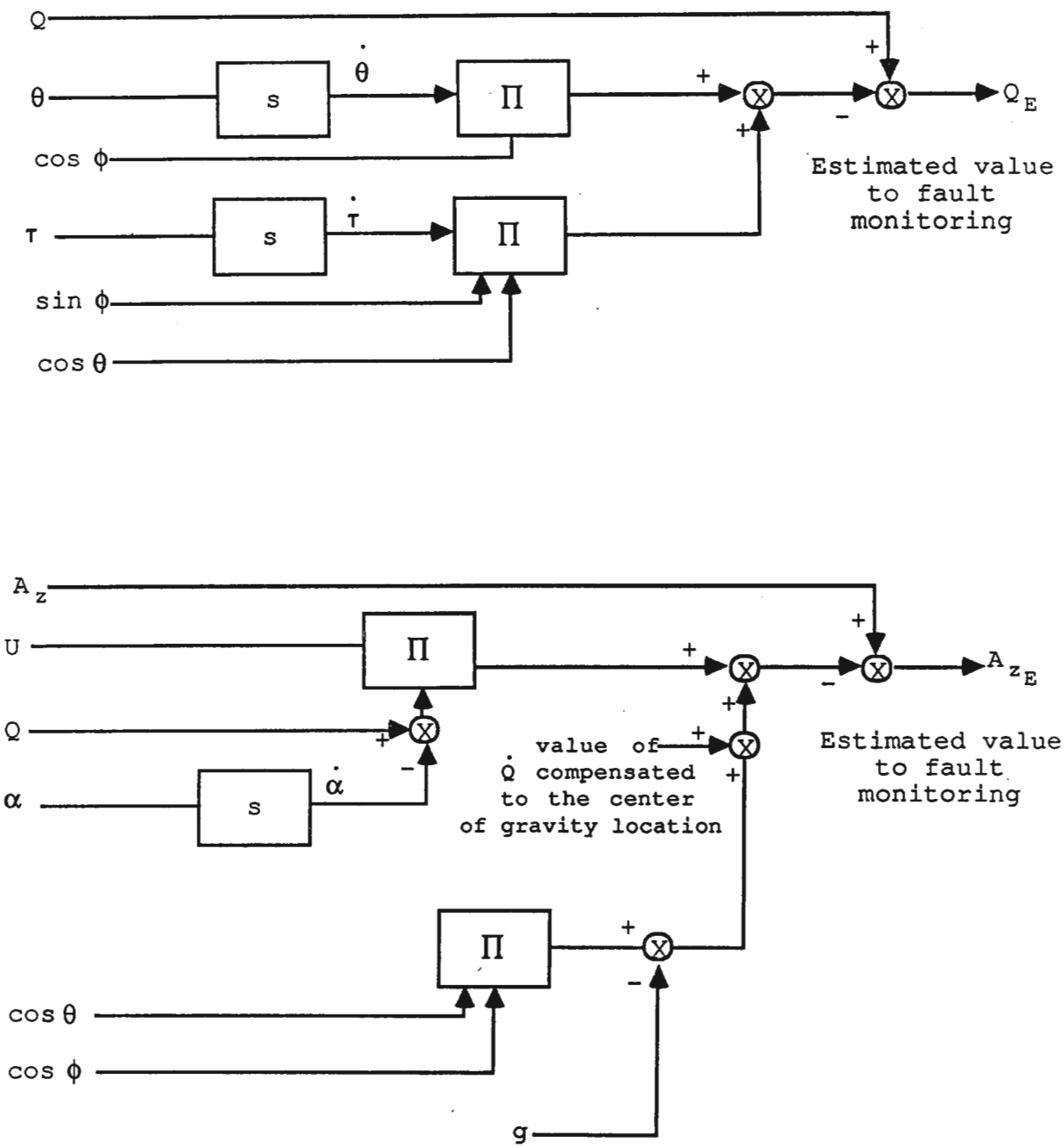



FIGURE 6.3-1. EXAMPLE FILTERS USED FOR STABILITY AUGMENTATION SYSTEM

## BIBLIOGRAPHY


Athans, M. and D. Willner, "A Practical Scheme for Adaptive Aircraft Flight Control Systems," Symposium on Parameter Estimation Techniques and Applications in Aircraft Flight Testing, NASA Flight Research Center, Edwards Air Force Base, April 24-25, 1973.

Beard, R. V., Failure Accommodation in Linear Systems Through Self-Reorganization, Report MVT-71-1, Man-Vehicle Laboratory, Cambridge, MA, February 1973.

Buxbaum, P. J. and R. A. Haddad, "Recursive Optimal Estimation for a Class of Nongaussian Processes," Proceedings of Symposium on Computer Processing in Communications, Polytech Institute of Brooklyn, April 8-10, 1969.

Chien, T. T., An Adaptive Technique for a Redundant-Sensor Navigation System, Report T-560, Draper Labs, Cambridge, MA, February 1972.

Clark, R. N., et al., "Detecting Instrument Malfunctions in Control Systems," IEEE Transactions on Aerospace and Electronic Systems, Vol. AES-11, No. 4, July 1975.

Cunningham, T., and G. Hartmann, Fault Tolerant Digital Flight Control with Analytical Redundancy, Technical Report AFFDL-TR-77-25, May 1977.

Davis, M. H. A., "The Application of Nonlinear Filtering to Fault Detection in Linear Systems," IEEE Transactions on Automatic Control, Vol. AC-20, No. 2, April 1975.

Deckert, J. C., et al., Reliable Dual- Redundant Sensor Failure Detection and Identification for the NASA F-8 DFBW Aircraft, NASA Contractor Report 2944, February 1978.

Desai, M. N., J. C. Deckert, and J.J. Deyst, "Dual Sensor Failure Identification Using Analytical Redundancy," Journal of Guidance and Control, Vol. 2, No. 3, May-June 1979.

Deyst, J. J. and J. C. Deckert, "RCS Jet Failure Identification for the Space Shuttle," Proceedings of IFAC 1975, Cambridge, MA, August 1975.

Hartmann, G. L., et al., Digital Adaptive F-8C Control Laws, Final Report NAS 1-13358, July 1975.

Jones, H. L., Failure Detection in Linear Systems, PhD Thesis, Department of Aeronautics and Astronautics, MIT, Cambridge, MA, September 1973.

Kaniuka, W. W., et al., New Flight Control Technologies for Future Naval Aircraft, Report No. NADC-82240-60, September 1982.

Kerr, T. H., "A Two Ellipsoid Overlap Test for Real-Time Failure Detection and Isolation by Confidence Regions," IEEE Conference on Decision and Control, Phoenix, AZ, November 1974.

Lainiotis, D. G., "Joint Detection, Estimation, and System Identification," Information and Control, Vol. 19, No. 1, August 1971.

Maybeck, P. S., "Failure Detection Through Functional Redundancy," AFFDL-TR-74-3, January 1974.

McAulay, R. J. and E. Denlinger, "A Decision-Directed Adaptive Tracker," IEEE Transactions on Aerospace and Electronic Systems, Vol. AES-9, March 1973.

McGarty, T. P., "State Estimation with Faulty Measurements: An Application of Bayesian Outlier Rejection," Proceedings of the Fifth Symposium on Nonlinear Estimation and Its Applications, San Diego, CA, September 1974.

Mehra, R. K. and J. Peschon, "An Innovations Approach to Fault Detection and Diagnosis in Dynamic Systems," Automatica, Vol. 7, 1971.

Meier, L., D. W. Ross, and M. B. Glaser, "Evaluation of the Feasibility of Using Internal Redundancy to Detect and Isolate On-Board Control Data Instrumentation Failures," AFFDL-TR-70-172, January 1971.

Montgomery, R. C. and A. K. Caglayan, "A Self-Reorganizing Digital Flight Control System for Aircraft," AIAA 12th Aerospace Sciences Meeting, Washington, D.C., January 30-February 1, 1974.

Montgomery, R. C. and D. B. Price, "Management of Analytical Redundancy in Digital Flight Control Systems for Aircraft," AIAA Mechanics and Control of Flight Conference, Anaheim, CA, August 5-9, 1974.

Mulcare, D. B., L. E. Downing, and M. K. Smith, Analytical Sensor Redundancy Assessment, DOT/FAA/CT-86/32, Draft Report, June 1987.

Pierce, B. D. and D. D. Sworder, "Bayes and Minimax Controllers for a Linear System with Stochastic Jump Parameters," IEEE Transactions on Automatic Control, Vol. AC-16, No. 4, August 1971.

Ratner, R. S. and D. G. Luenberger, "Performance-Adaptive Renewal Policies for Linear Systems," IEEE Transactions on Automatic Control, Vol. AC-14, No. 4, August 1969.

Ross, S. M., "Introduction to Probability Models," Academic Press, 1972.

Sanyal, P. and C. N. Shen, "Bayes' Decision Rule for Rapid Detection and Adaptive Estimation Scheme with Space Applications," IEEE Transactions on Automatic Control, Vol. AC-19, June 1974.

Stein, G. and G. L. Hartmann, F-8C Adaptive Control Extensions, NASA Contract NAS 1-13383, June 1976.

Sworder, D. D. and V. G. Robinson, "Feedback Regulators for Jump Parameter Systems with State and Control Dependent Transition Rates," IEEE Transactions on Automatic Control, Vol. AC-18, No. 4, August 1973.

Sworder, D. D., "Bayes' Controllers with Memory for a Linear System with Jump Parameters," IEEE Transactions on Automatic Control, Vol. AC-17, February 1972.

Weinstein, W., Feasibility and Design Studies of an Integrated Sensory Subsystem (ISS) for Advanced V/STOL Aircraft, Report No. NADC-76259-30, March 1978.

Willsky, A. S., J. J. Deyst, and B. S. Crawford, "Adaptive Filtering and Self-Test Methods for Failure Detection and Compensation," Proceedings of the 1974 JACC, Austin, Texas, June 19-21, 1974.

Willsky, A. S. and H. L. Jones, "A Generalized Likelihood Ratio Approach to State Estimation in Linear Systems Subject to Abrupt Changes," Proceedings of the 1974 Conference on Decision and Control, Phoenix, AZ, November 1974.

## GLOSSARY

ANALYTICAL REDUNDANCY. The use of software algorithms which use known mathematical relationships between different sensors for sensor failure detection and replace most of additional redundant sensor hardware.

CONTROL LAW. The physical relationship between various sensors and control surfaces.

COVERAGE. The percent confidence level of a given analytical redundancy fault detection and isolation algorithm for all types of faults.

DIAGNOSTIC FILTER. An analytical algorithm which processes data from N functionally related sensors. The data are used to estimate some sensor outputs and assess the correct functioning of the sensors.

DUAL FAIL-OPERATIONAL. A reliability requirement placed on a system which requires the system to be operational after two failures have occurred.

FAIL-OPERATIONAL. A reliability requirement placed on a system which requires the system to be operational after a single failure has occurred.

FAIL-SAFE. A reliability requirement placed on a system which requires that safe flight not be hindered even after a failure.

FALSE ALARM. The declaration of a fault by a fault detection monitor or algorithm when there is no fault.

FAULT DETECTION. The determination that a sensor is faulted by using a software algorithm.

FAULT ISOLATION. The determination that a particular sensor is faulted by using a software algorithm.

FAULT TOLERANCE. Accommodation of sensor hardware faults based on some type of comparator scheme.

FLIGHT-CRITICAL. A description of functions whose failure would contribute to or cause a failure condition preventing the continued safe flight and landing of the aircraft.

MISSED ALARM. The failure of a fault detection monitor or algorithm to detect a fault when there is a sensor fault.

MULTIPLE TRIP MONITOR. A fault detection algorithm which declares a fault after the sensor output has exceeded a predefined threshold N times.

OBSERVER. An algorithm which models physical relationships between sensor data and uses the data to provide fault detection for one or more sensors. This is also known as a Luenberger observer or a signal blender.

REDUNDANCY MANAGEMENT. The computer processing which is needed to implement fault detection and isolation algorithms.

REVERSION MODE. The high level of redundancy in a system having different redundancies requirements for some sensors. Critical sensors may have a high level of redundancy while other sensors have low levels.

SENSOR. An instrument which measures a particular physical parameter. The data output may be digital or analog and is utilized by the flight computer.

SEQUENTIAL LIKELIHOOD RATIO TEST. A fault detection algorithm which is based on two hypothesized density functions of no fault or sensor fault.

SEQUENTIAL PROBABILITY RATIO TEST. See sequential likelihood ratio test.

SUPER-DIAGNOSTIC FILTER. An algorithm which provides all the capabilities of a diagnostic filter. Additionally, it can isolate a specific faulted sensor. At the current time, this is the most complex technique used to implement analytical redundancy.

## ACRONYMS

| | |
|---|---|
| AFFDL | Air Force Flight Dynamics Laboratory |
| AK | Altitude Kinematics |
| AR | Analytical Redundancy |
| C | Comparator |
| CARSRA | Computer Aided Redundant System Reliability Analysis |
| DF | Diagnostic Filter |
| DFC | Digital Flight Control |
| DFCS | Digital Flight Control System |
| DOT | Department of Transportation |
| FAA | Federal Aviation Administration |
| FBW | Fly-By-Wire |
| FD | Fault Detection |
| FI | Fault Isolation |
| NADC | Naval Air Development Center |
| NASA | National Aeronautics and Space Administration |
| RM | Redundancy Management |
| RSS | Relaxed Static Stability |
| SAS | Stability Augmentation System |
| SDF | Super-Diagnostic Filter |
| SLRT | Sequential Likelihood Ratio Test |
| SPRT | Sequential Probability Ratio Test |

U.S. Department of Transportation
**Federal Aviation Administration**

DOT/FAA/CT-88/10

# HANDBOOK-VOLUME II
# DIGITAL SYSTEMS VALIDATION

## CHAPTER 8
## ESTIMATION AND MODELING FOR REAL-TIME SOFTWARE RELIABILITY MODELS

**PREPARED BY:**

**COMPUTER RESOURCE MANAGEMENT, INC.**
**950 HERNDON PARKWAY, SUITE 360**
**HERNDON, VIRGINIA 22070**

**PREPARED FOR:**

**FEDERAL AVIATION ADMINISTRATION**
**TECHNICAL CENTER**
**ATLANTIC CITY INTERNATIONAL AIRPORT, NEW JERSEY 08405**

NOTICE

This document is disseminated under the sponsorship of the U.S. Department of Transportation in the interest of information exchange. The United States Government assumes no liability for the contents or use thereof.

The United States Government does not endorse products or manufacturers. Trade or manufacturers' names appear herein solely because they are considered essential to the objective of this report.

TABLE OF CONTENTS


| Section | | | Page |
|---|---|---|---|
| 1. | INTRODUCTION | | 8-1 |
| | 1.1. What is Software Reliability | | 8-1 |
| | 1.2. The Scope of this Tutorial | | 8-1 |
| 2. | TOPIC HISTORY | | 8-3 |
| | 2.1. Introduction | | 8-3 |
| | 2.2. Achieving System Reliability Estimates | | 8-5 |
| 3. | THE SOFTWARE DEVELOPMENT PROCESS | | 8-9 |
| | 3.1. Relationship Between the Software Development Process and Software Reliability Estimation | | 8-9 |
| | 3.2. Software Testing | | 8-11 |
| 4. | FAULT-INTOLERANT RELIABILITY MODELS | | 8-13 |
| | 4.1. Introduction | | 8-13 |
| | 4.1.1. Step 1: | Evaluating the Failure Data Acquired From Testing | 8-13 |
| | 4.1.2. Step 2: | Curve Fitting | 8-16 |
| | 4.1.3. Step 3: | Model Selection | 8-19 |
| | 4.1.4. Step 4: | Statistical Computation to Obtain Parameter Estimates (from Goel, 1983) | 8-19 |
| | 4.1.5. Step 5: | Computation of Performance Statistics | 8-25 |
| | 4.1.6. Step 6: | Application of Predicted Fault Rates | 8-25 |
| | 4.2. Computation of a Time-Between-Failures Model | | 8-27 |
| | 4.2.1. Step 3: | Model Selection | 8-27 |
| | 4.2.2. Step 4: | Statistical Computation to Obtain Parameter Estimates | 8-27 |
| | 4.2.3. Step 5: | Computation of Performance Statistics | 8-30 |
| 5. | FAULT-TOLERANT SOFTWARE | | 8-33 |
| | 5.1. Introduction | | 8-33 |
| | 5.2. Principles of Fault Tolerance | | 8-33 |
| | 5.3. Fault-tolerant Software Reliability Models | | 8-34 |
| | 5.4. A Hierarchical Software Reliability Model | | 8-35 |
| | 5.4.1. Software Components | | 8-35 |
| | 5.4.2. Inputs to the Software Reliability Model | | 8-38 |

TABLE OF CONTENTS


| Section | | | Page |
|---|---|---|---|
| | 5.4.3. | Outputs from the Software Reliability Model | 8-38 |
| | 5.4.4. | Model Representation | 8-38 |
| | 5.4.5. | The Modeling Process | 8-38 |
| | 5.4.6. | Detailed Function Blocks | 8-40 |
| | 5.4.7. | Function Block States | 8-42 |
| | 5.4.8. | System Reliability | 8-44 |
| | 5.4.9. | General Equations | 8-44 |
| | 5.4.10. | N-version Software Calculations | 8-44 |
| | 5.4.11. | Recovery Block Calculations | 8-45 |
| | 5.4.12. | Examples of Potential Models: Simple, High-Level Model Example | 8-45 |
| | 5.4.13. | Simple, Detailed Level Model Examples | 8-47 |
| | 5.4.14. | Complex, High-Level Model Example | 8-49 |
| | 5.4.15. | Detailed Worked Examples | 8-51 |
| | 5.4.16. | Accuracy Constraints | 8-52 |
| | 5.4.17. | Accuracy | 8-53 |
| | 5.4.18. | Accuracy of the Hybrid N-version Software | 8-53 |
| | 5.4.19. | Accuracy of the Recovery Block | 8-53 |
| | 5.4.20. | Safety | 8-53 |
| | 5.4.21. | Availability | 8-54 |
| | 5.4.22. | Response to Undesired Events | 8-55 |
| | 5.4.23. | Assumptions | 8-56 |
| 5.5. | Scott et al.'s Model | | 8-57 |
| | 5.5.1. | Introduction | 8-57 |
| | 5.5.2. | Recovery Block | 8-57 |
| | 5.5.3. | The Recovery Block Reliability Model | 8-58 |
| | 5.5.4. | N-version Programming | 8-59 |
| | 5.5.5. | The N-version Programming Reliability Model | 8-60 |
| | 5.5.6. | Consensus Recovery Block | 8-60 |
| | 5.5.7. | The Consensus Recovery Block Reliability Model | 8-61 |
| | 5.5.8. | Verification | 8-62 |

APPENDICES

| | | |
|---|---|---|
| A - | ASSUMPTIONS UNIQUE TO FAULT COUNT SOFTWARE RELIABILITY MODELS | 8-63 |
| B - | ASSUMPTIONS UNIQUE TO TIME-BETWEEN-FAILURE SOFTWARE RELIABILITY MODELS | 8-67 |
| C - | WORKED EXAMPLE FOR THE GOEL-OKUMOTO ON-HOMOGENEOUS POISSON PROCESS MODEL | 8-69 |
| D - | WORKED EXAMPLE SHOWING COMPUTATIONS FOR THE JELINSKI-MORANDA DE-EUTROPHICATION MODEL | 8-79 |
| E - | WORKED EXAMPLES FOR THE N-VERSION SOFTWARE RELIABILITY DESIGN TECHNIQUES | 8-87 |

TABLE OF CONTENTS


| Section | Page |
|---|---|
| F - WORKED EXAMPLES FOR THE RECOVERY BLOCK SOFTWARE RELIABILITY DESIGN TECHNIQUES | 8-99 |
| BIBLIOGRAPHY | 8-113 |
| GLOSSARY | 8-115 |
| ACRONYMS AND ABBREVIATIONS | 8-116 |

LIST OF ILLUSTRATIONS


| Figure | | Page |
|---|---|---|
| 2.1-1 | HARDWARE AND SOFTWARE COST TRENDS | 8-4 |
| 3.1-1 | PHASES OF SOFTWARE DEVELOPMENT | 8-10 |
| 4.1-1 | FLOW FOR COMPLETING THE SOFTWARE RELIABILITY ESTIMATION PROCESS | 8-14 |
| 4.1-2 | OBSERVED FAILURES PER HOUR | 8-16 |
| 4.1-3 | ACTUAL VERSUS PREDICTED FAILURES PER HOUR | 8-18 |
| 4.1-4 | PREDICTED VERSUS ACTUAL NUMBER OF REMAINING FAULTS | 8-26 |
| 5.4-1 | SIMPLE BLOCK DIAGRAM EXAMPLE | 8-41 |
| 5.4-2 | OVERALL REPRESENTATION OF THE FUNCTIONAL CONSTRUCTS | 8-41 |
| 5.4-3 | SIMPLE BLOCK DIAGRAM EXAMPLE | 8-46 |
| 5.4-4 | DETAILED DIAGRAM FOR THE SINGLE VERSION SOFTWARE FUNCTION BLOCK OR THE DECISION ALGORITHM FUNCTION BLOCK | 8-47 |
| 5.4-5 | EQUIVALENT DIAGRAM INDICATING THE FUNCTIONAL STRUCTURE TO BE USED IN THE DETAILED DIAGRAM FOR A SINGLE VERSION SOFTWARE FUNCTION BLOCK | 8-48 |
| 5.4-6 | COMPLEX, HIGH LEVEL MODEL EXAMPLE | 8-50 |
| E-1 | GENERAL FORMAT FOR N-VERSION SOFTWARE | 8-87 |
| E-2 | N-VERSION SOFTWARE WITH ACCEPTANCE TESTS | 8-91 |
| E-3 | N-VERSION SOFTWARE IN WHICH THE OUTPUTS ARE SUBJECTED TO AN ACCEPTANCE TEST IF THE DECISION ALGORITHM FAILS | 8-94 |
| F-1 | GENERAL FORMAT OF A BACKWARD RECOVERY BLOCK | 8-99 |
| F-2 | GENERAL FORMAT OF A FORWARD RECOVERY BLOCK | 8-102 |
| F-3 | ALTERNATE FORMAT FOR A FORWARD RECOVERY BLOCK | 8-108 |

LIST OF TABLES


| Table | | Page |
|---|---|---|
| 2.2-1 | COMPARISON OF HARDWARE AND SOFTWARE PROPERTIES | 8-6 |
| 4.1-1 | FAILURES IN ONE HOUR (EXECUTION TIME) INTERVALS AND CUMULATIVE FAILURES | 8-15 |
| 4.1-2 | FAULT-INTOLERANT SOFTWARE RELIABILITY MODEL ASSUMPTIONS | 8-17 |
| 4.1-3 | FAULT-INTOLERANT SOFTWARE RELIABILITY MODELS | 8-19 |
| 4.1-4 | ASSUMPTIONS FOR THE GOEL-OKUMOTO NON-HOMOGENEOUS POISSON PROCESS MODEL | 8-20 |
| 4.1-5 | PERFORMANCE MEASURES FOR TIME-BETWEEN-FAILURE MODELS | 8-21 |
| 4.1-6 | PERFORMANCE MEASURES FOR FAULT-COUNT MODELS | 8-22 |
| 4.2-1 | ASSUMPTIONS FOR THE JELINSKI-MORANDA DE-EUTROPHICATION MODEL | 8-28 |
| 5.4-1 | RELIABILITY MODEL MENU SELECTIONS | 8-39 |
| 5.4-2 | CORRELATION OF FAULTS WITH THE DETAILED PORTIONS OF THE SINGLE VERSION SOFTWARE FUNCTION BLOCK | 8-42 |
| 5.4-3 | CORRELATION OF FAULTS WITH THE DETAILED PORTIONS OF THE DECISION ALGORITHM FUNCTION BLOCK | 8-43 |
| 5.4-4 | TRUTH TABLE OF FUNCTION BLOCK STATES | 8-43 |
| E-1 | RELIABILITY VALUES FOR THE SOFTWARE COMPONENTS IN FIGURE | 8-88 |
| E-2 | RELIABILITY VALUES FOR THE SOFTWARE COMPONENTS IN FIGURE E-2 | 8-91 |
| E-3 | RELIABILITY VALUES FOR THE SOFTWARE COMPONENTS IN FIGURE E-3 | 8-94 |
| F-1 | RELIABILITY VALUES FOR THE SOFTWARE COMPONENTS IN FIGURE F-1 | 8-100 |
| F-2 | RELIABILITY VALUES FOR THE SOFTWARE COMPONENTS IN FIGURE F-2 | 8-102 |
| F-3 | ACCURACY EFFECTS WHEN FEWER THAN n ALTERNATES ARE USED | 8-107 |
| F-4 | RELIABILITY VALUES FOR THE SOFTWARE COMPONENTS IN FIGURE F-3 | 8-108 |
| F-5 | COMPARISON OF RELIABILITY VALUES WHEN FEWER THAN n ALTERNATES ARE USED | 8-111 |

# 1. INTRODUCTION

## 1.1. What is Software Reliability

Industry concern for software reliability is growing as increasingly complicated computer software is being used to perform critical tasks. Software reliability is "the probability that a given software program operates for some time period, without a software error, on the machine for which it was designed given that it is used within design limits." (Dickson et al., cited by Gephart et al., 1978, p. 2). A number of methods have been proposed for improving software quality and predicting its reliability. Until recently, software developers assumed that the testing strategy selected would be sufficient to detect all faults or failures in the software prior to its implementation. However, rapid increases in software complexity have made it impossible to thoroughly test the software within the constraints of reasonable time and budget.

There are two possible approaches to the problem of software reliability. The first approach is to use reliability models that predict the number of software faults expected and the likelihood that these faults will cause a system failure. Such predictions are based upon data acquired through initial testing of the software. This information can then be used to guide the testing process and to provide a reliability confidence level. The objective of this approach is to find and remove all faults in the software. Fault-intolerant (or single-version software) is software to which these models can be applied.

A second approach assumes that software faults are likely to occur. Finding each fault, however, may not be possible. Rather than attempting to remove all faults, this approach tries to develop safeguards within the software that will support recovery from software failures or at least minimize the consequences of such failures. Reliability models have been developed in response to such fault-tolerant software. These models are used to predict the reliability of this type of software and to determine the types and number of safeguards that need to be included in a given type of software. Fault-tolerant reliability models are used even before the software has been coded. Consequently, results from the application of these models can guide the subsequent design and development of the actual software. This tutorial looks at both fault-tolerant and fault-intolerant reliability models that are appropriate for use with complex real-time software such as that used in avionics and flight control systems.

## 1.2. The Scope of this Tutorial

This tutorial provides aircraft certification specialists with the background needed to understand and evaluate documents that describe measures taken to assess software reliability submitted as part of the certification process for advanced digital avionics and flight control systems. It should be noted, however, that the development of software reliability methodologies is still in

its infancy. As a result, there is little agreement as to how and when these models should be applied, and which models are most appropriate for specific use situations. The reader who seeks hard and fast rules will be disappointed to discover that such rules simply do not exist. Consequently, this tutorial can only present the concept of reliability models, define their processes and goals, and describe those models which are considered acceptable and likely to be applied.

The unquestionable need for such models necessitates their use even though questions of theory and application remain. Specific guidelines and recommendations which can be directly applied to evaluating certification documents remains a future objective.

This tutorial is divided into five major sections:

- "Introduction" and "Topic History" provide overviews of software reliability: Why it is used and what it tries to accomplish. Two broad categories of software reliability models are also introduced.

- "The Software Development Process" briefly describes the stages involved in software development, the opportunities for faults to be introduced into the software, and limitations in current testing approaches intended to remove faults. This section is intended to serve as a framework for understanding when reliability models are used and how they compare with some existing approaches to supporting high software quality.

- "Fault-intolerant Software Reliability Models" looks at reliability models which are used with real-time, single-version or fault-intolerant software.

- "Fault-tolerant Reliability Models" addresses reliability models designed specifically for use with fault-tolerant software.

- The appendices contain worked examples demonstrating how the most important models (representing both the fault-tolerant and fault-intolerant categories) are used. Following the appendices are supplemental materials, including references, which can serve as a basis for further study in this area, and a glossary of terms.

# 2. TOPIC HISTORY

## 2.1. Introduction

Traditionally, improvements in aircraft design have been aimed at more efficient aerodynamic designs and propulsion systems (Larsen, et al., 1984). Recently, however, the greatest advancements have been in the development of digital electronic systems and devices for flight control and avionics systems. Examples of such systems include stability and control augmentation systems, active control systems, advanced displays, aircraft/crew interface, and aircraft handling and gust-load alleviation systems. Such systems rely on extensive software to monitor their functioning and maintain appropriate system conditions. With this growing emphasis on critical software functioning comes the requirement for methods to evaluate the reliability of this software.

Although reliable software is important for all applications, it is imperative for flight-critical software. Since in-use software testing of flight-critical systems is not a safe option, other methods must be implemented to demonstrate that the software can be relied upon to perform its intended function.

Hardware reliability assessment has been in use for more than 20 years. Hardware reliability models and procedures have proved useful and have achieved acceptance as validation tools. Although software reliability assessment has been around almost as long - 15 years - acceptance within the industry has been slow. Acceptance and implementation are expected to improve for many reasons, including:

- The recognition of a growing reliance on software to perform critical tasks. (A trend that is likely to continue.)
- The increasing use of software to perform tasks traditionally performed by hardware.
- The increased software development and operational costs compared to hardware costs.

Computer hardware costs are closely linked to technological advances in integrated circuit technology. Hardware advances pose a challenge to the software designer to develop software that will effectively utilize complex, innovative hardware. The complexity of such software has outgrown the technology for designing and testing it. Workable reliability assessment methods are therefore critical.

As figure 2.1-1 (Shooman, 1984) shows, hardware manufacturing costs have been decreasing over the past 30 years as hardware technology advances. Software costs, however, continue to increase.

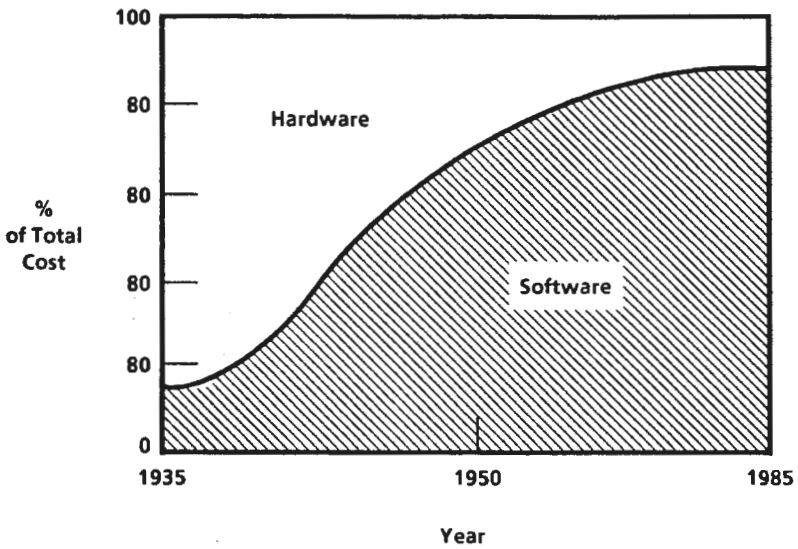



FIGURE 2.1-1. HARDWARE AND SOFTWARE COST TRENDS (from Shooman, 1984)

Software costs are roughly proportional to the number of lines of code in the program. Consequently, greater software complexity is accompanied by higher labor costs. Greater complexity also increases the likelihood of errors being made in the design and coding of software. Not surprisingly, the bulk of software development costs are devoted to testing and validation, a cost which also grows with increases in software complexity.

Concerns with software reliability can be traced back to two pressures: The need to reduce development costs, with the testing and validation stage as a major target; and the increasing likelihood of errors due to more complex software.

Attempts to reduce software errors conflict with attempts to reduce software costs. Obvious ways of reducing errors include devoting more time to the early stages of software design, such as development of software specifications and requirements (to enhance software quality) and thorough testing of the software. Both approaches are labor intensive and extremely costly, and can have a major impact on software development time. Therefore, there is a need to find other methods which promise effective but less expensive solutions.

Two approaches to offering more cost-effective quality control are described in this tutorial. The first approach is single-version software reliability modeling. The second approach is the development of fault-tolerant software.

- Single-version software reliability modeling serves two important functions (Goel, 1983). The modeling provides (1) a useful decision tool for software development planning, and (2) some assurance of how well the software can be expected to perform over time.

Based upon the results of these models, educated decisions can be made concerning how staff time and other costs can be allocated to produce the highest quality software. Predictions of the number of remaining faults enable software developers to determine the additional costs required for incremental enhancements in software reliability.

The cost of finding remaining faults rises very quickly. As faults are removed, remaining faults become more difficult to locate. At some point removing additional faults is no longer cost effective. Justification for decisions about reasonable reliability levels can be based, in part, on objective measures (i.e., performance parameters) provided by software reliability models.

- Fault-tolerant software has safeguards built into the software to enable it to detect and recover from faults, avoiding system failures or minimizing their impact.

  Examples of these safeguards include N-version Software (NVS) and Recovery Block (RB) methodology. NVS refers to incorporating multiple versions of the same software module, which have been developed by different coders, into the software program. When one version fails, a second version can be tried. The process continues until a version is successfully run.

  The RB method refers to providing components which may be alternated with a faulty component. Software reliability models have also been developed for fault-tolerant software. Such models provide the same benefits as single-version reliability models. Results of these models can be used for software development planning and assurance of software reliability.

## 2.2. Achieving System Reliability Estimates

Software reliability models attempt to achieve the same objectives as their hardware reliability counterparts: To provide reliability estimates that can contribute to estimation of overall system reliability. Initial attempts at software reliability modeling were based upon hardware reliability constructs. However, inherent differences between hardware and software have resulted in software reliability models assuming a somewhat different approach. This is especially true for single-version, fault-intolerant software models.

Hardware reliability models are developed on the basis of known failure rates for the components or modules comprising the system. These failure rates can be estimated using known values from Department of Defense (DOD) MIL-HDBK-217 or relying on past experience with similar hardware components. In the case of software reliability estimation, there are no failure rate standards. Although software is constructed by integrating existing pieces or modules into a newly integrated package, fault data do not exist for these entities.

There are other differences between software and hardware which prevent the simple application of hardware models to software estimation. A major difference lies in the source of faults. In the case of hardware, faults are introduced during the manufacturing stage, when each individual module is built. Each time a module is manufactured, there is a possibility that faults in the parts or

parts or entire assemblage may result. Faults may also appear during use as the module ages and wears. In contrast, software faults occur primarily during the design and development process. Copying the software does not produce new faults or wear out the software.

The primary consequence of these differences can be seen in failure rate curves viewed over the life cycle of the system. Hardware failures will start high and then drop as the faults are fixed. The failure rate will then remain constant until modules begin to wear out, at which time the failure rate will begin to increase again. Software, in contrast, will start high and drop, achieving a similar constant failure rate as is found in hardware. However, there will not be a subsequent increase over time since the software does not wear out. Consequently, software exhibits reliability growth. Other differences between hardware and software are summarized in table 2.2-1 (Krauson and Baker, 1982).

TABLE 2.2-1. COMPARISON OF HARDWARE AND SOFTWARE PROPERTIES (from Krauson and Baker, 1982)

| Property | Characteristic | |
|---|---|---|
| | Hardware | Software |
| Ratio of Errors | Errors cause a relatively large (60% to 80%) portion of system failures | Errors cause a relatively small (20% to 40%) but significant portion of system failures |
| Parts Interchangeability | Many components are identical | Each module has a unique function and implementation |
| Fault Replication | Different copies of identical components have different defects | All copies of the same master have the same faults |
| Usage | Usage causes wear defects | Usage has no effect on faults |

In the past, the inability to apply hardware reliability models to software has resulted in the assumption that software reliability has a value of 1.0. This value was then used to compute system reliability. In recent years, the increasingly critical role of software in overall system performance has made

this assumption untenable. Software reliability must be either obtained from exhaustive testing or estimated, rather than assumed, in order to properly assess system reliability.

These differences have affected the development of fault-intolerant and fault-tolerant software reliability models in unique ways. Assessment of single-version, fault-intolerant software reliability involves predicting the expected number of faults in the software as a whole. Since there are no a priori standards for expected number of faults, as are available for hardware, an initial testing phase must take place. Based upon the number of faults detected during testing, statistical methods can then be used to predict the fault distribution and estimate the remaining number of errors. Testing can then continue, if required, until remaining faults are found and corrected.

Fault-tolerant reliability methods are more similar to hardware methods. Within fault-tolerant software systems are individual software modules, such as each of the multiple versions of a given software module (NVS), RBs, and other "pieces." These pieces are conceptually similar to individual hardware modules. The similarity ends here in that currently there are no fault standards for these individual software modules. Rather, reliability for these modules must be estimated using testing and other procedures. Ideally, a database, which will include module reliability data, will be compiled over the next five to ten years as more experience is gained using these techniques. Overall software reliability is then determined through combining individual module reliabilities, as is the case with hardware.

# 3. THE SOFTWARE DEVELOPMENT PROCESS

## 3.1. Relationship Between the Software Development Process and Software Reliability Estimation

Gephart et al., 1978 have developed a generic model of software error rate as a function of software life cycle. Figure 3.1-1 (Gephart et al., 1978) shows how error rate varies with the phase of the software development cycle. Phase I, the design and initial testing period, involves designing equations and algorithms; preparing flowcharts; and writing, testing, and integrating individual software modules. Types of errors likely to occur during this phase include typographical errors, syntax errors, and logic errors. Typically, initial system integration testing is performed at the end of this phase.

"Advanced" testing, involving the program as a whole, takes place during the second phase, and includes subjecting the software to inputs thought to be typical of the use environment. By the end of this phase, the software is actually fielded. Phase III, an operational phase, involves only limited maintenance. Two types of errors will be removed in Phase III: (1) Those which have critical impact on system performance, and (2) Those whose removal is cost-effective. Phase IV involves operational use and, typically, no maintenance is performed.

Because Phase I is the design phase, the error rate is likely to be quite high and perhaps erratic. As the software becomes more structured and integrated, and extensive testing is performed, error rates will become more predictable and show a declining rate during Phase II as testing and debugging efforts continue. During Phase III, the error rate will continue to decline, eventually leveling off. Typically, error data used for reliability assessment are acquired during Phases II and III, allowing identification of error distributions and predicting the expected number of errors. The accuracy of reliability assessment depends, not surprisingly, on the quality of the error data obtained during these phases.

Since most of the errors have been identified and corrected during the first three phases, Phase IV errors tend to be infrequent and occur as a function of using a previously untested path as a result of an unanticipated situation. These errors are difficult if not impossible to find and correct. Some errors are never corrected, because they cannot be found within the constraints of reasonable time and cost. Also, an error may not always be successfully corrected, and the process of correcting one error may introduce new errors. Therefore, the error rate in Phase IV tends to be constant and mostly nonexistent.

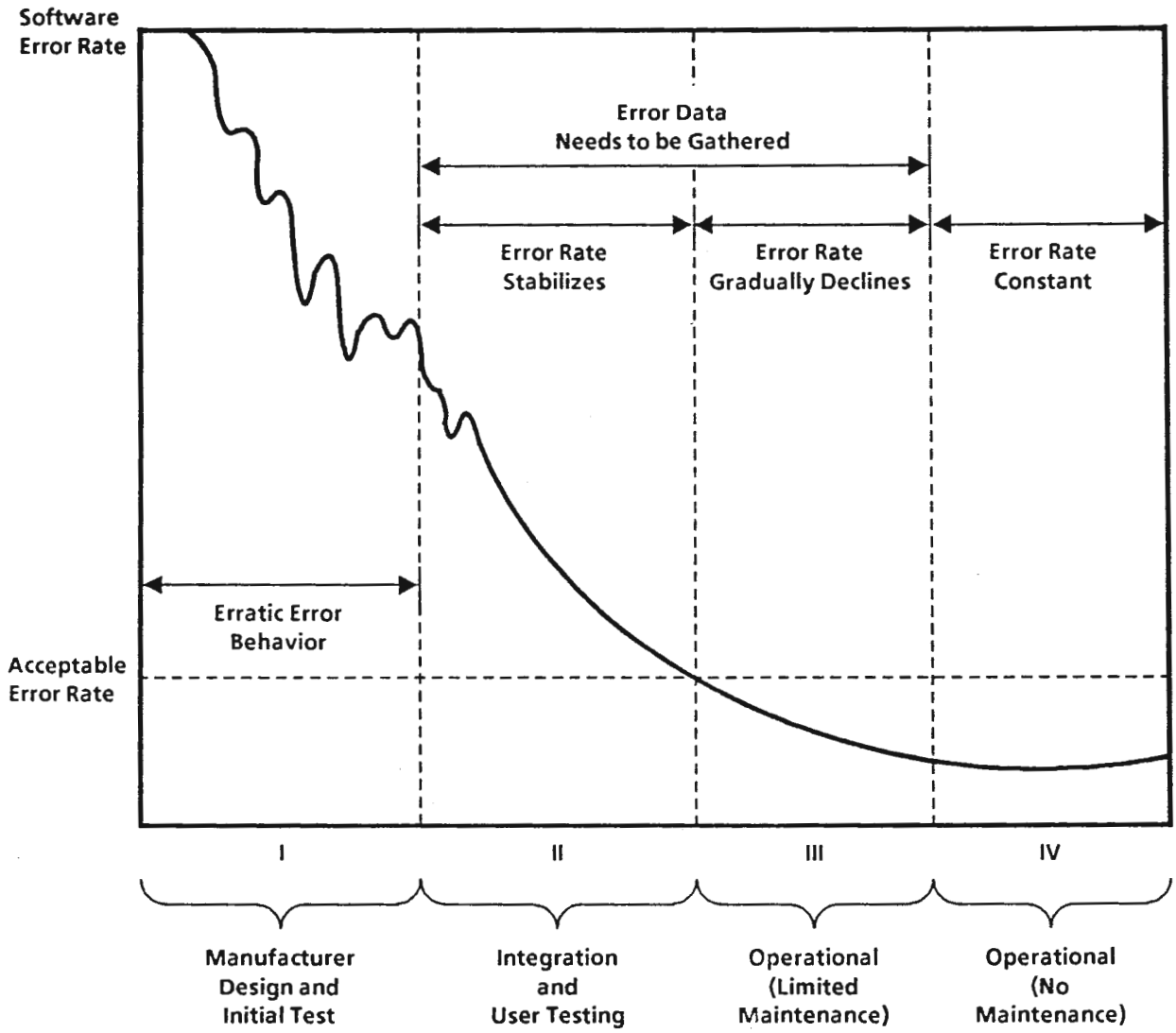



FIGURE 3.1-1. PHASES OF SOFTWARE DEVELOPMENT
(Gephart et al., 1978)

## 3.2. Software Testing

Software testing, used primarily during Phases II and III to determine the existence of errors in the coded software, is a somewhat heuristic process: Testing effectiveness is dependent upon the tester's ability to define test selection rules which, in turn, can be translated into tests that are sufficient for verifying the software's correctness. Test cases can be generated based upon the structure of the software (statements, control flow, etc.) or the functions identified in the functional specification. In either case, testing will reveal the presence of only those errors it was designed to identify; it will not detect those errors which are outside the testing boundaries.

A potential solution to this problem is the development of a metric (or methodology) which can be used to predict (based on data obtained during testing) the overall reliability of the software, including the expected number of errors in the program. This is the approach taken by software reliability models which have been specifically developed to address these problems. It is the subject of the next portion of this tutorial.

# 4. FAULT-INTOLERANT RELIABILITY MODELS

## 4.1. Introduction

In this section, the assumptions and characteristics of one category of software reliability models, those designed for single-version, fault-intolerant software, are presented. These models, which are based mainly on the failure history of the software, are time-dependent. The time-dependent approach uses either the times between failures (time-between-failure models) or the number of failures observed in a sequence of test time intervals (error-count models) to estimate the reliability of the software undergoing testing.

Both classes of time-dependent models are intended to be used during the software testing and debugging phase to provide either a reliability estimate or some other useful parameter (e.g., number of remaining errors, test reliability, and confidence in the program) which in turn yields some measure of the correctness of the program. Application of these models relies on methods of testing, verification, code walk-through, etc. to first provide the raw data (either as time between failures or as number of failures per individual test interval). Models are then used to estimate the expected performance statistics based on the use of maximum likelihood estimation (or least squares approximation) statistics and the underlying distribution of the data. This process, based on Goel's (1983) description and example data, is explained in more detail below and is summarized in figure 4.1-1.

### 4.1.1. Step 1: Evaluating the Failure Data Acquired From Testing

The initial step in selecting and applying a reliability model is to examine the test error data in order to identify the underlying data distribution. Identification of the data distribution is crucial because this distribution determines (1) whether the data can appropriately be modeled by time-dependent models; and (2) which class of time-dependent models (time-between-failure or failure-count) should be used. To ensure that the actual data distribution is attained, biases in the testing process need to be accounted for in the modeling process. For example, the actual distribution may be distorted if more and more functions were added during the subsystem or system test, or if test case severity changed during testing. Such biases may be compensated for by use of statistical procedures, such as normalizing the data to meet the steady-state system assumption of the models.

To demonstrate how the reliability modeling process occurs, an example will be shown. The data to be used are given as times between failures and are shown in table 4.1-1 (Goel, 1983). These data are also plotted in figure 4.1-2 (Goel, 1983).

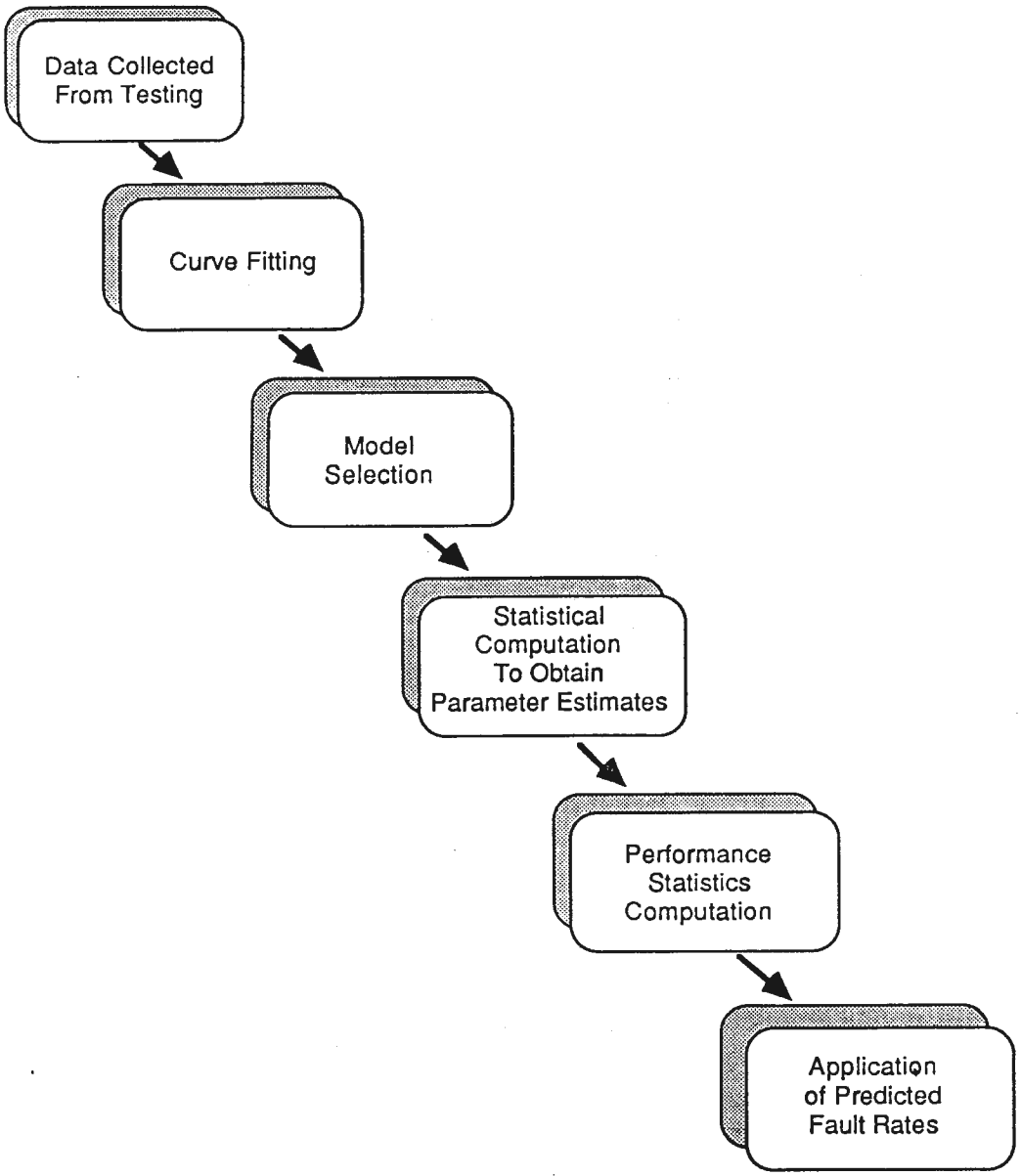



FIGURE 4.1-1. FLOW FOR COMPLETING THE SOFTWARE RELIABILITY ESTIMATION PROCESS

TABLE 4.1-1. FAILURES IN ONE HOUR (EXECUTION TIME) INTERVALS AND CUMULATIVE FAILURES (from Goel, 1983)

| Hour | No. | Cum. |
|---|---|---|
| 1 | 27 | 27 |
| 2 | 16 | 43 |
| 3 | 11 | 54 |
| 4 | 10 | 64 |
| 5 | 11 | 75 |
| 6 | 7 | 82 |
| 7 | 2 | 84 |
| 8 | 5 | 89 |
| 9 | 3 | 92 |
| 10 | 1 | 93 |
| 11 | 4 | 97 |
| 12 | 7 | 104 |
| 13 | 2 | 106 |
| 14 | 5 | 111 |
| 15 | 5 | 116 |
| 16 | 6 | 122 |
| 17 | 0 | 122 |
| 18 | 5 | 127 |
| 19 | 1 | 128 |
| 20 | 1 | 129 |
| 21 | 2 | 131 |
| 22 | 1 | 132 |
| 23 | 2 | 134 |
| 24 | 1 | 135 |
| 25 | 1 | 136 |

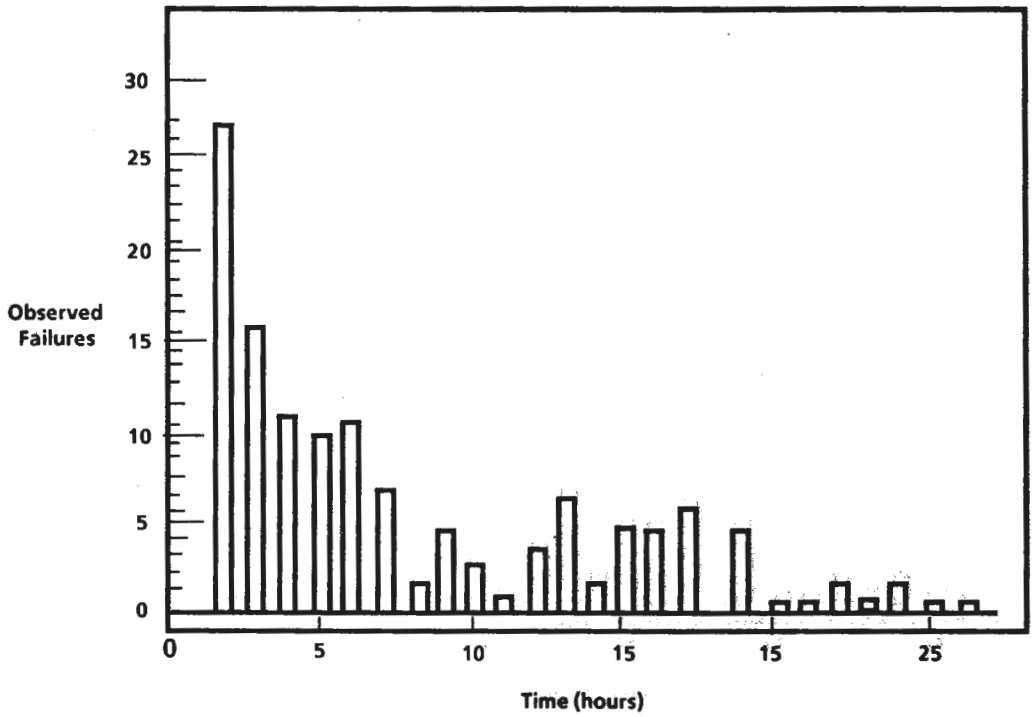



FIGURE 4.1-2. OBSERVED FAILURES PER HOUR (from Goel, 1983)

4.1.2. Step 2: Curve Fitting

After the data have been plotted, examined, and normalized (as required), decisions about the applicability of a class of models or the selection of a specific model can be made based upon the apparent distribution of the data. The two classes of models (time-between-failure, failure count) require different assumptions, as outlined in table 4.1-2. In addition, each class of model assumes a specific statistical distribution.

In the case of time-between-failure models, the test data must follow an exponential distribution since these models assume that the failure interval is constant and proportional to the current fault content of the program. The constant failure rate assumption means that for a given system of software, a given failure can be expected to occur during the next time interval regardless of when during testing the time interval is selected. However, successive times between failure are expected to increase in duration as faults are removed from the software. (Note that for a given set of test data this may not be entirely true, since the failure times are random variables and the resultant data are subject to statistical fluctuations.) Thus, for this given class of models, the assumed underlying distribution is expected to reflect the overall improvement in software quality as faults are detected and removed. The parameters of the distribution depend upon the number of faults remaining in the system, in a given interval, after the ith failure.

TABLE 4.1-2. FAULT-INTOLERANT SOFTWARE RELIABILITY MODEL ASSUMPTIONS (from Goel, 1983)

| Time-Between-Failure Models | Error-Count Models |
|---|---|
| • Independent times between failures | • Testing intervals are independent of each other |
| • Equal probability of the exposure of each fault | • Testing during intervals reasonably homogeneous |
| • Embedded faults are independent of each other | • Numbers of faults detected during non-overlapping intervals are independent of each other |
| • Immediate fault removal, perfect fault removal, no new faults introduced during correction | |

Failure count models attempt to model the number of failures in a given testing interval. As faults are removed from the system, these models assume that the number of failures observed per unit time will decrease and the curve representing cumulative number of failures versus time will eventually level off. Typically, duration of time intervals is fixed a priori and the number of failures in each interval is a random variable. Consequently, the underlying data must be distributed as a poisson distribution in that the failure rate (number of failures per unit time) is expected to decrease over time, thus demonstrating an increase in program reliability.

The data used in the example demonstrate a monotonically decreasing failure rate (number of failures per hour) and appear to conform to a poisson distribution (see figure 4.1-3; Goel, 1983). Consequently, a fault-count model is appropriate for these data.

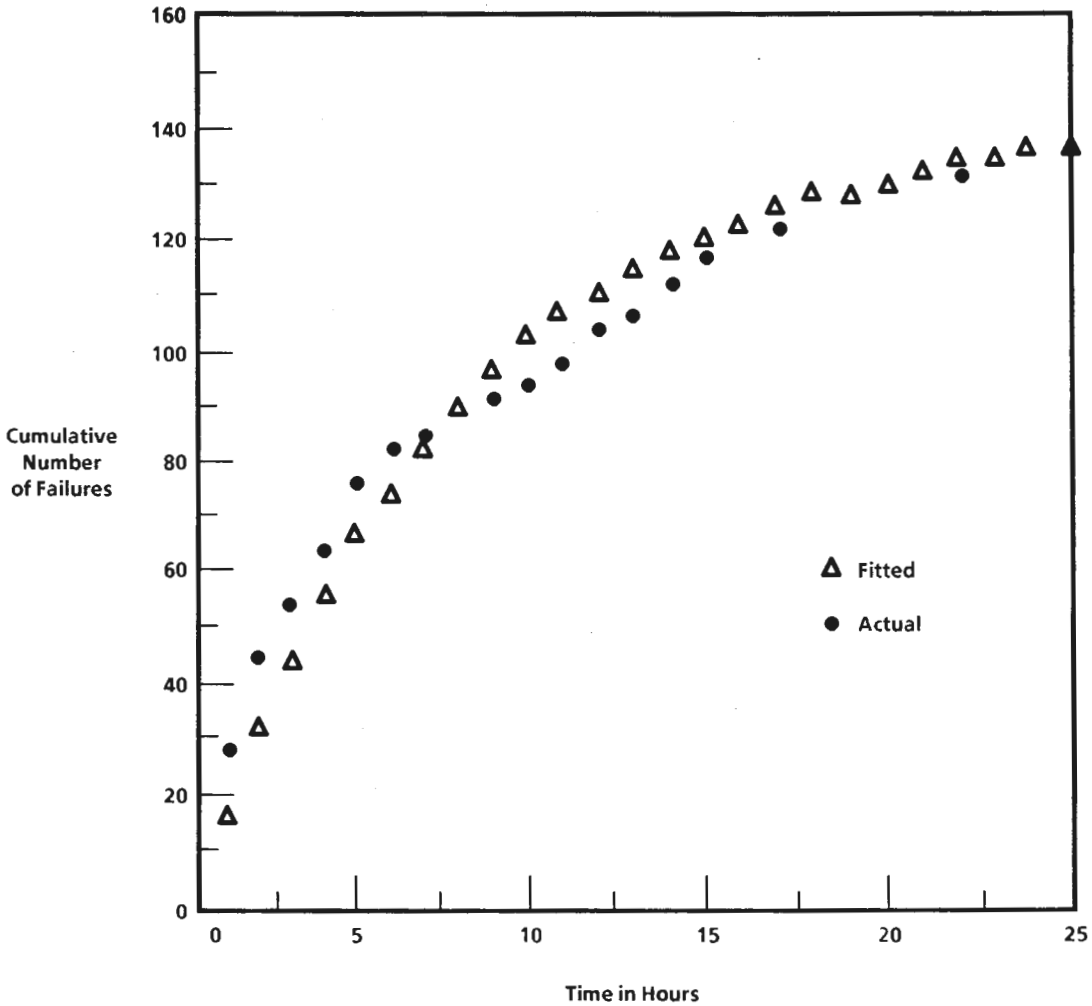



FIGURE 4.1-3. ACTUAL VERSUS PREDICTED FAILURES PER HOUR (from Goel, 1983)

### 4.1.3. Step 3: Model Selection

At this point, a specific fault-count model can be selected based upon the results obtained from the previous two steps. Listed in table 4.1-3 (Goel, 1983) are current time-dependent models appropriate for real-time applications. (See Prater, Hitt, and Eldredge, 1986, for a discussion of model requirements for real-time applications.) Included are both time-between-failure and failure-count models.

TABLE 4.1-3. FAULT-INTOLERANT SOFTWARE RELIABILITY MODELS (from Goel, 1983)

| Time-Between-Failure Models | Error-Count Models |
|---|---|
| • Jelinski-Moranda De-Eutrophication Model | • Goel-Okumoto Non-Homogeneous Poisson Process (NHPP) Model |
| • Geometric De-Eutrophication | • Geometric Poisson Model |
| • Generalized Imperfect Debugging Model | • Extended Jelinski-Moranda Model |
| • Bug-Proportional Model | • Modified Geometric De-Eutrophication Model |
| | • Shooman Exponential Model |

Selection of an appropriate model requires that the assumptions of the model be met by characteristics of the data. These assumptions are listed for each model in appendices A and B. (Goel, 1983, provides a useful discussion of the validity of these assumptions.)

For the current example, it was determined that the assumptions associated with the Goel-Okumoto model best corresponded to the data. The assumptions for this model are listed in table 4.1-4.

### 4.1.4. Step 4: Statistical Computation to Obtain Parameter Estimates (from Goel, 1983)

Shown in tables 4.1-5 and 4.1-6 are the various parameters and equations required for reliability estimation for each of the models. The parameters are used by the model to define the shape of the predicted fault distribution and to calculate the performance parameters (such as expected number of errors and expected time to the nth failure) which are given by the equations.

TABLE 4.1-4. ASSUMPTIONS FOR THE GOEL-OKUMOTO NON-HOMOGENEOUS POISSON PROCESS MODEL (from Goel, 1983)

- Initial fault content:

  Expected number of software faults to be eventually detected is an unknown fixed quantity.

  Actual number of faults to be observed is a random variable.

- Independence of faults:

  Each failure is caused by one fault and each of them is equally likely to cause a failure during testing.

  Number of software faults detected during non-overlapping testing intervals is independent of each other.

- Fault removal process:

  Fault removal time is negligible.

  No new faults are introduced during the fault removal process.

- Intensity function:

  Expected number of software faults detected during $(t, t + \Delta t)$ is proportional to the expected number of software faults undetected by time t, i.e.,

EQN 20

where

$m(t)$ = expected number of software faults detected by time t,

$a$ = expected number of software faults to be eventually detected,

$b$ = constant of proportionality.

The intensity function or the fault-detection rate $\lambda(t)$ is a decreasing function of t and is given by

EQN 21

where a, b, and m(t) are as defined above.

TABLE 4.1-5. PERFORMANCE MEASURES FOR TIME-BETWEEN-FAILURE MODELS

| Model | Hazard Function | Reliability Function | Mean Time to Failure |
|---|---|---|---|
| Jelinski-Moranda De-Eutrophication Model | $Z(X_i) = \phi[N - (i - 1)]$<br>where<br>N = the total number of initial errors in the program<br>$\phi$ = a proportionality constant<br>$X_i$ = the length of the ith debugging interval (the time between detection of the (i - 1)st and the ith errors)<br>i = the number of errors discovered | $R(X_i) = \exp[-\phi(N-n)X_i]$ | $MTTF = 1/[\phi (N - n)]$<br>where<br>n = the number of errors found to date |
| Geometric De-Eutrophication Model | $Z(X_i) = DK^{i-1}$<br>where<br>$X_i$ = the ith debugging interval<br>D = the initial error detection rate<br>K = a proportionality constant<br>i = the number of errors discovered after i interval | $R(Xi) = \exp[- DK^nX_i]$<br>where<br>n = the total number of errors discovered | $MTTF = \frac{1}{DK^n}$ |
| Generalized Imperfect Debugging Model | $\lambda(t) = \phi(N - p(i - 1))t^{\alpha - 1}$<br>where<br>$\phi$ = a proportionality constant<br>N = the total number of errors<br>p = the probability of perfect programmer debugging behavior<br>$\alpha$ = the parameter that controls the shape of the failure rate | $R(t) = \exp(- \frac{\phi(N - p(i - 1))t^{\alpha}}{\alpha}$ | $MTTF = \frac{1}{\alpha} \left( \frac{\alpha}{(N - p(i - 1))} \right)^{\frac{1}{\alpha}} \Gamma\left(\frac{1}{\alpha}\right)$ |
| Bug-Proportional Model | $Z(t) = K\varepsilon_r(\tau) = K[(E_T/I_T) - \varepsilon_c(\tau)]$<br>where<br>K = an arbitrary constant<br>$\varepsilon_r(\tau)$ = the number of remaining bugs<br>$E_T$ = the total number of errors originally present<br>$I_T$ = the total number of machine instructions<br>$\varepsilon_c(\tau)$ = the number of corrected bugs | $R(t) = \exp\{- [K\varepsilon_r(\tau)]t\}$<br>$= \exp\{-K[(E_T/I_T) - \varepsilon_c(\tau)]t\}$ | $MTTF = \frac{1}{K\varepsilon_r(\tau)} = \frac{1}{\beta(1 - \alpha\tau)}$<br>where<br>$\beta = \frac{E_T}{I_T} K$<br>$\alpha = \frac{\rho_0 I_T}{E_T}$<br>where<br>$\rho_0$ = a constant rate of error correction |

## TABLE 4.1-6. PERFORMANCE MEASURES FOR FAULT-COUNT MODELS

| Model | Hazard Function | Reliability Function | Mean Time to Failure |
|---|---|---|---|
| Geometric Poisson Model | $Z(t_i) = \lambda K^{i-1}$<br>where<br>$t_i$ = the ith debugging interval<br>$\lambda$ = the average number of faults occurring in the first interval<br>K = a proportionality constant, $0 < K < 1$ | $R(t) = e^{-\lambda K^i t}$ | $MTTF = \frac{1}{\lambda K^i}$ |
| Extended Jelinski-Moranda Model | $Z(t_i) = \phi[N - n_{i-1}]$<br>where<br>$\phi$ = a proportionality constant<br>N = the total number of initial errors<br>$n_i$ = the cumulative number of errors found through the ith time interval<br>$t_i$ = the ith debugging interval | $R(t) = e^{-\phi(N - n_i)t}$ | $MTTF = \frac{1}{\phi[N - n_i]}$ |
| Modified Geometric De-Eutrophication Model | $Z(t_i) = DK^{M_i - 1}$<br>where<br>D = the fault detection rate<br>K = a positive constant less than 1<br>$M_{i-1}$ = the cumulative number of errors detected<br>$M_i$ = the cumulative number of errors found up to the ith time interval | $R(t) = e^{-DK^{M_n}t}$ | $MTTF = \frac{1}{DK^{M_n}}$<br>where<br>n = the total number of time intervals |
| Shooman Exponential Model | $Z(t) = Ce_r(\tau)$<br>where<br>C = a proportionality constant | $R(t) = \exp\{-C[(E/I) - e_c(\tau)]t\}$<br>where<br>E = the total number of errors present at time $\tau = 0$<br>I = the total number of machine instructions<br>$e_c(\tau)$ = the total number of errors corrected in interval $\tau$ | $MTTF = 1/\{[C[E/I) - e_c(\tau)]\}$ |

| Model | Expected Number of Detected Faults | Expected Number of Remaining Faults | Software Reliability |
|---|---|---|---|
| Goel-Okumoto Non-Homogeneous Poisson Process Model | $\hat{m}(t) = \hat{a}(1 - e^{-\hat{b}t})$ | $E[\bar{N}(t)] = \hat{a}e^{-\hat{b}t}$<br>where<br>$\bar{N}(t)$ = number of faults remaining in the system at time t | $\hat{R}_{x_k}[S_{k-1}(x\|S) = \exp[-\hat{a}\{e^{-\hat{b}s} - e^{-\hat{b}(s+x)}\}]$<br>where<br>$X_k$ = the time between failures (k - 1) and k<br>$S_k$ = the time to k failures |

In order to obtain the parameter estimates, statistical methods (such as least squares and maximum likelihood estimation) are used. In the case of the Goel-Okumoto model, the method of maximum likelihood estimation is used. This statistic provides an estimate of the likelihood that the sample of error data collected during testing actually represents the true data. This is necessary because testing is based on a selected testing strategy that does not thoroughly test all aspects of the software. Consequently, the error data identified is only a subset of the total error data population and may be subject to testing bias.

The basic formula for the Goel-Okumoto model is:

$$m(t) = a(1 - e^{-bt})$$

where a and b are defined as in table 4.1-4.

The total number of software faults detected by time t, N(t), under the above assumptions is representative of the NHPP with mean value functions m(t) and intensity function $\lambda(t)$ as given above. The distribution of N(t), hence, is given by

$$P\{N(t) = y\} = \frac{\{m(t)\}^y}{y!} e^{-m(t)}, y = 0,1,2,\ldots$$

and

$$E[N(t)] = m(t) = a(1 - e^{-bt}) .$$

Using this equation and the method of maximum likelihood estimation as follows, estimates of a and b can be obtained.

Suppose $y_1, y_2, \ldots, y_n$ are the cumulative number of software faults detected by times $t_1, t_2, \ldots, t_n$, respectively. Then the likelihood function for (a,b) given the data pairs

$\{(y_i, t_i), i = 1,2,\ldots,n\}$ is

$$L(a,b \mid y_1, y_2, \ldots, y_n, t_1, t_2, \ldots, t_n) =$$

$$= \Pr\{N(t_1) = y_1, N(t_2) = y_2, \ldots, N(t_n) = y_n\}$$

$$= \prod_{i=1}^{n} \frac{[m(t_i) - m(t_{i-1})]^{y_i - y_{i-1}}}{(y_i - y_{i-1})!} e^{-\{m(t_i) - m(t_{i-1})\}}$$

$$= \prod_{i=1}^{n} \frac{\{a(e^{-bt_{i-1}} - e^{-bt_i})\}^{y_i - y_{i-1}}}{(y_i - y_{i-1})!} e^{-a(1 - e^{-bt_n})}$$

Maximum likelihood estimation of parameters a and b can then be obtained by solving the following pair of equations simultaneously:

$$a(1 - e^{-bt_n}) = y_n$$

and

$$at_n e^{-bt_n} = \sum_{i=1}^{n} \frac{(y_i - y_{i-1})(t_i e^{-bt_i} - t_{i-1} e^{-bt_{i-1}})}{e^{-bt_{i-1}} - e^{-bt_i}}$$

The above two equations yield

$$\frac{y_n t_n e^{-bt_n}}{(1 - e^{-bt_n})} = \sum_{i=1}^{n} \frac{(y_i - y_{i-1})(t_i e^{-bt_i} - t_{i-1} e^{-bt_{i-1}})}{e^{-bt_{i-1}} - e^{-bt_i}}$$

from which we can find $\hat{b}$ numerically and hence

$$a = \frac{y_n}{(1 - e^{-bt_n})}$$

$\hat{a}$ = 142.32 (Expected number of faults)

$\hat{b}$ = 0.1246 (Faults per fault per hour)

4.1.5. Step 5: Computation of Performance Statistics

Tables 4.1-5 and 4.1-6 show the equations used to calculate the performance statistics associated with each model. These performance statistics include: the hazard function, reliability function, and Mean Time to Failure (MTTF). The hazard function estimates the likelihood of software failure at a given instant of time. The reliability function provides an estimate of the probability that the software will not fail in a given time interval. MTTF is the expected value of the time to next failure.

The Goel-Okumoto model uses slightly different performance statistics which are roughly analogous to the performance statistics just described. Computation of these statistics is shown below:

Expected number of software faults detected by time t is given by

$$m(t) = a\,(1 - e^{-bt})$$

Expected number of remaining faults in the software system at time t is given by

$$E[N(t)] = ae^{-bt}$$

where $\hat{N}(t)$ = number of faults remaining in the system at time t.

Let $X_k$ be the time between failures (k-1) and k and let $S_k$ be the time to k failures. Then it can be shown that the conditional reliability function of $X_k$, given $S_{k-1} = s$, is,

$$R_{X_k \mid S_{k-1}}(x \mid s) = \exp\left[-a\left\{e^{-bs} - e^{-b(s+x)}\right\}\right].$$

The fitted model is obtained by substituting the above estimated values.

$$\hat{m}(t) = 142.32\ (1 - e^{-0.1246t})$$

and

$$\hat{\lambda}(t) = 17.73 \times e^{-0.1246t}$$

4.1.6. Step 6: Application of Predicted Fault Rates

At this stage, the obtained error distribution and the predicted error distribution (based upon the equation used to compute expected number of remaining faults) can be plotted and compared (see figure 4.1-4; Goel, 1983).

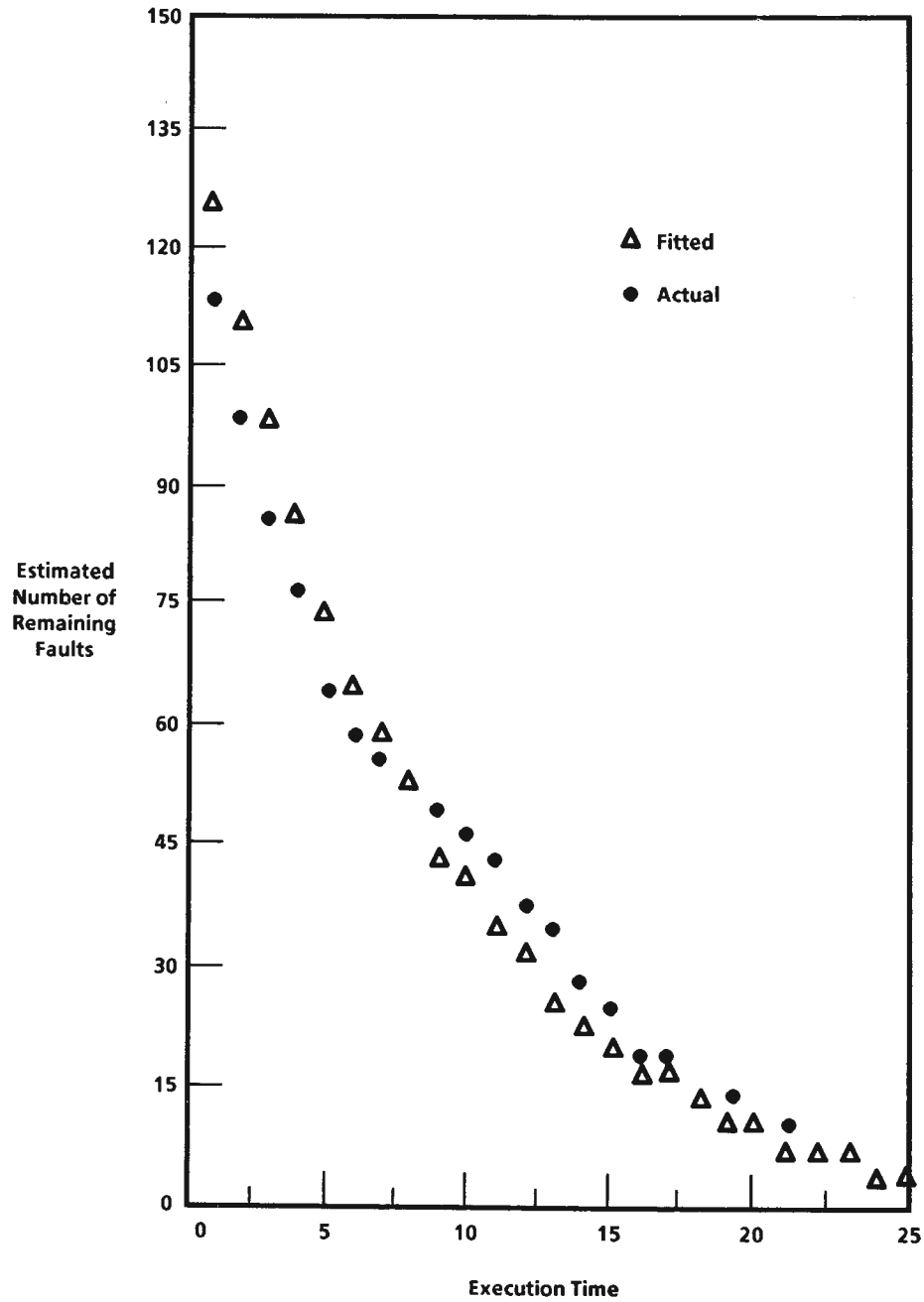



FIGURE 4.1-4. PREDICTED VERSUS ACTUAL NUMBER OF REMAINING FAULTS (from Goel, 1983)

In order to ensure that the predicted model fits the obtained data, goodness-of-fit tests should be applied. If the model fits, results of the reliability modeling process can be appropriately used to make decisions about the software program, for example, to determine whether further testing is required. If the model does not fit, additional testing and data collection may be needed and the process may have to be repeated either using the same model or a more appropriate model based upon the additional data.

## 4.2. Computation of a Time-Between-Failures Model

In section 4.1, the Goel-Okumoto model, a fault-count model which assumes a poisson distribution, was described in some detail. In Steps 1 and 2, if the data generated from the testing had been exponentially distributed and met the assumptions of one of the time-between-failure models, a different model would have been selected, such as the Jelinski-Moranda De-Eutrophication Model. Steps 3 through 5 would then have been performed as follows:

### 4.2.1. Step 3: Model Selection

As before, the assumptions of the selected model must be matched with the characteristics of the data. Assume a set of data which matches the assumptions of the Jelinski-Moranda model (see table 4.2-1; Goel, 1983).

### 4.2.2. Step 4: Statistical Computation to Obtain Parameter Estimates (from Goel, 1983)

The time between the (i-1)st and the ith failures $T_i$, is distributed exponentially with parameter $\Phi[N - (i-1)]$.

$$\text{pdf of } T_i\text{: } f(t_i) = \phi[N-(i-1)]\exp[-\phi\{N-(i-1)\}t_i]$$

$$\text{cdf of } T_i = F(t_i) = 1 - \exp[-\phi\{N-(i-1)t_i\}]$$

A series of n failures is observed with interfailure times as $t_1, t_2, \ldots, t_n$. Usually, the method of maximum likelihood is used to estimate the parameters N and $\Phi$ as shown below.

The likelihood function of N and $\Phi$ is

$$L(N, \phi | t_1, t_2, \ldots, t_n) = \prod_{i=1}^{n} \phi[N-(i-1)]\exp[-\phi\{N-(i-1)\}t_i]$$

TABLE 4.2-1. ASSUMPTIONS FOR THE JELINSKI-MORANDA DE-EUTROPHICATION MODEL (from Goel, 1983)

- Initial fault content:

  An unknown fixed constant N.

- Independence of faults:

  Each fault in the program is independent of other faults and each of them is equally likely to cause a failure during testing. Times between occurrences of faults are independent of each other.

- Fault removal process:

  A detected fault is removed with certainty at the end of each testing interval.

  Only one fault is removed during each testing interval.

  The fault removal time is negligible.

  No new faults are introduced during the fault removal process.

- Hazard function:

  The software failure rate of the hazard function during a failure interval is constant and is proportional to the current fault content of the tested program. Thus, during the ith testing interval,

$$z(t_i) = \phi\{N - (i-1)\},$$

  where $\Phi$ is a proportionality constant, and N is the initial number of faults in the program.

The Maximum Likelihood Estimates (MLE) of N and $\Phi$ are obtained by solving the following pair of equations simultaneously.

$$\sum_{i=1}^{n} \frac{1}{N-(i-1)} - \sum_{i=1}^{n} \phi t_i = 0$$

and

$$\frac{n}{\phi} - \sum_{i=1}^{n} [N-(i-1)]\, t_i = 0$$

The above two equations yield

$$\sum_{i=1}^{n} \frac{1}{N-(i-1)} = \frac{n}{N - \frac{1}{T} \sum_{i=1}^{n} (i-1)t_i},$$

where

$$T = \sum_{i=1}^{n} t_i .$$

The above equations can be solved numerically to find $\hat{N}$ and then $\hat{\Phi}$ can be obtained from the following equation.

$$\hat{\phi} = \frac{n}{NT - \sum_{i=1}^{n} (i-1)t_i}$$

Using the asymptotic properties of the maximum likelihood estimators, it can be shown that

$$\mathrm{Var}(\hat{N}) = \frac{n}{\phi 2} \cdot \frac{1}{\det A}$$

$$\mathrm{Var}(\hat{\phi}) = \Sigma \left( \frac{1}{N-i-1} \right)^2 \cdot \frac{1}{\det A}$$

$$\mathrm{Cov}(\hat{N},\hat{\phi}) = - \frac{T}{\det A}$$

$$\rho_{\hat{N},\hat{\phi}} = - \frac{T\phi}{\sqrt{n} \cdot \sqrt{\Sigma_2}},$$

where

$$\det A = \frac{n}{\phi 2} \sum_{i=1}^{n} \left( \frac{1}{N-i+1} \right)^2 - T^2,$$

and

$$\Sigma_2 = \sum_{i=1}^{n} \frac{1}{(N-i+1)^2}$$

4.2.3. Step 5: Computation of Performance Statistics

Reliability at time t after the nth failure is

$$R_n(t) \equiv P(T_{n+1} > t) = \exp[-\phi \{N-n\} t]$$

MTTF after n failures is

$$MTTF_n = \int_0^\infty R_n(t)\,dt = \frac{1}{\phi[N-n]}$$

Using the variance-covariance of $\hat{N}$, $\hat{\Phi}$, it can be shown that

$$\text{Variance}[R_n(t)] = \frac{e^{-2\phi_1(N-n)t}}{\det A}\{nt^2 - 2(N-n)Tt^2\phi + (N-n)^2t^2\Sigma_2\}$$

$$\text{Variance}(MTTF_n) = \frac{1}{\det A}\left[\frac{1}{\phi 2}v_1^2 - 2Tv_1v_2 + v_2^2\Sigma_2\right],$$

where

$$v_1 = -\frac{1}{(N-n)^2\phi}$$

$$v_2 = -\frac{1}{(N-n)\phi^2}, \text{ and}$$

$\Sigma_2$ and det A are as defined before.

All the above quantities can be computed by replacing N and $\Phi$ by $\hat{N}$ and $\hat{\Phi}$, respectively.

Worked examples of the Goel-Okumoto and Jelinski-Moranda models are provided in appendices C and D.

# 5. FAULT-TOLERANT SOFTWARE

## 5.1. Introduction

Fault-tolerant software is a redundancy technique that is currently being proposed for use in aircraft avionics/electronics applications where system failures are potentially disastrous, and where high reliability is required. The most widely used methods for implementation are the RB and N-version programming, which are roughly analogous to the hardware redundancy techniques of "standby sparing" and multiple modular redundancy. These fault-tolerant features for software are primarily aimed at overcoming the effects of errors in software design and coding; however, they may also compensate for failures due to hardware design deficiencies or malfunctions in input/output channels, depending upon how the versions are implemented. Fault-tolerant software by its very nature tries to incorporate techniques which will ensure that service is maintained by coping with any faults that remain after the application of all fault avoidance measures. In that respect, it does not rely on completion of validation or testing efforts, which can only establish the presence of errors but cannot assure their absence.

## 5.2. Principles of Fault Tolerance

Anderson and Knight (1982) identify four phases into which fault-tolerance techniques can be divided:

"1. ERROR DETECTION. In order to tolerate a fault its effects must first be detected. Clearly, this can only be achieved by performing checks to determine whether any erroneous situation has arisen.

2. DAMAGE ASSESSMENT. Having detected that the system is in error, it will usually be necessary to identify how much of the state of the system has been corrupted.

3. ERROR RECOVERY. Probably the most important aspect of fault tolerance is the provision of an effective means of transforming an erroneous state of the system into a well defined and error free state. Methods for achieving this transformation can sometimes make good use of the information retained in the erroneous state, but it can be more secure to simply discard the erroneous state and reset the system to some prior state (a recovery point).

4. CONTINUED SERVICE. In order to enable the system to continue to provide the service required by its specification, further action may be needed to ensure that the fault whose effects have been obviated does not immediately recur and thus ruin the whole

approach. Unless the fault was transient and will not recur in any case, it must either be rectified or circumvented (p. 24-3)."

As stated above, two techniques for achieving fault tolerance are the RB and N-version programming. In the case of N-version programming, a number of independently coded programs for a given function are run simultaneously (or nearly so) on N computers. The system is coupled in such a way that the N computers are hardware independent yet able to communicate efficiently so that rapid comparisons of results can be achieved. The key to successful implementation of this approach is the independent design and development of the N-versions, which prevents the occurrence of the same fault in the multiple versions.

The RB, on the other hand, requires only a single version of hardware (even though redundant hardware is often employed) and relies on the use of the acceptance test (often augmented by a "watch-dog" timer) to assure that an acceptable result is furnished within a reasonable, specified period of time. In the case where the acceptance test fails, or time expires, alternate software routines are called until one successfully passes the acceptance test. Once again, it is assumed that the alternates demonstrate independence in the way in which they were designed and coded. The acceptance test is usually structured to:

- Detect deviations from expected program execution.
- Prevent unsafe output.

Both are critical to safe performance (Hecht, 1979).

The advantage of the fault-tolerant software approach over other techniques for software reliability improvement (test, verification, etc.) is that it permits the use of system reliability modeling techniques similar to those used for hardware reliability modeling. Reliability models can be developed to describe the behavior of RB, N-version programming, or any combination of the two approaches.

## 5.3. Fault-tolerant Software Reliability Models

In the remainder of section 5, two software reliability models will be presented: the Hitt, E. F., J. J. Webb, and M. S. Bridgeman model (Prater, Hitt, and Eldredge, 1986) and the Scott et al., model (1983, 1984, 1987). The Hitt, E. F., J. J. Webb, and M. S. Bridgeman model will demonstrate, through worked examples, how various fault-tolerant software approaches can be decomposed and the reliability of the constituent parts estimated. The Scott et al., model is similar to the Hitt, E. F., J. J. Webb, and M. S. Bridgeman model in its assumptions, and the two models may provide equivalent results. The Scott et al., model is presented, however, to show concurrent work in the area of fault-tolerant software reliability modeling.

In contrast to the fault-intolerant models, neither fault-tolerant model has been verified/validated to the point where it has become accepted for general use. However, the use of fault-tolerant software has started to be accepted for

avionics applications, and the fault/failure data from these efforts are only now becoming available. It is expected that the two fault-tolerant software approaches will be verified and validated using the emerging database.

## 5.4. A Hierarchical Software Reliability Model

This part of the tutorial will describe the hierarchical software reliability model developed by Hitt, E. F., J. J. Webb, and M. S. Bridgeman (Prater, Hitt, and Eldredge, 1986) to predict the reliability of fault-tolerant software prior to its development (i.e., coding). Fault-tolerant software consists of individual software modules, such as single-version software, NVS, decision algorithms, RBs, and acceptance tests. In the model, these modules are treated as separate "function blocks," similar to piece-parts in hardware reliability modeling. Each function block is subject to specific types of faults.

### 5.4.1. Software Components

#### 5.4.1.1. Single Version Software

This is a probabilistic model of deterministic or random events. The program execution is usually deterministic, and the development process is probabilistic. Single version software contains characteristic faults that must be accounted for. Examples of these faults are as follows:

- Incorrect specification.
- Misunderstood or unclear specification.
- Algorithmic error (sometimes called a computational or logic error).
- Input data error.
- Program logic error.
- Output data error.

#### 5.4.1.2. N-version Software

This software is a fault-tolerant software technique which implements two or more functionally equivalent versions, usually in parallel. These versions may be produced independently by separate programming teams, or they may be designed differently by examining and forcing differences into the versions. Upon comparison of the alternate versions, the faults should be distinguishable. Some of the faults associated with NVS include:

- Specification error.
- Performance error (due to incomplete, inconsistent, or ambiguous specifications).

- Non-termination error.
- Algorithmic error.
- Input data error.
- Output data error.

5.4.1.3. Decision Algorithm

The decision algorithm determines the specific output. This algorithm may be a majority vote, a median select, a bit-by-bit comparison (with the number of bits that are to be compared or that are significant specified), or an average. Considerations to be made in this software design are as follows:

- The type of decision algorithm used.
- The allowable range of discrepancy of each input from all other inputs to the decision algorithm.
- The data sensitivity of the decision algorithm.

5.4.1.4. Recovery Block

The RB method is a fault-tolerant software technique that provides alternate components which may be switched in (usually serially) to replace a faulty component that has been rejected by the acceptance test. These alternate components are designed independently from the main software component (the primary alternate). They generally provide only partial functionality of the software component. The alternate components perform a reduced and simplified function. Prior to entering an alternate, the conditions existing just before entry to the primary alternate are restored. Some examples of faults that occur in the software for RBs include:

- Specification error.
- Performance error.
- Non-termination error.
- Algorithmic error.
- Input data error.
- Output data error.

5.4.1.4.1. Forward Recovery Block

A forward RB restores the system to a consistent state by compensating for inconsistencies found in the current state. For a single process, the forward RB technique requires a complete understanding of the extent of damage and a scheme for correcting the inconsistencies. Exception values (or ranges) must

be specified for each data abstraction. These exceptions are specified in response to run-time attempts to violate inherent invariant properties. These anticipated faults can be managed by forward RB techniques.

5.4.1.4.2. Backward Recovery Block

Backward RB techniques require the restoration of the system to the last correct state. This is referred to as rollback. The computation resumes from that point. Unanticipated faults, i.e., design faults, are managed by a default exception handler using automatic backward recovery.

5.4.1.5. Acceptance Test

An acceptance test checks the acceptability of data generated by a software component. This is usually implemented in software as a logical expression or algorithm. The faults associated with an acceptance test include:

- Specification error.
- Performance error.
- Algorithmic error.
- Input data error.
- Output data error.

5.4.1.6. Rollback

The rollback restores the software to the condition existing prior to the execution of an incorrect or faulty version. The rollback resets the software to the original input state allowing the next version to run.

A rollback is used in connection with RB and hybrid NVS systems. Faults that are characteristics of rollback are:

- Specification error.
- Input data error.
- Output data error.
- Unrecoverable state.

5.4.1.7. Roll-Forward

The roll-forward is always used in conjunction with a forward RB. The roll-forward transfers the restored state obtained from the forward RB to a forward position in the system. The state for which the forward RB has compensated determines the forward position for this transfer. Faults associated with roll-forward include:

- Specification error.
- Performance error.
- Input data error.
- Output data error.

5.4.2. Inputs to the Software Reliability Model

Software reliability in this model is estimated from the individual probability reliability values (or safety, availability, or accuracy values, if desired) for each function block. Values for each block are either obtained from a software reliability database (which is emerging or assumed to exist), estimated by the lines of code (and the language), derived experimentally by subjecting the function block's software to a number of test cases and counting the failures to determine a reliability value, or obtained from lower (detailed) level models. The reliability values must be real numbers with a range of $0.0 \leq$ probabilistic reliability value (or safety, availability, or accuracy value) $\leq 1.0$.

5.4.3. Outputs from the Software Reliability Model

The output of the software reliability model is an overall probability reliability value (or safety, availability, or accuracy values, if desired) for the software as a whole. Reliability values can also be estimated for a specific subset of modules within the software.

5.4.4. Model Representation

Control system notation is used to represent the model. Each software module represents a transformation of input to output. Therefore, the reliability value for an individual module estimates the probability of the desired input/output transformation actually occurring. Similarly, input/output transformations taking place through the interaction of multiple modules can also be represented, allowing similar probability estimations of the likelihood that the transformation will occur. Examples of the application of control system notation to reliability estimation are shown in figures 5.4-1 (Prater, Hitt, and Eldredge) through 5.4-6. These figures are explained in more detail below.

5.4.5. The Modeling Process

The Hitt, E. F., J. J. Webb, and M. S. Bridgeman model was developed to allow software reliability estimation prior to actual software development and coding. This model is intended to be used with a computer software applications package designed to allow alternate versions of the same software architecture to be evaluated. Once a preliminary software architecture has been developed and decisions have been made as to anticipated number of N-versions, etc., these decisions can be entered into the computer. Specifically, the software developer can choose the appropriate modules from a menu system such as that shown in table 5.4-1. With the proposed applications software package, the

selected set of module types can then be shown on the computer display in the form of a block diagram.

TABLE 5.4-1. RELIABILITY MODEL MENU SELECTIONS
(from Prater, Hitt, and Eldredge, 1986)

- Structure Functional Modules
  - Single version software
  - NVS (The number of versions must be specified by the user.)
  - Decision algorithm
  - Recovery block (The number of alternates must be specified by the user.)
  - Acceptance test
  - Rollback
  - Roll-forward
- Transfer Functional Modules
  - Forward path
  - Positive feedback
  - Negative feedback
  - Positive feed-forward
  - Negative feed-forward

By means of this representation, reliability values for each module can be entered and reliability estimates of the preliminary software system obtained. If the reliability estimate is not sufficiently high, the software architecture can then be modified, for example, by adding additional versions for a given N-version module. In this way, progressive iterations can be evaluated until the desired reliability level is attained.

Note that table 5.4-1 (Prater, Hitt, and Eldredge, 1986) lists the decision algorithm and acceptance test independently of the NVS and RBs. This separation is necessary to allow for flexibility in combining individual modules. For example, the decision algorithm and acceptance test can now be used independently, as well as in conjunction with their respective pairs:

- NVS and decision algorithm, and
- Recovery block and acceptance test.

Keeping the decision algorithm and acceptance test as separate entities requires that each NVS, decision algorithm, RB, and acceptance test module have individual reliability values in order to compute the overall software system reliability. This is intended to improve the accuracy of the transfer functions for this portion of the diagram since it will accommodate variations in the implementation of these concepts.

Unfortunately, a software applications package capable of performing iterative reliability estimations is not yet available, in spite of the potential contributions such a tool could make to the process of ensuring high-quality software. Use of the Hitt, E. F., J. J. Webb, and M. S. Bridgeman model is not contingent upon the availability of such an applications package, however. This same concept of selecting individual modules and estimating the reliability of the software as a whole can be used without a software applications package. The steps involved are described below.

### 5.4.6. Detailed Function Blocks

The Hitt, E. F., J. J. Webb, and M. S. Bridgeman model uses the structure of the software (i.e., the expected reliabilities of the individual modules) to predict overall software reliability. Consequently, the accuracy of these predictions depends upon the extent to which the structure is "filled in." Dissecting the structure into progressively finer modules, and determining reliabilities for these modules, leads to greater predictive accuracy.

Figure 5.4-1 provides an example of a software program which has been decomposed into relatively high-level software modules. This example shows how reliability values are substituted for each function block to permit the determination of the overall system reliability. Because the software structure is represented at a very high level, the reliability values may be less predictive than desired. More precise predictions can be obtained if each of the software design techniques (single version software, N-version software, decision algorithm, RB, and acceptance test) are broken down into more detailed function blocks. (For a more finely structured model, see figure 5.4-6.)

Figure 5.4-2 is an overall representation of the functional constructs associated with each of the function blocks. Errors associated with each of these constructs are shown in tables 5.4-2 and 5.4-3. Each module in figure 5.4-1 has a reliability value associated with it. The occurrence of these errors within a specific software module drops the reliability of that module below 1.0.

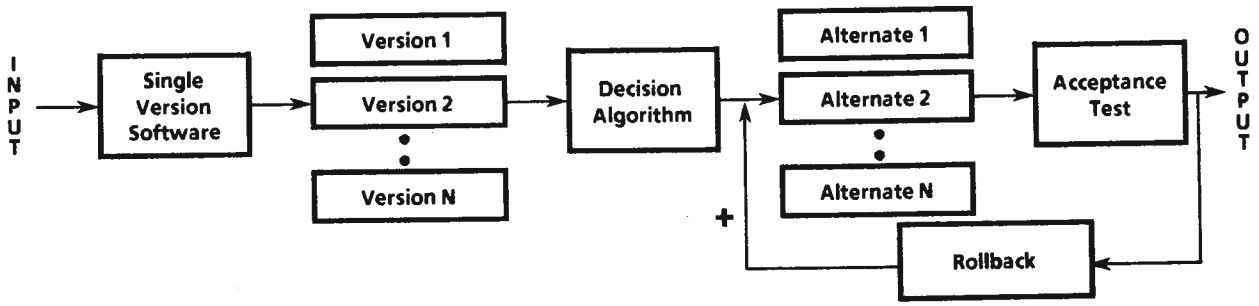



FIGURE 5.4-1. SIMPLE BLOCK DIAGRAM EXAMPLE
(from Prater, Hitt, and Eldredge, 1986)

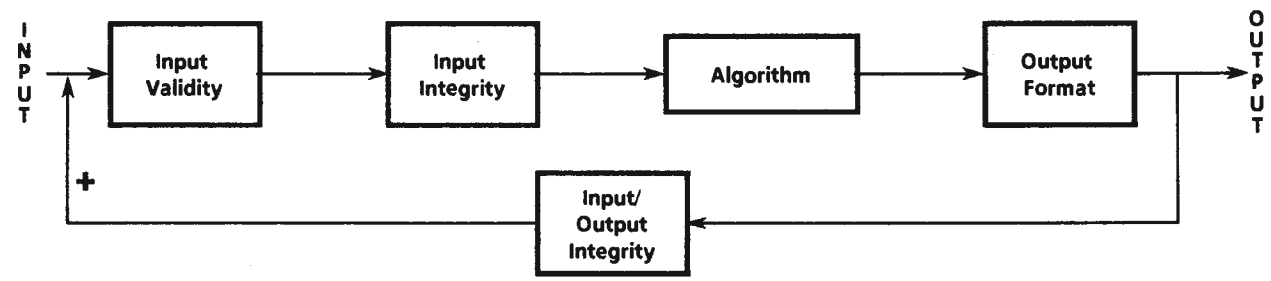



FIGURE 5.4-2. OVERALL REPRESENTATION OF THE FUNCTIONAL CONSTRUCTS
(from Prater, Hitt, and Eldredge, 1986)

TABLE 5.4-2. CORRELATION OF FAULTS WITH THE DETAILED PORTIONS OF THE SINGLE VERSION SOFTWARE FUNCTION BLOCK

| | Detailed Function Blocks | | | | |
|---|---|---|---|---|---|
| Faults | Input Validity | Input Integrity | Algorithm | Output Format | Input/ Output Integrity |
| Incorrect Specification | X | X | X | X | X |
| Misunderstood or Unclear Specification | X | X | X | X | X |
| Algorithmic Error | X | X | X | X | X |
| Input Data Error | X | | | | X |
| Program Logic Error | X | X | X | X | X |
| Output Data Error | | | | X | X |

5.4.7. Function Block States

The four possible states for any function block are as follows:

- The function block fails (an error is detected in the function block) and it is corrupt (contains one or more error).
- The function block passes, yet it is corrupt.
- The function block fails and it is error free.
- The function block passes and it is error free.

A mathematical truth table for these states is given in table 5.4-4.

TABLE 5.4-3. CORRELATION OF FAULTS WITH THE DETAILED PORTIONS OF THE DECISION ALGORITHM FUNCTION BLOCK

| | Detailed Function Blocks | | | | |
|---|---|---|---|---|---|
| Faults | Input Validity | Input Integrity | Algorithm | Output Format | Input/ Output Integrity |
| Input Range Error | X | X | | | |
| Algorithmic Error | X | X | X | X | X |

TABLE 5.4-4. TRUTH TABLE OF FUNCTION BLOCK STATES

| State | Error Exists in the Function Block | Error Detected in the Function Block | Error Corrected in the Function Block | Safe | Reliable | Available |
|---|---|---|---|---|---|---|
| a | T | T | T | T | T | T |
| a | T | T | F | T | T* | F |
| b | T | F | N/A | F | F | F* |
| c | F | T | N/A | T | F | F |
| d | F | F | N/A | T | T | T |

Key: N/A = not applicable
* = true or false (the common interpretation is given)

5.4.8. System Reliability

The function blocks will each have an associated transfer function ("reliability"). Block diagram reduction techniques can then be used to determine overall system reliability while signal-flow diagram reduction techniques provide the overall system transfer function ("reliability"). The signal-flow diagram is useful in analyzing multiple-loop feedback systems and in determining the effect of a particular element or parameter in an overall feedback system, whereas the block diagram is useful in the design and analysis of sections of a feedback system. Block diagram reduction techniques become tedious and time consuming as the number of feedback paths increases. To solve complex problems, it is much simpler to use the theorems and properties of signal-flow graphs.

5.4.9. General Equations

The equations used in this analysis follow S. J. Mason's theorems on the properties of signal-flow graphs. The general expression for the (closed loop) system transfer function using the signal-flow diagram reduction technique is given by

$$\text{Reliability} = \frac{\Sigma_K G_K \Delta_K}{\Delta}$$

where

$\Delta = 1 - \Sigma L_1 + \Sigma L_2 - \Sigma L_3 + \ldots + (-1)^n \Sigma L_n$,

$L_1$ = the gain of each closed loop in the graph,

$L_2$ = the product of the loop gains of any two non-touching closed loops,

$L_3$ = the product of the loop gains of any three non-touching closed loops,

$L_n$ = the product of the loop gains of any n non-touching closed loops,

$G_K$ = the gain of the Kth forward path,

$\Delta_K$ = the value of $\Delta$ for that part of the graph not touching the Kth forward path (Shinners, 1978).

The transfer function for NVS and RBs are dependent upon the number of versions or alternates (n).

5.4.10. N-version Software Calculations

The transfer function for NVS is given by the following expression:

$$1 - \prod_{i=1}^{C_n} (1 - \alpha_i)$$

with

n = the number of versions in the N-version Software;

Cn = (n,z) = the number of z combinations of the n element set;

$\alpha_i$ = the product of reliabilities of the ith combination required for success; and

i = 1, 2,....Cn.

z = [(n/2) + 1] if n is an even number.

z = [(n+1)/2] if n is an odd number.

The examples in this section and appendix E utilize this equation.

5.4.11. Recovery Block Calculations

The transfer function for a RB is dependent upon the number of alternates (n) that are used. This transfer function is calculated with the following equation:

$$G_1 + (1 - G_1)G_2 + (1 - G_1)(1 - G_2)G_3 + \ldots.$$

with $G_i$ = the reliability value for alternate i and

i = 1, 2, 3,...n.

The examples in this section and appendix F demonstrate the determination of the overall software reliability value with this equation and the software reliability model.

The reliability values (transfer functions) for the hybrid NVS and the RB will vary if not all of the N versions or N alternates are used. The above equations will give a higher reliability value than the actual situation in these cases. Section 5.4.16 discusses the accuracy of the reliability values for the NVS and RB and how they can be calculated to reflect the actual situation.

Although the transfer function for most of the function blocks is the reliability value, one exception to this is with rollback or any feedback block. For any feedback path, the transfer function of the equivalent block in the path is (1.0 - reliability value). The second exception is with feed-forward paths. The transfer function of the equivalent block in a feed-forward path (for example, roll-forward) is (1.0 - reliability value).

5.4.12. Examples of Potential Models: Simple, High-Level Model Example

For a sample problem involving a simple, high level model, the example given in figure 5.4-3 will be used. The single-version software will be a commonly used algorithm or process, and therefore, it will have a high reliability which is well documented and stored in the software reliability database. For this example, the reliability will be 0.9991.

The NVS will have three independent versions, running in parallel. This is a common form of NVS, but not one with the highest reliability. Through the user's tests, it is determined that this function block will have a reliability of 0.994.

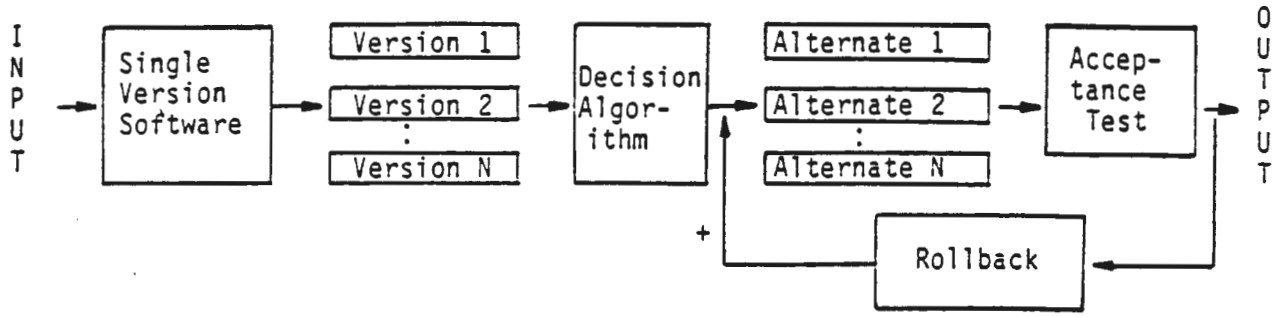



FIGURE 5.4-3. SIMPLE BLOCK DIAGRAM EXAMPLE
(from Prater, Hitt, and Eldredge, 1986)

The decision algorithm will take an average of the outputs from the NVS, excluding any version which does not meet the timing constraints. This type of decision algorithm has a high reliability since it does not have a range check or any other source for determining the validity of the output. This decision algorithm will not eliminate any erroneous outputs and will not detect the occurrence of correlated faults. The decision algorithm is simple (and consequently highly reliable), but it is not always the most desirable since its simplicity detracts from its capability of detecting errors. For this example, it is assumed that the reliability of the decision algorithm is 0.988.

The RB is a backward RB with a primary alternate and two additional, extremely simplified alternates. The RB is of a common form and its reliability can be obtained from the software reliability database. For this example, it will have a reliability value of 0.97.

The acceptance test is an output format check. This is a simplistic algorithm that is commonly used in various models. The reliability, as determined from the software reliability database, will be 0.9997.

Finally, the rollback recovers the input state of the software to its condition upon entry to the RB. This is a retrieval of the data from its memory location. The reliability of this common form of Rollback will be assumed to be available in the software reliability database. For this example, the reliability is 0.9999. This makes the transfer function for the rollback equal to (1.0 - 0.9999).

Hence, for this example,

$L_1 = (0.97) \times (0.9997) \times (+1.0 - 0.9999) = 0.0000969709$

$L_2$ through $L_n = 0$

$G_1 = (0.9991) \times (0.994) \times (0.988) \times (0.97) \times (0.9997)$

$= 0.951467$

$G_2$ through $G_K = 0$

$\Delta_1 = 1$

$\Delta = 1 - (+0.0000969709) = 0.99990302$

Therefore,

$$\text{Reliability} = \frac{(0.951467) \times (1)}{(0.99990302)} = 0.951559$$

Reliability = 0.95

5.4.13. Simple, Detailed Level Model Examples

An example of a simple, detailed level model is given in figure 5.4-4. For this example, the structure functional modules would all be single-version software and the transfer functional modules would be either forward path or positive feedback. Hence, an equivalent block diagram for this detailed block diagram is given in figure 5.4-5.

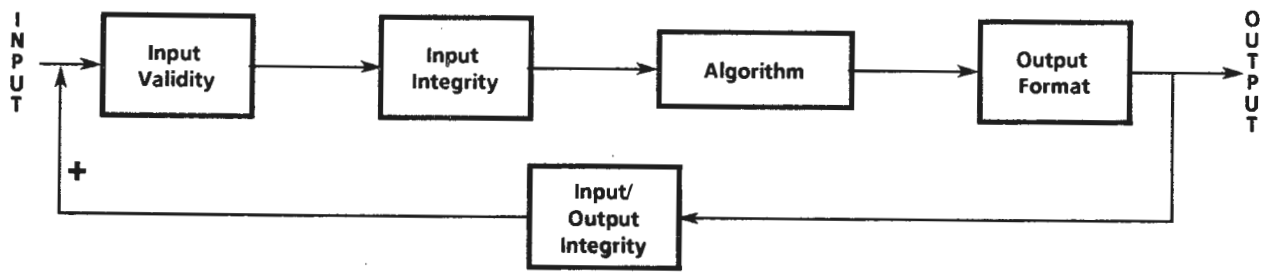



FIGURE 5.4-4. DETAILED DIAGRAM FOR THE SINGLE VERSION SOFTWARE FUNCTION BLOCK OR THE DECISION ALGORITHM FUNCTION BLOCK

In figure 5.4-5, block 1 is a single-version software block which checks the input set. This is a common process, so the reliability value will be assumed to be available in the software reliability database. For this example, the reliability value for block 1 is assumed to be 0.98.

Block 2, a single-version software block, will represent an input integrity check. It will be assumed that this algorithm is common and can consequently be found in the software reliability database. For this example, the reliability value will be 0.97.

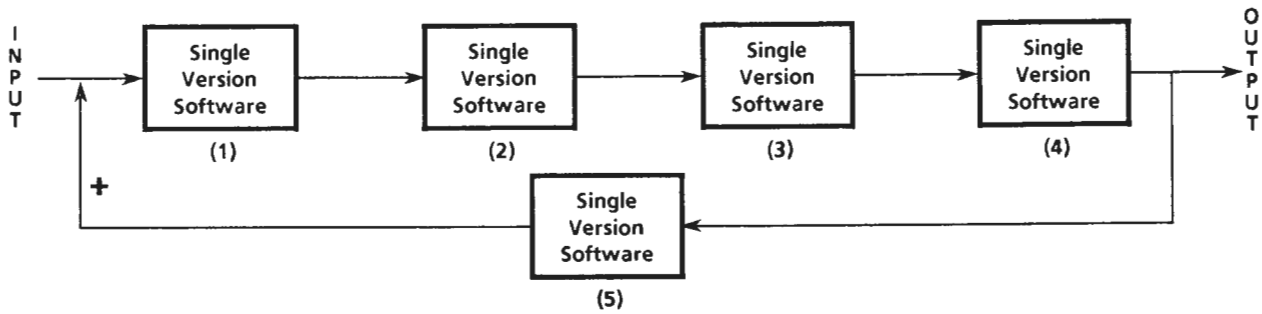



FIGURE 5.4-5. EQUIVALENT DIAGRAM INDICATING THE FUNCTIONAL STRUCTURE TO BE USED IN THE DETAILED DIAGRAM FOR A SINGLE VERSION SOFTWARE FUNCTION BLOCK

Block 3 is a single-version software block which performs an algorithm. For this example, the algorithm will be a simple one. Therefore, the reliability value for this transfer block will be assumed to be 0.992.

Block 4 will represent an output format check, performed by a single-version software block. For this example, the reliability value for this single-version software will be 0.996.

The final single-version software block, block 5, will be used to perform the input/output integrity check in which the output is checked against the input to verify the integrity of the input data. For this example, the reliability value for block 5 will be 0.964. Hence, the transfer function for this block will be (1.0 - 0.964).

Thus, for this example,

$L_1 = (0.98) \times (0.97) \times (0.992) \times (0.996) \times (1.0 - 0.964)$

$= 0.033812$

$L_2$ through $L_n = 0$

$G_1 = (0.98) \times (0.97) \times (0.992) \times (0.996) = 0.9392232$

$G_2$ through $G_K = 0$

$\Delta_1 = 1$

$\Delta = 1 - (0.033812)$

$= 0.966188$

Therefore,

$$\text{Reliability} = \frac{(0.9392232) \times (1)}{(0.966188)} = 0.9720916$$

Reliability = 0.97

5.4.14. Complex, High-Level Model Example

The complex, high level model example is shown in figure 5.4-6. Block 1 is a single-version software block. For this example, it will be a simple algorithm with a reliability value of 0.998.

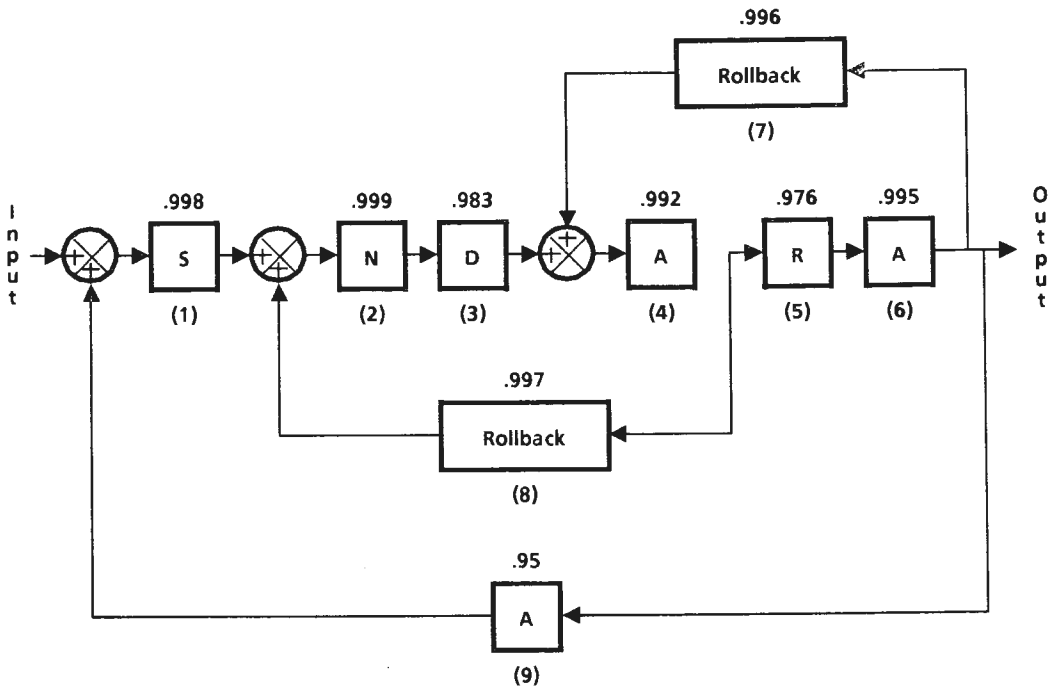
Block 2 is NVS with nine independent versions in which only three versions are run at a time. For this example, it is assumed that the reliability value for the NVS is 0.999.

Block 3 is the decision algorithm for the NVS. In this example, the decision algorithm will be median select with a reliability value of 0.983.

Block 4 is an acceptance test which will check that the range of the output from the decision algorithm is correct. If the decision algorithm failed, then the software will rollback after the acceptance test. Similarly, if the acceptance test fails, then the software will rollback to the NVS. The input to the acceptance test will be stored to accommodate for the rollback from the RB. For this example, the reliability value for the acceptance test will be 0.992.

Block 5 is a RB of the common form with a primary and two additional alternates. For this example, the reliability value of the RB will be 0.976.

Block 6 is the acceptance test for the RB. In this example, it is assumed that the reliability value for this acceptance test is 0.995.




Key:

(1) Single Version Software
(2) N-version Software
(3) Decision Algorithm
(4) Acceptance Test #1
(5) Recovery Block
(6) Acceptance Test #2
(7) Rollback #1--Backward Recovery Block
(8) Rollback #2--N-version Software in Which
Only x Versions are Used at a Time
(9) Acceptance Test #3

FIGURE 5.4-6. COMPLEX, HIGH LEVEL MODEL EXAMPLE

The rollback for the backward RB is given in block 7. This is a retrieval of the data that was stored prior to entry into Acceptance Test #1 (Block 4). For this example, the reliability value for this rollback is 0.996. Thus, the transfer function for this block is (1.0 - 0.996).

The rollback for the NVS is given in block 8. For this example, the reliability value is 0.997. Consequently, the transfer function is (1.0 - 0.997).

Finally, block (9) is an acceptance test which checks the final output against the input. For this example, Acceptance Test #3 will have a reliability value of 0.95, giving a transfer function of (1.0 - 0.95).

Hence, for this example,

Closed Loop 1 = (0.999) x (0.983) x (0.992) x (1.0 - 0.997)

= 0.0029225

Closed Loop 2 = (0.992) x (0.976) x (0.995) x (1.0 - 0.996)

= 0.0038534

Closed Loop 3 = (0.998) x (0.999) x (0.983) x (0.992) x

(0.976) x (0.995) x (1.0 - 0.95)

= 0.0472068

$\Sigma L_1$ = (Closed Loop 1) + (Closed Loop 2) + (Closed Loop 3)

= 0.0029225 + 0.0038534 + 0.0472068

= 0.0539827

$L_2$ through $L_n$ = 0

$G_1$ = (0.998) x (0.999) x (0.983) x (0.992) x (0.976) x (0.995)

= 0.944135

$G_2$ through $G_K$ = 0

$\Delta_1$ = 1

$\Delta$ = 1 - (0.0539827) = 0.9460173

Therefore,

$$\text{Reliability} = \frac{(0.944135) \times (1)}{(0.9460173)} = 0.9980103$$

Reliability = 0.998

5.4.15. Detailed Worked Examples

Appendices E and F contained detailed worked examples of both N-version and RB systems.

5.4.16. Accuracy Constraints

The accuracy and reliability of the NVS, decision algorithm, RB, and acceptance test are dependent upon the way in which these concepts are implemented. For example, NVS may be implemented in one of the following two ways:

- The x of the N-versions are run at a time, and if these versions fail for some reason, then x or fewer of the remaining (n-x) versions are run.

- A combination of x versions is run, and if these x versions fail for some reason, a different combination of x versions is run, and so on.

  NOTE: This concept is different from the first item above, because this implementation groups different combinations of x versions. The first item, however, uses x versions, and if they fail, the x versions are in essence thrown out, and a completely new group of x versions are used; not a new combination of x versions, but x completely new versions.

Some of the differences which affect the reliability and accuracy of the decision algorithm are:

- Majority vote.

- Median select.

- Average.

- The decision algorithm only considers those values which are in certain range and then it uses one of the methods above.

A few of the items which affect the RB's accuracy and reliability are:

- Backward.

  - How far it rolls back.

  - The number of alternates available.

- Forward.

  - How far it rolls forward.

  - The accuracy of the value(s) assigned prior to the roll.

Some of the variables affecting the accuracy and reliability of the acceptance test are:

- The range of the values accepted.

- The rate of change determination for the variables.

- The format of the data.

Needless to say, all of these implementation characteristics must be considered and will affect the accuracy of the reliability values. With this software reliability model design, the accuracy is improved since these considerations can be taken into account in the assignment of reliability values (or accuracy or safety or availability values) to the function blocks.

5.4.17. Accuracy

The accuracy of the software reliability model will depend upon the accuracy of the individual values that are used as the transfer functions for each of the function blocks. In most cases, the accuracy of a model with detailed function blocks will be better than the high level software reliability model since the accuracy of the individual transfer functions will be improved.

The accuracy of the reliability values that are used for the function blocks' transfer functions will depend upon the method used to obtain such values. If the values are obtained from the software reliability database, then the accuracy of these values are indicated in the technical reports for the research method that determined the values. If the values are obtained through the use of a different software reliability model, then the accuracy is dependent on the type of model used. Regardless, accuracy values can be obtained for any and all of the transfer functions, and therefore, an accuracy for the overall software reliability values can be calculated.

5.4.18. Accuracy of the Hybrid N-version Software

The accuracy of the hybrid NVS reliability value (or safety or availability values) depends upon the number of versions that are actually used. Usually, if the software is run and only y of the N-versions is used, then the reliability value (or safety or availability values) has been overrated by considering the additional (n-y) versions in the calculation of the transfer function for the hybrid NVS. Certainly, if it is known that only y of the N-versions are actually being used, then the calculation of the transfer function for the NVS should only include those y versions.

5.4.19. Accuracy of the Recovery Block

The RB, by definition, consists of n alternates. Of these n alternates, only one is run at a time, and only if that alternate fails will the software rollback and run the next alternate. Hence, if less than the n alternates are actually used, the reliability (or safety or availability) of the RB will generally decrease. Furthermore, this decrease in the RB's transfer function (reliability, safety, or availability value) will cause a decrease in the overall software reliability value (or safety or availability value), calculated with the software reliability model.

5.4.20. Safety

Safety is concerned with the state in which the function block fails (an error is detected), but the function block is error free. Although a function block

is considered to be unsafe when the system is unreliable, safety also covers this extra state. Hence,

safety = [(the probability that an error exists and it is detected) + (the probability that no error exists)]

or

safety = [1 - (the probability that an error exists and it is not detected)]

while

reliability = [(the probability that an error exists and it is detected) + (the probability that no error exists and no error is detected)]

or

reliability = {1 - [(the probability that an error exists, but the error is not detected) + (the probability that no error exists, but an error is detected)]}

Therefore, safety ≥ reliability.

The software reliability model can also be used to determine the safety of the system. Replacing the reliability values with safety values in each function block yields the safety of the overall system.

5.4.21. Availability

As with reliability and safety, availability can be determined through the use of the software reliability model. To do so, the availability values should be used for each function block in the determination of the overall transfer function. By using availability values instead of reliability values, the resultant value will be the availability of the overall system.

The availability values are determined as follows:

availability = [The probability that an error exists, it is detected, and it is corrected) + (the probability that no error exists and no error is detected)]

or

availability = {1 - [(the probability that an error exists, it is detected, but it is not corrected) + (the probability that an error exists and no error is detected) + (the probability that no error exists and an error is detected)]}

Therefore, $0 \leq$ availability $\leq 1.0$. By comparison with reliability and safety, availability $\leq$ reliability $\leq$ safety.

5.4.22. Response to Undesired Events

The software reliability model described herein assumes independence between the function blocks. This model neglects the existence of:

- Multiple faults which produce dissimilar outputs but are induced by the same input conditions, or
- Related software design faults causing identical incorrect outputs.

The errors that are manifested by these faults are known as coincident errors and cause a degradation in the reliability (or safety or availability). Therefore, to improve the accuracy of the software reliability model, the coincident errors must be considered. This might be done with an analysis similar to that suggested by Eckhardt and Lee (1985). The analysis makes the assumptions that (1) the input series $X_1$, $X_2$,...., is stationary and independent and (2) the versions of software components are designed independently (Eckhardt & Lee, 1985).

When evaluating the probability of coincident errors, the area of concern is the NVS. This analysis is interested in the probability that z or more of the functions fail at the same time, with $z = (n/2)$ if n is even and $z = [(n + 1)/2]$ if n is odd. The following analysis will give a conservative estimate (maximum possible) of the probability of coincident errors for the NVS. This value might be subtracted from the transfer function of the NVS block to produce a conservative value (minimum) of the reliability (or safety or availability) of the NVS and consequently a conservative estimate (minimum) of the overall software reliability value (or safety or availability value).

The following equation gives the maximum probability of coincident errors (E).

$$E = \binom{n}{z}^{*} (1-G_L)^{z} + \binom{n}{z+1}^{*} (1-G_L)^{z+1} +$$

$$\binom{n}{z+2}^{*} (1-G_L)^{z+2} + \ldots + \binom{n}{n}^{*} \binom{n}{1-G_L}^{n}$$

with n = the number of versions in the NVS:

$G_i$ = the reliability (or safety or availability) value for versions i;

i = 1, 2,...n;

$G_L$ = the largest reliability (or safety or availability) value among the group of r versions being evaluated;

z = (n/2) if n is an even number;

$z$ = $[(n + 1)/2]$ if n is an odd number;

$\binom{n}{r}$ = the number of r combinations of an n element set;

$\binom{n}{r}^*$ = the actual r combinations of $(1 - G_i)$ values for the different versions in an n element set of versions.

5.4.23. Assumptions

The assumptions that are frequently made with the various software methods are described below, along with the reasons for such assumptions. These assumptions are grouped logically below the corresponding software type to benefit the reader. In the development of this software reliability model, it is assumed that the software of the function blocks will have complete probabilistic independence. However, paragraph 5.4.21 tries to compensate for the deficiencies resulting from this basic assumption.

Single-version Software

- Errors are not always corrected when detected and new errors may be spawned when correcting errors.
- The time to remove a failure is considered to be negligible and is ignored.
- Inputs which exercise the program are randomly selected.
- The failure rate at any time is proportional to the current number of errors remaining in the program (Prater, Hitt and Eldredge, 1986).

N-version Software

- To benefit from increased reliability, N-version assumes that the probability of a common fault among the versions is extremely low.
- When a fault is determined, the damage incurred is limited to the encapsulation of the individual software version and the overall function that the versions are performing.

Decision Algorithm

- For a majority vote, it is assumed that damage will be limited to the versions in the minority when the decision algorithm is invoked.
- It is possible for a majority vote to yield an incorrect result if a majority of the inputs are incorrect.

Recovery Block

- Faults will manifest themselves within a recovery region.
- The alternate versions of software components are independent; correlated faults are either eliminated or reduced to an acceptably low level.
- The n alternate blocks are independent from the acceptance tests (Hitt, Webb, and Bridgeman, 1984).

Acceptance Test

- The acceptance test will recognize the faults.

## 5.5. Scott et al.'s Model

### 5.5.1. Introduction

A second approach to fault-tolerant modeling is that of Scott et al., (1983, 1984, 1987) who developed "data domain" reliability models for the N-version method, RB method, and for an integrated approach called the Consensus RB Method. The models were based on the assumption that "modules with the same functional behavior produce identical correct answers" and that "modules with distinct error behavior produce errors that are distinct from each other as well as distinct from the correct answer." They also assumed that "considerable care would be taken to create the multiple versions using different programmers working independently. The goal of this independent development would be to produce modules with certain independent execution characteristics. For example, it is desirable that the modules have independent error probabilities, that is, the probability that an error is detected in both module A and module B at the same time is the product of the individual module error probabilities. It usually is not desirable, however, that the modules produce distinct correct answers." (Scott et al., 1984, p. 21-1.)

The data domain reliability models for all three fault-tolerant approaches are presented in paragraphs 5.5.3, 5.5.5, and 5.5.7.

### 5.5.2. Recovery Block (from Scott et al., 1984)

In the RB scheme the problem is to detect a software fault in a program, recover the machine state at the time the faulty program was entered, and execute another program that performs the same function as the faulty one. A number of independently designed programs that perform the same function must be developed. The versions are ranked. When the given task is requested, only the "best" program is executed. If a fault is detected in this program, then the "next best" one is executed, etc., until an acceptable output is obtained. If all versions are deemed faulty, an error is posted.

5.5.3. The Recovery Block Reliability Model

In the RB Reliability Model there are four distinct types of errors that can occur in a RB system. A Type 1 error occurs when any program version produces an incorrect result, but the acceptance test labels the result as correct. In a Type 2 error, the final version produces correct results, but the acceptance test erroneously determines that the results are incorrect. A Type 3 error occurs when the recovery program cannot successfully recover the input state of the previous version in preparation for executing another version. Finally, a Type 4 error occurs when the last version produces incorrect results and the acceptance test judges that the results are incorrect.

These error types are roughly equivalent to the function block states identified by Hitt, E. F., J. J. Webb, and M. S. Bridgeman in the previous section of this report. A Type 1 error [$P(A_I)$ below] is equivalent to E1, a Type 2 error [$P(R_I)$ below] to E2, a Type 3 error [$P(R_C)$ below] to E3, and a Type 4 error [$P(A_C)$ below] to E4.

The reliability model presented is developed using the probabilities of certain RB events. These events and their probabilities are:

$P(C_i)$ = PROBABILITY OF VERSION i EXECUTING CORRECTLY

$P(I_i) = 1 - P(C_i)$

$P(C_R)$ = PROBABILITY OF THE RECOVERY PROGRAM EXECUTING CORRECTLY

$P(I_R) = 1 - P(C_R)$

$P(A_I)$ = PROBABILITY OF ACCEPTING AN INCORRECT RESULT

$P(R_I)$ = PROBABILITY OF REJECTING AN INCORRECT RESULT

$= 1 - P(A_I)$

$P(R_c)$ = PROBABILITY OF REJECTING A CORRECT RESULT

$P(A_c)$ = PROBABILITY OF ACCEPTING A CORRECT RESULT

$= 1 - P(R_c)$

$P(A_B)$ = PROBABILITY OF THE JOINT EVENT A AND B

$P_{RB}(E_K,N)$ = PROBABILITY OF A TYPE K ERROR GIVEN N-VERSIONS

The probability of each of the four types of errors in a 3-version RB with perfect recovery can be found. Assuming version and acceptance test independence, the probabilities are:

$$P_{RB}(E_1,3) = P(I_1)P(A_I) + P(C_1)P(R_C)P(I_2)P(A_I) + P(I_1)P(R_I)P(I_2)P(A_I) + P(C_1)P(R_C)P(C_2)P(R_C)P(I_3)P(A_I) + P(C_1)P(R_C)P(I_2)P(R_I)P(I_3)P(A_I) + P(I_1)P(R_I)P(C_2)P(R_C)P(I_3)P(A_I) + P(I_1)P(R_I)P(I_2)P(R_I)P(I_3)P(A_I)$$

$$P_{RB}(E_2,3) = P(C_1)P(R_C)P(C_2)P(R_C)P(C_3)P(R_C) + P(I_1)P(R_I)P(C_2)P(R_C)P(C_3)P(R_C) + P(C_1)P(R_C)P(I_2)P(R_I)P(C_3)P(R_C) + P(I_1)P(R_I)P(I_2)P(R_I)P(C_3)P(R_C)$$

$$P_{RB}(E_3,3) = 0$$

$$P_{RP}(E_4,3) = P(C_1)P(R_C)P(C_2)P(R_C)P(I_3)P(R_I) + P(I_1)P(R_I)P(C_2)P(R_C)P(I_3)P(R_I) + P(C_1)P(R_C)P(I_2)P(R_I)P(I_3)P(R_I) + P(I_1)P(R_I)P(I_2)P(R_I)P(I_3)P(R_I)$$

If there is dependence among the three program versions, but no dependence between the versions and the acceptance test, the RB model becomes:

$$P_{RB}(E_1,3) = P(I_1)P(A_I) + P(C_1I_2)P(R_C)P(A_I) + P(I_1I_2)P(R_I)P(A_I) + P(C_1C_2C_3)P(R_C)^2P(A_I) + P(C_1I_2I_3)P(R_C)P(R_I)P(A_I) + P(I_1C_2I_3)P(R_I)P(R_C)P(A_I) + P(I_1I_2I_3)P(R_I)^2P(A_I)$$

$$P_{RB}(E_2,3) = P(C_1C_2C_3)P(R_C)^3 + P(C_1I_2C_3)P(R_C)^2P(R_I) + P(I_1C_2C_3)P(R_C)^2P(R_I) + P(I_1I_2C_3)P(R_I)^2P(R_C)$$

$$P_{RB}(E_3,3) = 0$$

$$P_{RB}(E_4,3) = P(I_1I_2I_3)P(R_I)^3 + P(I_1C_2I_3)P(R_I)^2P(R_C) + P(C_1I_2I_3)P(R_I)^2P(R_C) + P(C_1C_2I_3)P(R_C)^2P(R_I)$$

The probability of getting any error is:

$$P_{RB}(E,3) = P_{RB}(E_1,3) + P_{RB}(E_2,3) + P_{RB}(E_3,3) + P_{RB}(E_4,3)$$

And of course, system reliability is:

$$P_{RB} = 1 - P_{RB}(E,n)$$

### 5.5.4. N-version Programming

Another prevalent method of software fault-tolerance is the N-version Programming method. In this method, n programmers ($n > 1$) are asked to independently develop and debug a program, each working from a common specification. The n programs are placed under the control of a supervisor program that dispatches all versions of the program with the necessary input when the specified task is required. Upon completion, the outputs of the n programs are compared. Since all versions of the program are independently developed, it is assumed that the

probability of a common software error is zero. Therefore, if at least two program versions agree on an output, that output is assumed to be "correct."

5.5.5. The N-version Programming Reliability Model

In the N-version Programming Reliability Model, there are three types of errors that can occur in an N-version Programming system. A Type 1 error occurs when all outputs disagree. A Type 2 error occurs when a particular incorrect output occurs more than once. Finally, a Type 3 error occurs when there is an error in the voting procedure. The probability of a system error is then the sum of the three probabilities of a Type 1, 2, or 3 error.

The probability of a Type 1 error in a 2 of 3 N-version Programming system composed of independent program versions is:

$$P_{NVP}(E_1,3) = P(I_1)P(I_2)P(I_3) + P(C_1)P(I_2)P(I_3) + P(I_1)P(C_2)P(I_3) + P(I_1)P(I_2)P(C_3)$$

$$= P(I_1)P(I_2) + P(I_1)P(I_3) + P(I_2)P(I_3) - 2P(I_1)P(I_2)P(I_3)$$

If the program versions are not independent, then probability of a Type 1 error becomes:

$$P_{NVP}(E_1,3) = P(I_1I_2I_3)P(I_2I_3)P(I_2I_3)P(I_3) + P(C_1I_2I_3)P(I_2I_3)P(I_3) + P(I_1C_2I_3)P(C_2I_3)P(I_3) + P(I_1I_2C_3)P(I_2C_3)P(C_3)$$

$$= P(I_1I_2I_3)P(I_2I_3)P(I_3) + [1 - P(I_1I_2I_3)]P(I_2I_3)P(I_3) + P(I_1C_2I_3)[1 - P(I_2I_3)]P(I_3) + P(I_1I_2C_3)P(I_2C_3)[1 - P(I_3)]$$

If the versions are dependent, the probability of a Type 2 error, the multiple occurrence of an incorrect answer, cannot be assumed to be zero. It is still reasonable that a Type 3 error is zero. Therefore, the probability of an error in a dependent N-version Programming system is:

$$P_{NVP}(E,n) = P_{NVP}(E_1,n) + P_{NVP}(E_2,n)$$

Therefore, the reliability of a 2 of 3 N-version Programming system is:

$$R_{NVP} = 1 - P_{NVP}(E,3)$$

5.5.6. Consensus Recovery Block (from Scott et al., 1984)

The Consensus RB is a software fault-tolerant technique that combines aspects of RBs and N-version Programming. This hybrid technique attempts to overcome some serious problems in the two established fault-tolerant techniques.

The Consensus RB requires the development of n independent versions of a program, an acceptance test, and a voting procedure. The versions are ranked as in a conventional RB system. Upon invocation of the Consensus RB, all versions execute and submit their outputs to a voting procedure. Since the voting procedure assumed that there are no common faults, if two or more of the versions agree on one output, that output is designated as correct. If there

is no agreement, that is, the versions supply incorrect outputs or multiple correct outputs, then a modified RB is entered. The output of the "best" version is examined by the acceptance test. If that output is judged acceptable, it is treated as a correct output and system execution continues. If, on the other hand, the output is not accepted, the next best version's output is subjected to the acceptance test. This process continues until an acceptable output is found, or the n outputs are exhausted. Notice that there is no requirement for input state recovery since all versions execute in a parallel fashion as in an N-version Programming system, not in a serial fashion as in a conventional RB.

5.5.7. The Consensus Recovery Block Reliability Model (from Scott et al., 1984)

The probabilities of error in the Consensus RB Reliability Model are the same as those in the RB model multiplied by P(D), the probability of output disagreement. In addition, a new error, Type 5 Error, can occur if there is agreement on an incorrect output. The Type 5 Error in a Consensus RB is analogous to a Type 2 error in an N-version Programming system. Therefore, in a 3-version independent Consensus RB, the probabilities of error are:

$$P_{CRB}(E_1,3) = P(D)[P(I_1)P(A_1) + P(C_1)P(R_C)P(I_2)P(A_1) + P(I_1)P(R_I)P(I_2)P(A_I) + P(C_1)P(R_C)P(C_2)P(R_C)P(I_3)P(A_I) + P(C_1)P(R_C)P(I_2)P(R_I)P(I_3)P(A_I) + P(I_1)P(R_I)P(C_2)P(R_C)P(I_3)P(A_I) + P(I_1)P(R_I)P(I_2)P(R_I)P(I_3)P(A_I)]$$

$$P_{CRB}(E_2,3) = P(D)\ [P(C_1)P(R_C)P(C_2)P(R_C)P(C_3)P(R_C) + P(I_1)P(R_I)P(C_2)P(R_C)P(C_3)P(R_C) + P(C_1)P(R_C)P(I_2)P(R_I)P(C_3)P(R_C) + P(I_1)P(R_I)P(I_2)P(R_I)P(C_3)P(R_C)]$$

$$P_{CRB}(E_3,3) = 0$$

$$P_{CRB}(E_4,3) = P(D)\ [P(C_1)P(R_C)P(C_2)P(R_C)P(I_3)P(R_I) + P(I_1)P(R_I)P(C_2)P(R_C)P(I_3)P(R_I) + P(C_1)P(R_C)P(I_2)P(R_I)P(I_3)P(R_I) + P(I_1)P(R_I)P(I_2)P(R_I)P(I_2)P(R_I)P(I_3)P(R_I)]$$

$$P_{CRB}(E_5,3) = P(\text{AGREEMENT ON INCORRECT OUTPUT})$$

If there is dependence among the three program versions, but no dependence between the versions and the acceptance test, the Consensus RB model becomes:

$$P_{CRB}(E_1,3) = P(DI_1)P(A_I) + P(DC_1I_2)P(R_C)P(A_I) + P(DI_1I_2)P(R_I)P(A_I) + P(DC_1C_2C_3)P(R_C)^2P(A_I) + P(DC_1I_2I_3)P(R_C)P(R_I)P(A_I) + P(DI_1C_2I_3)P(R_I)P(R_C)P(A_I) + P(DI_1I_2I_3)P(R_I)^2P(A_I)$$

$$P_{CRB}(E_2,3) = P(DC_1C_2C_3)P(R_C)^3 + P(DC_1I_2C_3)P(R_C)^2P(R_I) + P(DI_1C_2C_3)P(R_C)^2P(R_I) + P(DI_1I_2C_3)P(R_I)^2P(R_C)$$

$$P_{CRB}(E_3,3) = 0$$

$$P_{CRB}(E_4,3) = P(DI_1I_2I_3)P(R_I)^3 + P(DI_1C_2I_3)P(R_I)^2P(R_C) + P(DC_1I_2I_3)P(R_I)^2P(R_C) + P(DC_1C_2I_3)P(R_C)^2P(R_I)$$

The probability of getting any error is:

$$P_{CRB}(E,3) = P_{CRB}(E_1,3) + P_{CRB}(E_2,3) + P_{CRB}(E_3,3) + P_{CRB}(E_4,3) + P_{CRB}(E_5,3)$$

And of course, system reliability is:

$$R_{CRB} = 1 - P_{CRB}(E,n)$$

5.5.8. Verification

Scott et al., 1984 and 1987, have presented data from a number of experiments conducted in a university environment. The purpose of these experiments was to validate the ability of the three proposed fault-tolerant models to accurately predict system reliability. The data from these experiments, appear to indicate that the RB method and the Consensus RB method can provide the required system reliability, and that the equations developed to represent each of the models can provide the reliability estimates, which in turn can accurately predict system reliability.

## APPENDIX A - ASSUMPTIONS UNIQUE TO FAULT COUNT SOFTWARE RELIABILITY MODELS

This appendix provides a list of the underlying assumptions that are unique to each of the Fault Count Software Reliability Models referenced in section 4 of this report.

- ASSUMPTIONS UNDERLYING THE GOEL-OKUMOTO NON-HOMOGENEOUS POISSON PROCESS MODEL

  - Expected number of software faults to be eventually detected is an unknown fixed quantity.
  - Actual number of faults to be observed is a random variable.
  - Each failure is caused by one fault, and each fault is equally likely to cause a failure during testing.
  - Number of software faults detected during each non-overlapping testing interval is independent of each other.
  - Expected number of software faults detected during $(t, t + \Delta t)$ is proportional to the expected number of software faults undetected by time t.
  - Fault removal time is negligible.
  - No new faults are introduced during the fault removal process.

- ASSUMPTIONS UNDERLYING THE GEOMETRIC POISSON MODEL

  - There is an infinite number of total errors.
  - Each fault in the program is independent of the others and each of them is equally likely to occur.
  - The errors do not have the same likelihood of detection.
  - Data are available only at discrete intervals.
  - During a fixed interval of time, the number of errors detected follows a Poisson distribution.
  - During each of these periods of time, the detection rate is constant.

- Each error discovered is immediately removed or no longer counted.
- No new fault is introduced during the correction process.

- ASSUMPTIONS UNDERLYING THE EXTENDED JELINSKI-MORANDA MODEL
  - There is a fixed number of errors in the program.
  - Each error has an equal chance of being detected.
  - The number of machine language instructions remains relatively constant.
  - The failure rate is proportional to the current error content (number of remaining errors).
  - There is a constant failure rate between consecutive errors.
  - More than one error may occur in a given debugging time period.
  - The program is not being altered except for error correction.
  - No new errors are added during the debugging process.
  - Each error has an equal chance of being detected.

- ASSUMPTIONS UNDERLYING THE MODIFIED GEOMETRIC DE-EUTROPHICATION MODEL
  - The program contains an unknown number of errors.
  - Each fault in the program is independent, and each fault is equally likely to cause a failure during testing.
  - The number of faults detected in any time interval is independent of the number of faults detected in any other time interval.
  - The program is not being altered except for error correction.
  - The error correction time is negligible. Each error discovered is immediately removed.
  - No new errors are added during the debugging process.

- ASSUMPTIONS UNDERLYING THE SHOOMAN EXPONENTIAL MODEL

  - The initial number of errors in a program is a constant.
  - The number of errors remaining in the program decreases as errors are corrected.
  - Each error has an equal chance of being detected.
  - Software errors occur with a probability distribution of

$$f(t) = \lambda \exp(-\lambda t)$$

  - The error detection rate (failure rate) is proportional to the number of remaining errors.
  - The total number of machine language instructions remains constant.
  - Operational software errors occur due to the occasional traversing of a portion of the program in which a software bug is hidden.

## APPENDIX B - ASSUMPTIONS UNIQUE TO TIME-BETWEEN-FAILURE SOFTWARE RELIABILITY MODELS

This appendix provides a list of the underlying assumptions that are unique to each of the Time-Between-Failure Software Reliability Models referenced in section 4 of this report.

- ASSUMPTIONS UNDERLYING THE JELINSKI AND MORANDA DE-EUTROPHICATION MODEL

  - Initial fault content is an unknown fixed constant n.
  - Each fault in the program is independent of other faults, and each fault is equally likely to cause a failure during testing.
  - Times between occurrences of faults are independent of each other.
  - The software failure rate of the hazard function during a failure interval is constant and is proportional to the current fault content of the tested program.
  - A detected fault is removed with certainty at the end of each testing interval.
  - Only one fault is removed during each testing interval.
  - The fault removal time is negligible.
  - No new faults are introduced during the fault removal process.

- ASSUMPTIONS UNDERLYING THE GEOMETRIC DE-EUTROPHICATION MODEL

  - There is an infinite number of total errors.
  - Each fault in the program is independent of the others and each of them is equally likely to occur.
  - The errors do not have the same likelihood of detection.
  - The failure rate between successive errors forms a geometric progression and is constant in the interval between errors.
  - Each error discovered is immediately removed. The time to correct the detected faults is negligible.

- No new fault is introduced during the correction time.

- ASSUMPTIONS UNDERLYING THE GENERALIZED IMPERFECT DEBUGGING MODEL

  - All failures are observable and independent.

  - Testing is of uniform intensity and representative of the operational environment.

  - Inputs which exercise the program are randomly selected.

  - The failure rate at any time is proportional to the current number of errors remaining in the program.

  - The failure rate between the (i-1)th failure and the ith failure is $\lambda(t_i) = \phi[N-(i-1)]t^{\alpha-1}$.

  - The time to remove a failure is considered to be negligible and is ignored in the model.

  - Errors are not always corrected when detected, and errors may be spawned when correcting errors.

- ASSUMPTIONS UNDERLYING THE BUG-PROPORTIONAL MODEL

  - The initial number of errors in a program is a constant.

  - The number of errors remaining in the program decreases as errors are corrected.

  - Software errors are caused by the uncovering of residual bugs in a program.

  - The probability that a bug is encountered in the time interval, $\Delta t$, after t successful hours of operation is proportional to the fractional number of remaining bugs.

  - The fractional number of remaining bugs is independent of the operating time.

  - The rate of error correction is constant.

## APPENDIX C - WORKED EXAMPLE FOR THE GOEL-OKUMOTO ON-HOMOGENEOUS POISSON PROCESS MODEL

Worked example showing computations for the Goel-Okumoto NHPP Model (Fault Count). This example has been adapted from Goel (1983).

Consider the following failure counts.

| Interval No. | Cumulative No. of Failures |
|---|---|
| 1 | 2 |
| 2 | 4 |
| 3 | 5 |

The parameters to be estimated are a and b, where a is the expected number of software faults to be eventually detected; and b is a proportionality constant.

The MLE of a and b can be obtained by solving the following pair of equations.

$$a(1-e^{-bt_n}) = y_n$$

$$at_n e^{-bt_n} = \sum_{i=1}^{n} \frac{(y_i - y_{i-1})(t_i e^{-bt_i} - t_{i-1} e^{-bt_{i-1}})}{e^{-bt_{i-1}} - e^{-bt_i}},$$

These equations yield

$$\frac{y_n t_n e^{-bt_n}}{(1-e^{-bt_n})} = \sum_{i=1}^{n} \frac{(y_i - y_{i-1})(t_i e^{-bt_i} - t_{i-1} e^{-bt_{i-1}})}{e^{-bt_{i-1}} - e^{-bt_i}}$$

which can be used to obtain the MLE of b by the Newton-Raphson method.

Using the Newton-Raphson method to find $\hat{b}$:

$$\text{Let } F \equiv \sum_{i=1}^{n} \frac{(y_i - y_{i-1})(t_i e^{-bt_i} - t_{i-1} e^{-bt_{i-1}})}{e^{-bt_{i-1}} - e^{-bt_i}} - \frac{y_n t_n e^{-bt_n}}{(1 - e^{-bt_n})}$$

then

$$\frac{dF}{db} = \sum_{i=1}^{n} (y_i - y_{i-1}) \left[ \frac{(e^{-bt_{i-1}} - e^{-bt_i})(t_{i-1}^2 e^{-bt_{i-1}} - t_i^2 e^{-bt_i})}{(e^{-bt_{i-1}} - e^{-bt_i})^2} \right.$$

$$\left. - \frac{(t_i e^{-bt_i} - t_{i-1} e^{-bt_{i-1}})^2}{(e^{-bt_{i-1}} - e^{-bt_i})^2} \right] + \frac{y_n t_n^2 e^{-bt_n}}{(1 - e^{-bt_n})^2}$$

We can then compute the MLE of a as

$$a = \frac{y_n}{(1 - e^{-bt_n})}$$

For the above data set

$$n = 3,\ t_0 = 0,\ t_1 = 1,\ t_2 = 2,\ t_3 = 3$$

$$y_0 = 0,\ y_1 = 2,\ y_2 = 4,\ y_3 = 5$$

The computation of MLE to Estimate a and b is as follows:

Iteration 1. Let an initial value of b = .01; i.e., $b^{(1)} = 0.01$

Then

$$F = \sum_{i=1}^{3} \frac{(y_i - y_{i-1})(t_i e^{-.01t_i} - t_{i-1} e^{-.01t_{i-1}})}{e^{-.01t_{i-1}} - e^{-.01t_i}} - \frac{y_3 t_3 e^{-.01t_3}}{(1 - e^{-.01t_3})}$$

$$= \frac{y_1 (t_1 e^{-.01t_1})}{1 - e^{-.01t_1}} + \frac{(y_2 - y_1)(t_2 e^{-.01t_2} - t_1 e^{-.01t_1})}{e^{-.01t_1} - e^{-.01t_2}}$$

$$+ \frac{(y_3 - y_2)(t_3 e^{-.01t_3} - t_2 e^{-.01t_2})}{e^{-.01t_2} - e^{-.01t_3}} - \frac{y_3 t_3 e^{-.01t_3}}{(1 - e^{-.01t_3})}$$

$$= \frac{2 (e^{-.01})}{1 - e^{-.01}} + \frac{(4 - 2)(2e^{-.02} - e^{-.01})}{e^{-.01} - e^{-.02}} + \frac{(5 - 4)(3e^{-.03} - 2e^{-.02})}{e^{-.02} - e^{-.03}}$$

$$- \frac{5 \cdot 3 \cdot e^{-.03}}{(1 - e^{-.03})}$$

$$= 493.5041667 - 492.5374994$$

$$= .966667222210$$

Now,

$$\frac{dF}{db} = \sum_{i=1}^{3} (y_i - y_{i-1}) \left[ \frac{(e^{-.01t_{i-1}} - e^{-.01t_i})(t_{i-1}^2 e^{-.01t_{i-1}} - t_i^2 e^{-.01t_i})}{(e^{-.01t_{i-1}} - e^{-.01t_i})^2} \right.$$

$$\left. - \frac{(t_i e^{-.01t_i} - t_{i-1} e^{-.01t_{i-1}})^2}{(e^{-.01t_{i-1}} - e^{-.01t_i})^2} \right] + \frac{y_3 t_3^2 e^{-.01t_3}}{(1 - e^{-.01t_3})^2}$$

$$= \frac{y_1[(e^{-0.1x0} - e^{-.01})(-e^{-.01}) - e^{-.01x2}]}{(e^{-.01x0} - e^{-.01})^2}$$

$$+ \frac{(y_3 - y_2)[(e^{-.02} - e^{.03})(4e^{-.02} - 9e^{-.03}) - (3e^{-.03} - 2e^{-.02})^2]}{(e^{-.02} - e^{-.03})^2}$$

$$+ \frac{5 \cdot 9 \cdot e^{-.03}}{(1 - e^{-.03})^2}$$

$$+ \frac{(y_2 - y_1)[(e^{-.01} - e^{-.02})(e^{-.01} - 4e^{-.02}) - (2e^{-.02} - e^{-.01})^2]}{(e^{-.01} - e^{-.02})^2}$$

$$= \frac{2[(1 - e^{-.01})(-e^{-.01}) - e^{-.02}]}{(1 - e^{-.01})^2}$$

$$+ \frac{2[(e^{-.01} - e^{-.02})(-e^{-.01} - 4e^{-.02}) - (2e^{-.02} - e^{-.01})^2]}{(e^{-.01} - e^{-.02})^2}$$

$$+ \frac{1[(e^{-.02} - e^{-.03})(4e^{-.02} - 9e^{-.03}) - (3e^{-.03} - 2e^{-.02})^2]}{(e^{-.02} - e^{-.03})^2}$$

$$+ \frac{45 \cdot e^{-.03}}{(1 - e^{-.03})^2}$$

$$= -\left(\frac{2e^{-.01}}{(1 - e^{-.03})^2} + \frac{2e^{-.03}}{(e^{-.01} - e^{-.02})^2} + \frac{e^{-.05}}{(e^{-.02} - e^{-.03})^2}\right) + \frac{45e^{-.03}}{(1 - e^{-.03})^2}$$

$$= -49999.58334 + 49996.25017$$

$$= -3.33316667270$$

and

$$\Delta b^{(1)} = -F \div dF/db = -.966667222210 \div -3.33316667270$$
$$= .2900146669 > .0001$$

Let $\epsilon = 0.0001$. Since $\Delta b^{(1)} > \epsilon$, we go to iteration 2.

Iteration 2.

$$b^{(2)} = b^{(1)} + \Delta b^{(1)} = .3000146669 \simeq .3$$

$$F = \sum_{i=1}^{3} \frac{(y_i - y_{i-1})(t_i e^{-.3t_i} - t_{i-1} e^{-.3t_{i-1}})}{e^{-.3t_{i-1}} e^{-.3t_i}} - \frac{y_3 t_3 e^{-.3t_3}}{(1 - e^{-.3t_3})}$$

$$= \frac{2(e^{-.3})}{1-e^{-.3}} + \frac{2(2e^{-.6} - e^{-.3})}{e^{-.3} - e^{-.6}} + \frac{1(3e^{-.9} - 2e^{-.6})}{(e^{-.6} - e^{-.9})} - \frac{5 \cdot 3e^{-.9}}{(1 - e^{-.9})}$$

$$= 10.2906087 - 10.27600431$$
$$= .0146665525129$$

Now,

$$\frac{dF}{db} = \sum_{i=1}^{3} (y_i - y_{i-1}) \left[ \frac{(e^{-.3t_{i-1}} - e^{-.3t_i})(t_{i-1}^2 e^{-.3t_{i-1}} - t_i^2 e^{-.3_i})}{(e^{-.3t_{i-1}} - e^{-.3t_i})^2} \right.$$

$$\left. - \frac{(t_i e^{-.3t_i} - t_{i-1} e^{-.3t_{i-1}})^2}{(e^{-.3t_{i-1}} - e^{-.3t_i})^2} \right] + \frac{y_3 t_3^2 e^{-.3t_3}}{(1 - e^{-.3t_3})^2}$$

$$= \frac{2[(1 - e^{-.3})(-e^{-.3}) - e^{-.6}]}{(1 - e^{-.3})^2}$$

$$+ \frac{2[(e^{-.3} - e^{-.6})(e^{-.3} - 4e^{-.6}) - (2e^{-.6} - e^{-.3})^2}{(e^{-.3} - e^{-.6})^2}$$

$$+ \frac{1[(e^{-.6} - e^{-.9})(4e^{-.6} - 9e^{-.9}) - (3e^{-.9} - 2e^{-.6})^2]}{(e^{-.6} - e^{-.9})^2} + \frac{45e^{-.9}}{(1 - e^{-.9})^2}$$

$$= -\left( \frac{2e^{-.3}}{(1 - e^{-.3})^2} + \frac{2e^{-.9}}{(e^{-.3} - e^{-.6})^2} + \frac{e^{-1.5}}{(e^{-.6} - e^{-.9})^2} \right) + \frac{45e^{-.9}}{(1 - e^{-.9})^2}$$

$$= -55.13532562 + 51.94726586 = -3.18805975584$$

$$\Delta b^{(2)} = -F \div dF/db = -.0146665525129 \div -3.18805975584$$
$$= .004600463491$$

Since $\Delta b^{(2)} > \epsilon$, we continue the computations.

Iteration 3.

$$b^{(3)} = b^{(2)} + \Delta b^{(2)} = .3000146669 + .004600463491$$
$$= .30461513036 \approx .305$$

$$F = \sum_{i=1}^{3} \frac{(y_i - y_{i-1})(t_i e^{-.305t_i} - t_{i-1} e^{-.305t_{i-1}})}{e^{-.305t_{i-1}} - e^{-.305t_i}} - \frac{y_3 t_3 e^{-.305t_3}}{1 - e^{-.305t_3}}$$

$$= \frac{2(e^{-.305})}{1 - e^{-.305}} + \frac{2(2e^{-.610} - e^{-.305})}{e^{-.305} - e^{-.610}} + \frac{1(3e^{-.915} - 2e^{-.610})}{e^{-.610} - e^{-.915}}$$

$$- \frac{5 \cdot 3 \cdot e^{-.915}}{(1 - e^{-.915})}$$

$$= 10.04088223 - 10.04087226 = 9.96719302759 \times 10^{-6}$$

$$\frac{dF}{db} = \sum_{i=1}^{3} (y_i - y_{i-1}) \left[ \frac{(e^{-.305t_i - 1} - e^{-.305t_i})(t_{i-1}^2 e^{-.305t_{i-1}} - t_i^2 e^{-.305t_i})}{(e^{-.305t_{i-1}} - e^{-.305t_i})^2} \right.$$

$$\left. - \frac{(t_i e^{-.305t_i} - t_{i-1} e^{-.305t_{i-1}})^2}{(e^{-.305t_{i-1}} - e^{-.305t_i})^2} \right] + \frac{y_3 t_3^2 e^{-.305t_3}}{(1 - e^{-.305t_3})^2}$$

$$= \frac{2[(1 - e^{-.305})(-e^{-.305}) - e^{-.610}]}{(1 - e^{-.305})^2} +$$

$$+ \frac{2[(e^{-.305} - e^{-.610})(e^{-.305} - 4e^{-.610}) - (2e^{-.610} - e^{-.305})^2]}{(e^{-.305} - e^{-.610})^2}$$

$$+ \frac{1[(e^{-.610} - e^{-.915})(4e^{-.610} - 9e^{-.915}) - (3e^{-.915} - 2e^{-.610})^2]}{(e^{-.610} - e^{-.915})^2}$$

$$+ \frac{45 \cdot e^{-.915}}{(1 - e^{-.915})^2}$$

$$= -\left( \frac{2e^{-.305}}{(1 - e^{-.305})^2} + \frac{2e^{-.915}}{(e^{-305} - e^{-.610})^2} + \frac{e^{-1.525}}{(e^{-.610} - e^{-.915})^2} \right)$$

$$+ \frac{45e^{-.915}}{(1 - e^{-.915})^2}$$

$$= -53.470157 + 50.28643996 = -3.18371704092$$

$$\Delta b^{(3)} = -F \div dF/db = -9.96719302759 \times 10^{-6} \div -3.18371704092$$

$$= 3.1306781 \times 10^{-6}$$

Since $\Delta b^{(3)} < \epsilon$ (0.0001), we terminate further iterations.

Computation of $\hat{a}$ and $\hat{b}$

Since we met the criteria for $\Delta b^{(3)} < \epsilon$ (0.0001) we do not have to compute additional iterations, and we can compute $\hat{a}$ and $\hat{b}$ as follows:

$$\hat{b} = b^{(3)} + \Delta b^{(3)} = 3.0461826104 \times 10^{-1}$$

and

$$\hat{a} = \frac{y_3}{(1 - e^{-\hat{b}t_3})} = \frac{5}{(1 - e^{-.30461826104 \times 3})} = 8.346957421$$

Thus, for this data set

$$\hat{a} = 8.34695742$$
$$\hat{b} = 3.0461826104 \times 10^{-1}$$

After estimation of the parameters a and b the next step is to obtain performance measures as follows:

The expected number of faults detected by time t, $E[\hat{N}(t)]$, is obtained using the following formula:

$$E[\hat{N}(t)] = \hat{a}(1 - e^{-bt})$$

Let t=10. Substituting values for $\hat{a}$, $\hat{b}$, and t in the above formula we get

$$E[\hat{N}(10)] = 8.34695742 \times (1 - e^{-3.0461826104 \times 10^{-1} \times 10})$$
$$= 7.950397851 \quad .$$

The expected number of remaining faults by time t, $E[\hat{N}(t)]$, is obtained using the following formula:

$$E[\underset{\sim}{\hat{N}}(t)] = \hat{a}e^{-\hat{b}t} \quad .$$

Let t=10. Substituting values for $\hat{a}$, $\hat{b}$, and t in the above formula we get

$$E[\underset{\sim}{\hat{N}}(10)] = 8.3469742 \times (e^{-3.0461826104 \times 10^{-1} \times 10})$$
$$= 8.34695742 \times e^{-3.0461826104}$$
$$= 0.3965595689$$

The reliability at time t after nth testing interval, $\hat{R}_n(t)$, is obtained from the following formula:

$$\hat{R}_n(t) = \exp[-\hat{a}\{\exp[-\hat{b}xn] - \exp[\hat{b}x(n+t)]\}]$$

For the above example n=3. Let t be 1. After substituting values $\hat{a}$, $\hat{b}$, n, and t in the above formula we get

$$\hat{R}_3(1) = \exp[-8.34695742 \times \{\exp[-0.30461826104 \times 3]$$
$$-\exp[-0.30461826104 \times (3+1)]\} = 0.4152494843$$

The MTTF after nth testing interval, $\hat{MTTF}_n$, is obtained using the following formula:

$$\hat{MTTF}_n = \frac{1}{\hat{\lambda}(n)} = \frac{1}{\hat{a}\hat{b}\exp[-\hat{b}xn]}$$

For the above example n=3 and substituting values for $\hat{a}$ and $\hat{b}$ in the above formula we get

$$\hat{MTTF}_3 = \frac{1}{8.34695742 \times 0.30461826104 \times \exp[-0.30461826104 \times 3]}$$

$$= .9808216948$$

## APPENDIX D - WORKED EXAMPLE SHOWING COMPUTATIONS FOR THE JELINSKI-MORANDA DE-EUTROPHICATION MODEL

Worked example showing computations for the Jelinski-Moranda De-Eutrophication Model (Time Between Failures). This example has been adapted from Goel (1983).

Consider a set of failure times as follows:

| Failure No. | Time Between Failure $t_i$ |
|---|---|
| 1 | 3 |
| 2 | 30 |
| 3 | 113 |
| 4 | 31 |
| 5 | 115 |

The parameters to be estimated are N and $\Phi$, where N is the total number of initial errors in the program; and $\Phi$ is a proportionality constant.

The maximum likelihood estimate (MLE) of N can be obtained from the following equation using the Newton-Raphson method:

$$\sum_{i=1}^{n} \frac{1}{N-(i-1)} - \frac{n}{N - \frac{1}{\sum_{i=1}^{n} t_i}\left(\sum_{i=1}^{n} (i-1)t_i\right)} = 0$$

Therefore, we can use the Newton-Raphson method to find $\hat{N}$ as follows:

$$F = \sum_{i=1}^{n} \frac{1}{N-(i-1)} - \frac{n}{N - \frac{1}{\sum_{i=1}^{n} t_i}\left(\sum_{i=1}^{n} (i-1)t_i\right)}$$

then

$$\frac{dF}{dN} \equiv \frac{n}{\left\{N - \frac{1}{\sum_{i=1}^{n} t_i}\left(\sum_{i=1}^{n} (i-1)\, t_i\right)\right\}^2} - \sum_{i=1}^{n} \frac{1}{\{N-(i-1)\}^2}$$

We can then find $\hat{\Phi}$ from the following equation by substituting $\hat{N}$ for N

$$\hat{\Phi} = \frac{n}{\hat{N}\left(\sum_{i=1}^{n} t_i\right) - \sum_{i=1}^{n} (i-1)t_i}$$

For the above data set

$$\sum_{i=1}^{n} (i-1)t_i = \sum_{i=1}^{5} (i-1)t_i = t_2 + 2t_3 + 3t_4 + 4t_5$$

$$= 30 + 226 + 93 + 460 = 809$$

The computation of MLE to estimate $\hat{N}$ and $\hat{\Phi}$ is as follows:

Iteration 1. Let an initial value of $N^{(1)} = 5$

$$\text{Then } F^{(1)} = \sum_{i=1}^{5} \frac{1}{5-i+1} - \frac{5}{5 - \frac{1}{292}\cdot(809)}$$

$$= \left(\frac{1}{5} + \frac{1}{4} + \frac{1}{3} + \frac{1}{2} + 1\right) - \frac{5}{2.229452055}$$

$$= 2.283333333 - 2.242703532$$

$$= 4.062980 \times 10^{-2}$$

and

$$\frac{dF^{(1)}}{dN^{(1)}} = \frac{5}{\{5 - \frac{1}{292}(809)\}^2} - \sum_{i=1}^{5} \frac{1}{\{5-i+1\}^2}$$

$$= \frac{5}{\{2.229452055\}^2} - \left(\frac{1}{25} + \frac{1}{16} + \frac{1}{9} + \frac{1}{4} + 1\right)$$

$$\text{or } \frac{dF^{(1)}}{dN^{(1)}} = -4.57667284 \times 10^{-1}$$

Next

$$\Delta N^{(1)} = -F^{(1)} \div \frac{dF^{(1)}}{dN^{(1)}} = (4.062980 \times 10^{-2}) \div (4.57667284 \times 10^{-1})$$

$$= 0.08877584602$$

Let error $\epsilon \leq 10^{-4}$. Since $\Delta N^{(1)} \not< 10^{-4}$, we go through the next iteration.

Iteration 2. The value of N for the second iteration is

$$N^{(2)} = N^{(1)} + \Delta N^{(1)} = 5.08877584602$$

$$F^{(2)} = \sum_{i=1}^{5} \frac{1}{5.08877584602 - i + 1} - \frac{5}{5.08877584602 - \frac{809}{292}}$$

$$\text{or } F^{(2)} = 5.22788908141 \times 10^{-3}$$

and

$$\frac{dF^{(2)}}{dN^{(2)}} = \frac{5}{\{5.08877584602 - \frac{809}{292}\}^2} - \sum_{i=1}^{5} \frac{1}{\{5.08877584602 - i + 1\}^2}$$

$$= .9303743904 - 1.276022534 = -3.45648143667 \times 10^{-1}$$

Now

$$\Delta N^{(2)} = -F^{(2)} \div \frac{dF^{(2)}}{dN^{(2)}} = (5.22788908141 \times 10^{-3}) \div (3.45648143667 \times 10^{-1})$$

$$= 0.0151248869$$

Since this number is $> 10^{-4}$, we go to the next iteration.

Iteration 3. The value of N for the third iteration is

$$N^{(3)} = N^{(2)} + \Delta N^{(2)} = 5.08877584602 + 0.0151248869$$

$$= 5.10390073293$$

$$F^{(3)} = \sum_{i=1}^{5} \frac{1}{5.10390073293 - i + 1} - \frac{5}{5.10390073293 - \frac{809}{292}}$$

$$= 2.142960589 - 2.142839277 = 1.21312176929 \times 10^{-4}$$

and

$$\frac{dF^{(3)}}{dN^{(3)}} = \frac{5}{\{5.10390073293 - \frac{809}{292}\}^2} - \sum_{i=1}^{5} \frac{1}{\{5.10390073293 - i + 1\}^2}$$

$$= .9183520332 - 1.248093532 = -3.29741499203 \times 10^{-1}$$

$$\Delta N^{(3)} = -F^{(3)} \div \frac{dF^{(3)}}{dN^{(3)}} = (1.21312176929 \times 10^{-3}) \div (3.29741499203 \times 10^{-1})$$

$$= 3.679008472 \times 10^{-4}$$

Since $\Delta N^{(3)} > 10^{-4}$, we continue the computations.

Iteration 4. The value of N for the iteration is

$$N^{(4)} = N^{(3)} + \Delta N^{(3)} = 5.10390073293 + 3.679008472 \times 10^{-4}$$
$$= 5.10426863377$$

$$F^{(4)} = \sum_{i=1}^{5} \frac{1}{5.10426863377 - i + 1} - \frac{5}{5.10426863377 - \frac{809}{292}}$$

$$= 2.142501537 - .142501468 = 6.93541917229 \times 10^{-8}$$

$$\frac{dF^{(4)}}{dN^{(4)}} = \frac{5}{\{5.10426863377 - \frac{809}{292}\}^2} - \sum_{i=1}^{5} \frac{1}{\{5.10426863377 - i + 1\}^2}$$

$$= 0.9180625077 - 1.247427058 = -3.29364549966 \times 10^{-1}$$

and

$$\Delta N^{(4)} = -F^{(4)} \div \frac{dF^{(4)}}{dN^{(4)}} = (6.9354191722 \times 10^{-8}) \div (3.29364549966 \times 10^{-1})$$

$$= 2.105696916 \times 10^{-7}$$

Since $\Delta N^{(4)} < 10^{-4}$, we terminate further iterations.

Since we met the criteria for $\Delta N^{(4)} \leq 10^{-4}$, we do not have to compute additional iterations, and we can compute $\hat{N}$ and $\hat{\Phi}$ as follows:

$$\hat{N} = N^{(4)} + \Delta N^{(4)}$$

$$= 5.10426863377 + 2.105696916 \times 10^{-7}$$

or

$$\hat{N} = 5.104268844 \ ;$$

Using the above value of $\hat{N}$, we get

$$\hat{\phi} = \frac{5}{5.104268844 \cdot (292) - 809}$$

$$= \frac{5}{1490.446502 - 809} = 5/681.446502$$

or

$$\hat{\phi} = 7.337333131 \times 10^{-3}$$

Thus, for this data set, we get

$$\hat{N} = 5.104268844$$

$$\hat{\phi} = 7.337333131 \times 10^{-3}$$

After estimating the parameters $\hat{N}$ and $\hat{\Phi}$, we calculate the performance measures using these estimates as follows:

The reliability at time t after nth failure, $\hat{R}_n(t)$, is obtained using the following formula:

$$\hat{R}_n(t) = \exp[-\hat{\phi}(\hat{N} - n)t]$$

Let t=10. Substituting values for $\hat{N}$, $\hat{\Phi}$, n, and t in the above formula we get

$$\hat{R}_5(10) = \exp[-7.337333131 \times 10^{-3} \times (5.104268844 - 5) \times 10]$$

$$= .9923786386$$

The MTTF after nth failure, $\hat{MTTF}_n$, is obtained using the following formula:

$$\hat{MTTF}_n = \frac{1}{\hat{\Phi}(\hat{N} - n)}$$

Substituting values for $\hat{N}$, $\hat{\Phi}$, and n in the above formula we get

$$\hat{MTTF}_5 = \frac{1}{7.33733131 \times 10^{-3} \times (5.104268844 - 5)}$$

$$= 1307.095152$$

## APPENDIX E - WORKED EXAMPLES FOR THE N-VERSION SOFTWARE RELIABILITY DESIGN TECHNIQUES

Appendix E contains three worked examples using NVS reliability design techniques.

Example 1: Analysis of a system with NVS whose outputs go into a decision algorithm.

Example 2: Analysis of a system with NVS whose outputs are submitted to an acceptance test, and the outputs from the acceptance test are input to the decision algorithm.

Example 3: Analysis of a hybrid NVS which makes use of acceptance test, decision algorithm, and rollback.

Example 1:

Using the block diagram shown in figure E-1, a system with NVS whose outputs go into a decision algorithm will be analyzed. For this example, NVS will consist of five versions. The reliability values for the five versions and the decision algorithm are given in table E-1.

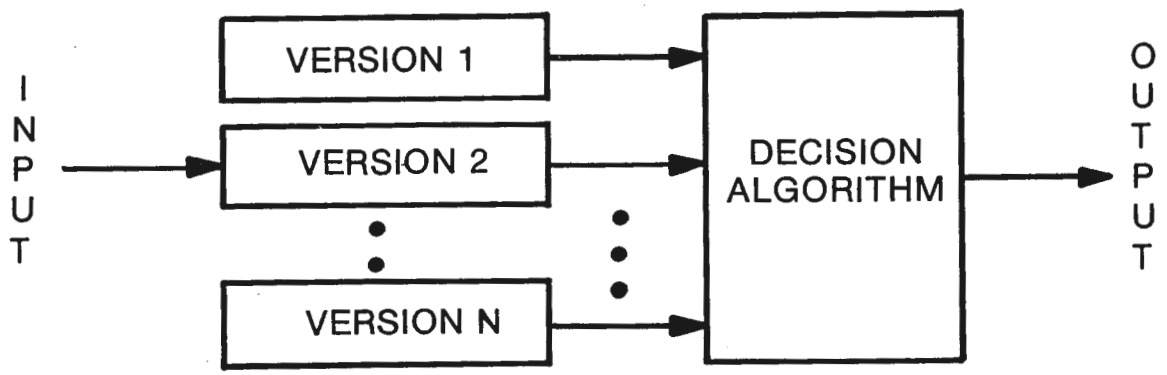



FIGURE E-1. GENERAL FORMAT FOR N-VERSION SOFTWARE

TABLE E-1. RELIABILITY VALUES FOR THE SOFTWARE COMPONENTS IN FIGURE E-1

| Software Component | Reliability Value |
|---|---|
| Version 1 | 0.77 |
| Version 2 | 0.82 |
| Version 3 | 0.65 |
| Version 4 | 0.91 |
| Version 5 | 0.89 |
| Decision Algorithm | 0.997 |

To determine the overall software reliability value for this block diagram, the transfer function for the NVS, which is dependent upon the number of versions (in this example n = 5), must first be determined. Hence, the transfer function for the NVS is

$$NVS = [1 - (1-G_1G_2G_3)(1-G_1G_2G_4)(1-G_1G_2G_5)(1-G_1G_3G_4)(1-G_1G_3G_5) \times (1-G_1G_4G_5)(1-G_2G_3G_4)(1-G_2G_3G_5)(1-G_2G_4G_5)(1-G_3G_4G_5)]$$

By substituting in the appropriate reliability values,

$$NVS = 1 - [1-(0.77)(0.82)(0.65)][1-(0.77)(0.82)(0.91)] \times [1-(0.77)(0.82)(0.89)][1-(0.77)(0.65)(0.91)] \times [1-(0.77)(0.65)(0.89)][1-(0.77)(0.91)(0.89)] \times [1-(0.82)(0.65)(0.91)][1-(0.82)(0.65)(0.89)] \times [1-(0.82)(0.91)(0.89)][1-(0.65)(0.91)(0.89)]$$

$$= [1 - (1-0.41041)(1-0.574574)(1-0.561946)(1-0.455455) \times (1-0.445445)(1-0.623623)(1-0.48503)(1-0.47437) \times (1-0.664118)(1-0.526435)]$$

$$= [1 - (0.58959)(0.425426)(0.438054)(0.544545)(0.554555) \times (0.376377)(0.51497)(0.52563)(0.335882)(0.473565)]$$

$$= 1 - 0.0005377 = 0.9994623$$

$$NVS = 0.99946$$

By applying the software reliability model,

$L_1$ through $L_n$ = 0

$G_1 = (0.99946) \times (0.997) = 0.9964616$

$G_2$ through $G_K = 0$

$\Delta_1 = 1$

$\Delta = 1$

Therefore,

$$\text{Reliability} = \frac{(0.9964616) \times (1)}{(1)} = 0.9964616$$

Reliability = 0.996

The probability of coincident errors (E) should be determined and subtracted from the transfer function for the NVS to improve the accuracy of the reliability value of the NVS and the accuracy of the overall software reliability value. For this example,

$$E = \binom{5}{3}^* (1-G_L)^3 + \binom{5}{4}^* (1-G_L)^4 + \binom{5}{5}^* (1-G_L)^5$$

The groups for $\binom{5}{3}^*$ would be

$G_1G_2G_3$, $G_1G_2G_4$, $G_1G_2G_5$, $G_1G_3G_4$, $G_1G_3G_5$, $G_1G_4G_5$, $G_2G_3G_4$, $G_2G_3G_5$, $G_2G_4G_5$, and $G_3G_4G_5$,

with respective $G_L$ values being

$G_2$, $G_4$, $G_5$, $G_4$, $G_5$, $G_4$, $G_4$, $G_5$, $G_4$, and $G_4$.

The $G_L$ values can be grouped as $G_2 + (6 \times G_4) + (3 \times G_5)$.

The groups for $\binom{5}{4}^*$ would be

$G_1G_2G_3G_4$, $G_1G_2G_3G_5$, $G_1G_2G_4G_5$, $G_1G_3G_4G_5$, and $G_2G_3G_4G_5$,

with the respective $G_L$ values being $G_4$, $G_5$, $G_4$, $G_4$, and $G_4$, giving

$4 \times G_4 + G_5$.

The group for $\binom{5}{5}^*$ is $G_1G_2G_3G_4G_5$ with $G_L = G_4$.

Therefore,

$$E = (1-G_2)^3 + 6 \times (1-G_4)^3 + 3 \times (1-G_5)^3 + 4 \times (1-G_4)^4 + (1-G_5)^4 + (1-G_4)^5$$

$$= (1-0.82)^3 + 6 \times (1-0.91)^3 + 3 \times (1-0.89)^3 + 4 \times (1-0.91)^4 + (1-0.89)^4 + (1-0.91)^5$$

$$= (0.18)^3 + 6 \times (0.09)^3 + 3 \times (0.11)^3 + 4 \times (0.09)^4 + (0.11)^4 + (0.09)^5$$

$$= 0.005832 + 0.004374 + 0.003993 + 0.0002624 + 0.0001464 + 0.0000059$$

$$E = 0.0146137$$

Subtracting E from the transfer function for the NVS gives

$$NVS = 0.9994623 - 0.0146137 = 0.9848486$$

$$NVS = 0.98485$$

$L_1$ through $L_n = 0$

$G_1 = (0.98485) \times (0.997) = 0.9818955$

$G_2$ through $G_K = 0$

$\Delta_1 = 1$

$\Delta = 1$

Hence, the adjusted reliability value is

$$\text{Reliability} = \frac{(0.9818955) \times (1)}{(1)} = 0.9818955$$

$$\text{Reliability} = 0.982$$

Example 2:

This example will evaluate the overall reliability for the system shown in figure E-2. In this example, the NVS will consist of four versions. Each of the outputs from the NVS are submitted to an acceptance test (the identical acceptance test is used for all four versions), and then the outputs from the acceptance test are input to the decision algorithm. The reliability values for each of the software components are given in table E-2.

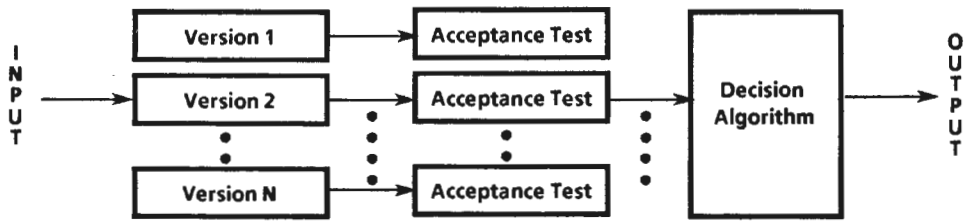



FIGURE E-2. N-VERSION SOFTWARE WITH ACCEPTANCE TESTS

TABLE E-2. RELIABILITY VALUES FOR THE SOFTWARE COMPONENTS IN FIGURE E-2

| Software Component | Reliability Value |
|---|---|
| Version 1 | 0.86 |
| Version 2 | 0.79 |
| Version 3 | 0.94 |
| Version 4 | 0.68 |
| Acceptance Test | 0.98 |
| Decision Algorithm | 0.93 |

First, the transfer function for the NVS must be determined. In this example,

$$NVS = [1 - (1-G_1G_2G_3)(1-G_1G_3G_4)(1-G_1G_2G_4)(1-G_2G_3G_4)].$$

By substituting in the appropriate reliability values,

$$NVS = \{1 - [1 - (0.86)(0.79)(0.94)][1 - (0.86)(0.94)(0.68)] \times [1 - (0.86)(0.79)(0.68)][1 - (0.79)(0.94)(0.68)]\}$$

$$= [1 - (1-0.638636)(1-0.549712)(1-0.461992)(1-0.504968)]$$

$$= [1 - (0.361364)(0.450288)(0.538008)(0.495032)]$$

$$= 1 - 0.0433368 = 0.9566632$$

$$NVS = 0.95666$$

By applying the software reliability model,

$L_1$ through $L_n = 0$

$G_1 = (0.95666) \times (0.98) \times (0.93) = 0.8718999$

$G_2$ through $G_K = 0$

$\Delta_1 = 1$

$\Delta = 1$

Therefore,

$$\text{Reliability} = \frac{(0.8718999) \times (1)}{(1)} = 0.8718999$$

$$\text{Reliability} = 0.872$$

The probability of coincident errors (E) for this example is

$$E = \binom{4}{2}^* (1-G_L)^2 + \binom{4}{3}^* (1-G_L)^3 + \binom{4}{4}^* (1-G_L)^4$$

The groups for $\binom{4}{2}^*$ are

$G_1G_2$, $G_1G_3$, $G_1G_4$, $G_2G_3$, $G_2G_4$, and $G_3G_4$.

Their respective $G_L$ values are $G_1$, $G_3$, $G_1$, $G_3$, $G_2$, and $G_3$. These values can be grouped as $(2 \times G_1) + (2 \times G_2) + (2 \times G_3)$.

The groups for $\binom{4}{3}^*$ are $G_1G_2G_3$, $G_1G_2G_4$, $G_1G_3G_4$, and $G_2G_3G_4$, with the $G_L$ values $G_3$, $G_1$, $G_3$, and $G_3$, respectively, giving $G_1 + (3 \times G_3)$.

The group for $\binom{4}{4}^*$ 4 is $G_1G_2G_3G_4$ with $G_L = G_3$.

Hence,

$$E = 2 \times (1-G_1)^2 + 2 \times (1-G_2)^2 + 2 \times (1-G_3)^2 + (1-G_1)^3 + 3 \times (1-G_3)^3 + (1-G_3)^4$$

$$= 2 \times (1-0.86)^2 + 2 \times (1-0.79)^2 + 2 \times (1-0.94)^2 + (1-0.86)^3 + 3 \times (1-0.94)^3 + (1-0.94)^4$$

$$= 2 \times (0.14)^2 + 2 \times (0.21)^2 + 2 \times (0.06)^2 + (0.14)^3 + 3 \times (0.06)^3 + (0.06)^4$$

$$= 0.0392 + 0.0882 + 0.0072 + 0.002744 + 0.000648 + 0.00001296$$

$$E = 0.1380049$$

Re-evaluating the reliability value for the NVS and the overall software reliability value gives

$$NVS = 0.9566632 - 0.00001296 = 0.9566503$$

$$NVS = 0.95665$$

$L_1$ through $L_n = 0$

$$G_1 = (0.95665) \times (0.98) \times (0.93) = 0.8718908$$

$G_2$ through $G_K = 0$

$\Delta_1 = 1$

$\Delta = 1$

Therefore, the adjusted reliability value is

$$\text{Reliability} = \frac{0.8718908 \times 1}{1} = 0.8718908$$

$$\text{Reliability} = 0.872$$

Example 3:

The hybrid NVS format, shown in figure E-3, will be evaluated in this example. The NVS here consists of three versions. The outputs of these versions are fed into a decision algorithm. If the decision algorithm fails, then the software will rollback and run through the three versions again. However, this time the outputs of the versions are input to an acceptance test prior to entry to the decision algorithm. For this example, it is assumed that the reliability values for the individual software components are as given in table E-3.

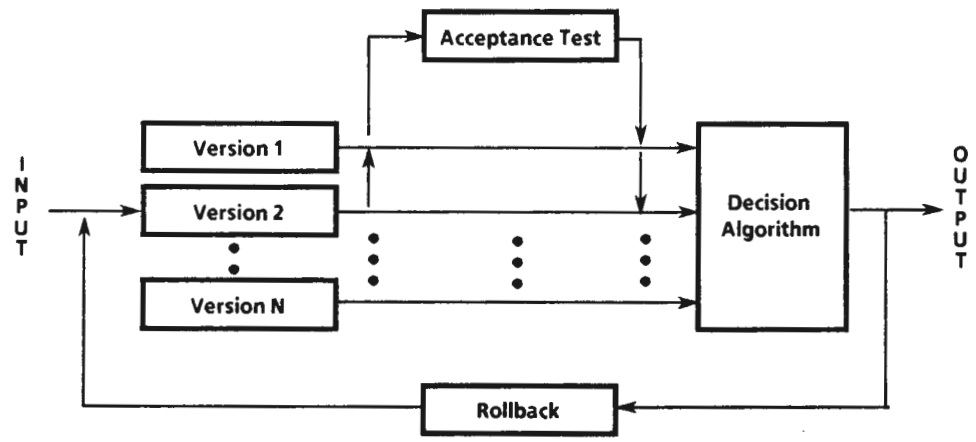



FIGURE E-3. N-VERSION SOFTWARE IN WHICH THE OUTPUTS ARE SUBJECTED TO AN ACCEPTANCE TEST IF THE DECISION ALGORITHM FAILS (Scott)

TABLE E-3. RELIABILITY VALUES FOR THE SOFTWARE COMPONENTS IN FIGURE E-3

| Software Component | Reliability Value |
|---|---|
| Version 1 | 0.67 |
| Version 2 | 0.78 |
| Version 3 | 0.89 |
| Acceptance Test | 0.86 |
| Decision Algorithm | 0.88 |
| Rollback | 0.98 |

The transfer function for the NVS, based on the data in table E-3, is

$$NVS = 1-(1-G_1G_2)(1-G_1G_3)(1-G_2G_3)$$

$$= 1-[1-(0.67)(0.78)][1-(0.67)(0.89)][1-(0.78)(0.89)]$$

$$= 1-(1-0.5226)(1-0.5963)(1-0.6942)$$

$$= 1-(0.4774)(0.4037)(0.3058)$$

$$NVS = 0.94106428$$

The variables of the software reliability model are

$$\text{Closed Loop \#1} = (0.94106428) \times (0.88) \times (1-0.98)$$

$$= 0.01656273$$

$$\text{Closed Loop \#2} = (0.94106428) \times (1-0.86) \times (0.88) \times (1-0.98)$$

$$= 0.0023187824$$

[Remember that the transfer function of the equivalent block in a feedback or feed-forward path is (1.0 - reliability value).]

$\Sigma L_1$ = Closed Loop #1 + Closed Loop #2 = 0.018881512

$\Sigma L_2$ through $\Sigma L_n = 0$

$G_1 = (0.94106428) \times (0.88) = 0.82813657$

$G_2 = (0.94106428) \times (1-0.86) \times (0.88) = 0.11593912$

$G_3$ through $G_K = 0$

$\Delta_1 = 1$

$\Delta_2 = 1$

$\Delta_3$ through $\Delta_K = 0$

$\Delta = 1 - \Sigma L_1 = 1 - 0.018881512 = 0.981118488$

Therefore,

$$\text{Reliability} = \frac{(0.82813657 \times 1) + (0.11593912 \times 1)}{(0.981118408)} = 0.9622444$$

Reliability = 0.962

To improve the accuracy of the software reliability model, the probability of coincident errors (E) might be considered. For this example,

$$E = \binom{3}{2}^* \ (1\text{-}GL)^2 + \binom{3}{3}^* \ (1\text{-}GL)^3$$

The groups for $\binom{3}{2}^*$ are $G_1G_2$, $G_1G_3$, and $G_2G_3$ with respective $G_L$ values of $G_2$, $G_3$, and $G_3$, or $G_2 + (2 \times G_3)$. The group for $\binom{3}{3}^*$ is $G_1G_2G_3$ with $G_L = G_3$.

Thus,

$$E = (1\text{-}G_2)^2 + 2 \times (1\text{-}G_3)^2 + (1\text{-}G_3)^3$$

$$= (1\text{-}0.78)^2 + 2 \times (1\text{-}0.89)^2 + (1\text{-}0.89)^3$$

$$= (0.22)^2 + 2 \times (0.11)^2 + (0.11)^3$$

$$= 0.0484 + 0.0242 + 0.001331$$

$$E = 0.073931$$

By subtracting the probability of coincident errors from the NVS transfer function, a conservative value of the reliability value for the NVS and the overall software reliability value can be determined.

$$NVS = 0.94106428 - 0.073931 = 0.86713328$$

$$\text{Closed Loop \#1} = (0.86713328) \times (0.88) \times (1\text{-}0.98)$$

$$= 0.015261546$$

$$\text{Closed Loop \#2} = (0.86713328) \times (1\text{-}0.86) \times (0.88) \times (1\text{-}0.98)$$

$$= 0.0021366164$$

$$\Sigma L_1 = 0.015261546 + 0.0021366164 = 0.017398162$$

$\Sigma L_2$ through $\Sigma L_n = 0$

$G_1 = (0.86713328) \times (0.88) = 0.76307729$

$G_2 = (0.86713328) \times (1\text{-}0.86) \times (0.88) = 0.10683082$

$G_3$ through $G_K = 0$

$\Delta_1 = 1$

$\Delta_2 = 1$

$\Delta_3$ through $\Delta_K = 0$

$\Delta = 1 - \Sigma L_1 = 1 - 0.017398162 = 0.98260184$

Therefore,

$$\text{Reliability} = \frac{(0.76307729 \times 1) + (0.10683082 \times 1)}{(0.98260184)} = 0.88531089$$

Reliability = 0.885

## APPENDIX F - WORKED EXAMPLES FOR THE RECOVERY BLOCK SOFTWARE RELIABILITY DESIGN TECHNIQUES

Appendix F contains three worked examples using RB software reliability design techniques.

Example 1: Analysis of a system which uses the general format of the backward RB.

Example 2: Analysis of a system which uses the general format of a forward RB.

Example 3: Analysis of a system which uses a variation of the forward RB format.

Example 1:

Figure F-1 shows the general format of a backward RB. For this example, the number of alternates will be four. The reliability value for each of the software components is listed in table F-1.

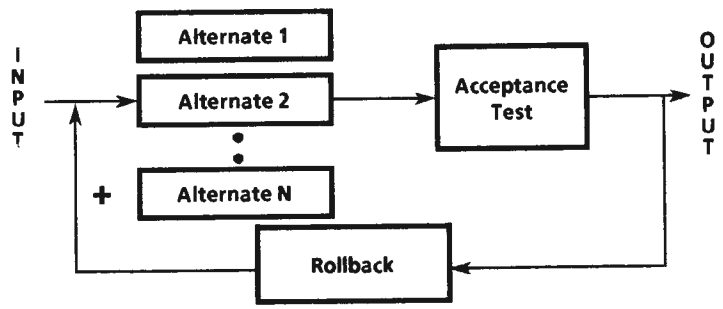



FIGURE F-1. GENERAL FORMAT OF A BACKWARD RECOVERY BLOCK

TABLE F-1. RELIABILITY VALUES FOR THE SOFTWARE COMPONENTS IN FIGURE F-1

| Software Component | Reliability Value |
|---|---|
| Alternate 1 | 0.86 |
| Alternate 2 | 0.75 |
| Alternate 3 | 0.79 |
| Alternate 4 | 0.84 |
| Acceptance Test | 0.91 |
| Rollback | 0.93 |

The transfer function for the RB is

$$RB = G_1 + (1 - G_1)G_2 + (1 - G_1)(1 - G_2)G_3 + (1 - G_1)(1 - G_2)(1 - G_3)\,G_4$$

$$= 0.86 + (1 - 0.86)(0.75) + (1 - 0.86)(1 - 0.75)(0.79) + (1 - 0.86)(1 - 0.75)(1 - 0.79)(0.84)$$

$$= 0.86 + (0.14)(0.75) + (0.14)(0.25)(0.79) + (0.14)(0.25)(0.21)(0.84)$$

$$= 0.86 + 0.105 + 0.02765 + 0.006174$$

$$RB = 0.998824$$

The variables of the software reliability model will be

$$L_1 = (0.998824) \times (0.91) \times (1.0 - 0.93) = 0.0636251$$

[Recall that the transfer function for rollback is (1.0 - reliability value).]

$L_2$ through $L_n = 0$

$G_1 = (0.998824) \times (0.91) = 0.9089298$

$G_2$ through $G_K = 0$

$\Delta_1 = 1$

$\Delta_2$ through $\Delta_K = 0$

$\Delta = 1 - 0.0636251 = 0.9363749$

Therefore,

$$\text{Reliability} = \frac{(0.9089298) \times (1)}{(0.9363749)} = 0.9706901$$

Reliability = 0.97

If only two of the four possible alternatives are actually used, the reliability of the RB will decrease. The overall software reliability will also decrease. The following calculations show this:

$RB = G_1 + (1 - G_1)G_2 = 0.86 + (1 - 0.86)(0.75)$
$= 0.86 + 0.105 = 0.965$

$L_1 = (0.965) \times (0.91) \times (1 - 0.93) = 0.0614705$

$L_2$ through $L_n = 0$

$G_1 = (0.965) \times (0.91) = 0.9089298$

$G_2$ through $G_K = 0$

$\Delta_1 = 1$

$\Delta_2$ through $\Delta_K = 0$

$\Delta = 1 - 0.0614705 = 0.9385295$

Hence,

$$\text{Reliability} = \frac{(0.9089298) \times (1)}{(0.9385295)} = 0.9684616$$

Reliability = 0.968

when only two of the alternates are used.

Example 2:

The general format of a forward RB is shown in figure F-2. This example will evaluate the overall software reliability of this figure (six alternatives will be used: one primary and five alternates), with the reliability values assigned as shown in table F-2.

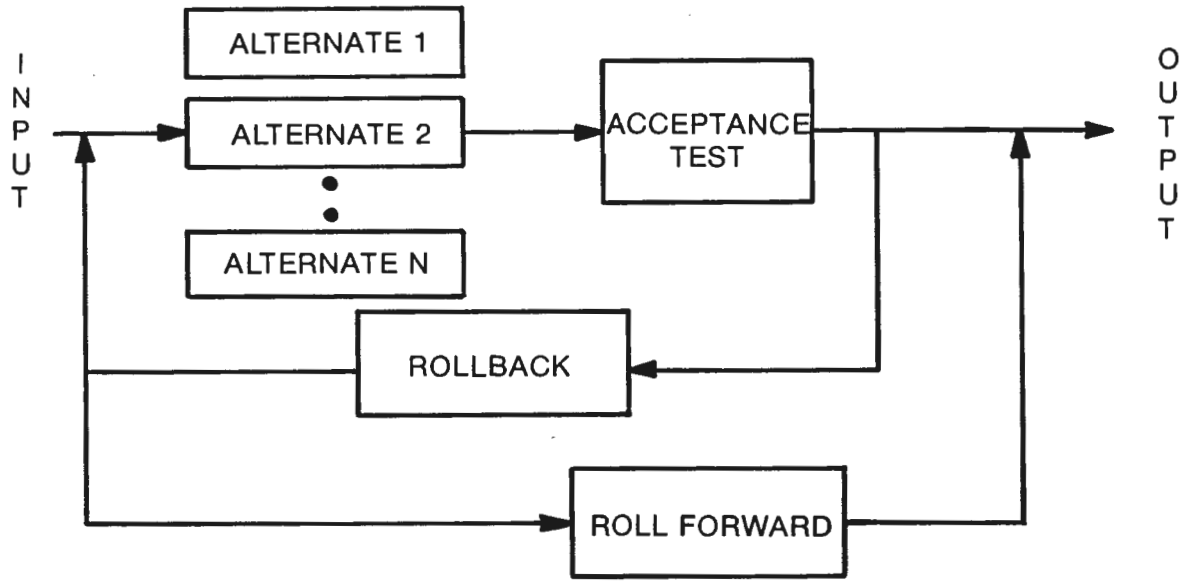



FIGURE F-2. GENERAL FORMAT OF A FORWARD RECOVERY BLOCK

TABLE F-2. RELIABILITY VALUES FOR THE SOFTWARE COMPONENTS IN FIGURE F-2

| Software Component | Reliability Value |
|---|---|
| Alternate 1 | 0.81 |
| Alternate 2 | 0.72 |
| Alternate 3 | 0.73 |
| Alternate 4 | 0.74 |
| Alternate 5 | 0.85 |
| Alternate 6 | 0.86 |
| Acceptance Test | 0.97 |
| Rollback | 0.98 |
| Roll-Forward | 0.89 |

The transfer function for the RB is

$$RB = G_1 + (1 - G_1)G_2 + (1 - G_1)(1 - G_2)G_3 + (1 - G_1)(1 - G_2)(1 - G_3)G_4 + (1 - G_1)(1 - G_2)(1 - G_3)(1 - G_4)G_5 + (1 - G_1)(1 - G_2)(1 - G_3)(1 - G_4)(1 - G_5)G_6$$

$$= 0.81 + (1 - 0.81)(0.72) + (1 - 0.81)(1 - 0.72)(0.73) + (1 - 0.81)(1 - 0.72)(1 - 0.73)(0.74) + (1 - 0.81)(1 - 0.72)(1 - 0.73)(1 - 0.74)(0.85) + (1 - 0.81)(1 - 0.72)(1 - 0.73)(1 - 0.74)(1 - 0.85)(0.86)$$

$$= 0.81 + (0.19)(0.72) + (0.19)(0.28)(0.73) + (0.19)(0.28)(0.27)(0.74) + (0.19)(0.28)(0.27)(0.26)(0.85) + (0.19)(0.28)(0.27)(0.26)(0.15)(0.86)$$

$$= 0.81 + 0.1368 + 0.038836 + 0.01062936 + 0.003174444 + 0.00048176856$$

$$= 0.99992157256$$

$$RB = 0.9999216$$

With the software reliability model,

$$L_1 = (0.9999216) \times (0.97) \times (1 - 0.98) = 0.019398479$$

[Note that the transfer function for rollback is (1.0 - reliability value).]

$L_2$ through $L_n = 0$

$$G_1 = (0.9999216) \times (0.97) = 0.96992395$$

$$G_2 = (0.9999216) \times (0.97) \times (1 - 0.98) \times (1 - 0.89) = 0.0021338327$$

[Remember that the transfer function for rollback and roll-forward is (1.0 - reliability value).]

$G_3$ through $G_K = 0$

$\Delta_1 = 1$

$\Delta_2 = 1$

$\Delta_3$ through $\Delta_K = 0$

$\Delta = 1 - 0.019398479 = 0.98060152$

Therefore,

$$\text{Reliability} = \frac{(0.96992395 \times 1) + (0.0021338327 \times 1)}{(0.98060152)}$$

$$= (0.97205778)/(0.98060152) = 0.99128725$$

$$\text{Reliability} = 0.991$$

Discussion of the Results

This result is as expected. With just the n alternates, acceptance test, and rollback, the overall reliability would be

$$\text{Reliability} = \frac{(0.9999216)(0.97)}{1 - (0.9999216)(0.97)(1 - 0.98)}$$

$$\text{Reliability} = 0.9891112 = 0.989$$

The roll-forward should increase this reliability value, as it does.

Alternative Calculations (Fewer than n alternatives)

To evaluate the effect on accuracy when fewer than the n alternates (in this example n = 6) are actually used, the reliability of this example will be evaluated with n = 3, n = 4, and n = 5.

For n = 3,

$$RB = G_1 + (1 - G_1)G_2 + (1 - G_1)(1 - G_2)G_3$$

$$= 0.81 + (1 - 0.81)(0.72) + (1 - 0.81)(1 - 0.72)(0.73)$$

$$= 0.81 + 0.1368 + 0.038836$$

$$RB = 0.985636$$

Using the software reliability model,

$$L_1 = (0.985636)(0.97)(1 - 0.98) = 0.019121338$$

$L_2$ through $L_n = 0$

$$G_1 = (0.985636)(0.97) = 0.95606692$$

$$G_2 = (0.985636)(0.97)(1 - 0.98)(1 - 0.89) = 0.0021033472$$

$G_3$ through $G_K = 0$

$\Delta_1 = 1$

$\Delta_2 = 1$

$\Delta_3$ through $\Delta_K = 0$

$\Delta = 1 - 0.019121338 = 0.98087866$

gives

$$\text{Reliability} = \frac{(0.95606692 \times 1) + (0.0021033472 \times 1)}{(0.98087866)} = 0.97684893$$

Reliability = 0.977

For n = 4,

RB = 0.81 + 0.1368 + 0.038836 + 0.01062936

RB = 0.99626536

Using the software reliability model,

$L_1 = (0.99626536)(0.97)(1 - 0.98) = 0.019327547$

$L_2$ through $L_n = 0$

$G_1 = (0.99626536)(0.97) = 0.9663774$

$G_2 = (0.99626536)(0.97)(1 - 0.98)(1 - 0.89) = 0.0021260303$

$G_3$ through $G_K = 0$

$\Delta_1 = 1$

$\Delta_2 = 1$

$\Delta_3$ through $\Delta_K = 0$

$\Delta = 1 - 0.019327547 = 0.98067245$

gives

$$\text{Reliability} = \frac{(0.9663774 \times 1) + (0.0021260303 \times 1)}{(0.98067245)} = 0.98759115$$

Reliability = 0.988

For n = 5,

$RB = 0.99626536 + 0.003174444$

$RB = 0.999439804$

Using the software reliability model,

$L_1 = (0.999439804)(0.97)(1 - 0.98) = 0.019389132$

$L_2$ through $L_n = 0$

$G_1 = (0.999439804)(0.97) = 0.96945661$

$G_2 = (0.999439804)(0.97)(1 - 0.98)(1 - 0.89) = 0.0021328045$

$G_3$ through $G_K = 0$

$\Delta_1 = 1$

$\Delta_2 = 1$

$\Delta_3$ through $\Delta_K = 0$

$\Delta = 1 - 0.019389132 = 0.98061087$

gives

$$\text{Reliability} = \frac{(0.96945661 \times 1) + (0.0021328045 \times 1)}{(0.98061087)} = 0.99080017$$

Reliability = 0.991

Discussion of the Results

Table F-3 compares the reliability values that are obtained by using fewer than n alternates in this example.

This is as expected. Actually using fewer than the n alternates, the reliability values for the RB and the overall software will generally decrease. However, as was seen in the case with n = 5, by not using the sixth alternate (which has a reliability value of 0.86 in this example), an extremely slight increase in reliability was found.

TABLE F-3. ACCURACY EFFECTS WHEN FEWER THAN n ALTERNATES ARE USED

| Number of Alternates Used | Recovery Block Reliability Value | Overall Software Reliability Value |
|---|---|---|
| n = 3 | 0.98564 | 0.977 |
| n = 4 | 0.99627 | 0.988 |
| n = 5 | 0.99944 | 0.991 |
| n = 6 | 0.99992 | 0.991 |

Example 3:

Figure F-3 shows a possible variation of a forward RB. For this example, the number of alternates will be three. The reliability value for each of the software components is listed in table F-4.

A variation of the forward RB format might be:

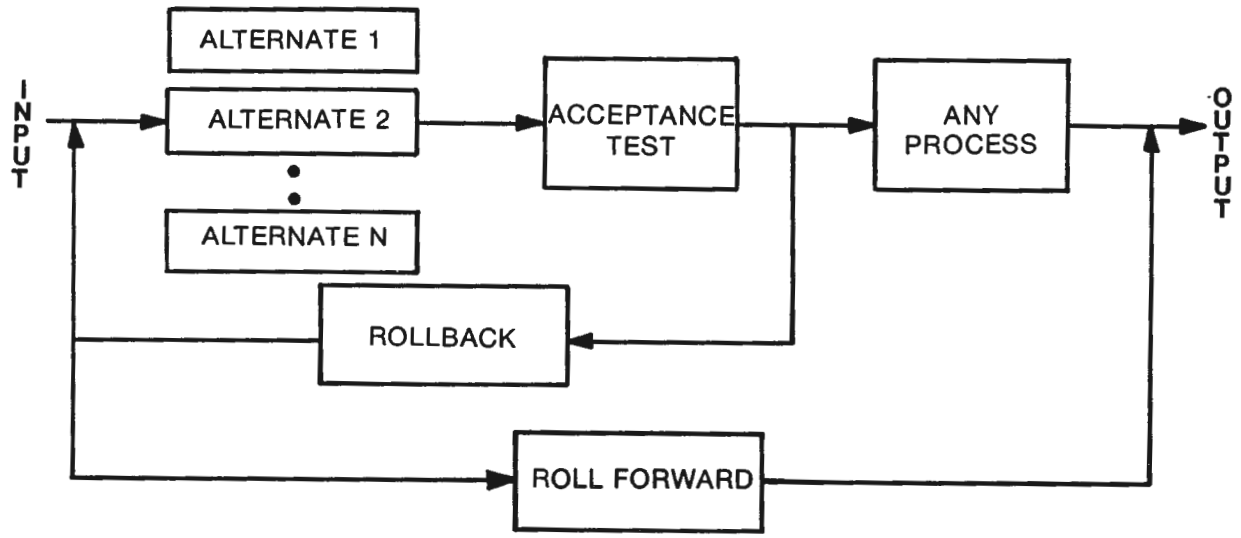



FIGURE F-3. ALTERNATE FORMAT FOR A FORWARD RECOVERY BLOCK

TABLE F-4. RELIABILITY VALUES FOR THE SOFTWARE COMPONENTS IN FIGURE F-3

| Software Component | Reliability Value |
|---|---|
| Alternate 1 | 0.80 |
| Alternate 2 | 0.70 |
| Alternate 3 | 0.90 |
| Acceptance Test | 0.98 |
| Any Process | 0.97 |
| Rollback | 0.95 |
| Roll-Forward | 0.96 |

The transfer function for the RB is

$$RB = G_1 + (1 - G_1)G_2 + (1 - G_1)(1 - G_2)G_3$$

$$= 0.80 + (1 - 0.80) \times (0.70) + (1 - 0.80) \times (1 - 0.70) \times (0.90)$$

$$= 0.80 + (0.20 \times 0.70) + (0.20 \times 0.30 \times 0.90)$$

$$= 0.80 + 0.14 + 0.054$$

$$RB = 0.994$$

The variables of the software reliability model are

$L_1 = (0.994) \times (0.98) \times (1 - 0.95) = 0.048706$

$L_2$ through $L_n = 0$

$G_1 = (0.994) \times (0.98) \times (0.97) = 0.9448964$

$G_2 = (0.994) \times (0.98) \times (1 - 0.95) \times (1 - 0.96) = 0.0019482$

$G_3$ through $G_K = 0$

$\Delta_1 = 1$

$\Delta_2 = 1$

$\Delta_3$ through $\Delta_K = 0$

$\Delta = 1 - L_1 = 1 - 0.048706 = 0.951294$

Therefore,

$$\text{Reliability} = \frac{(0.9448964 \times 1) + (0.0019482 \times 1)}{(0.951294)} = 0.99532279$$

Reliability = 0.995

To demonstrate the effect on accuracy if fewer than the n alternates (in this example n = 3) are actually used, the reliability of the RB and overall software reliability value will be re-calculated for n = 1 and n = 2.

For n = 1, RB = 0.80

With the software reliability model,

$L_1 = (0.80) \times (0.98) \times (1 - 0.95) = 0.0392$

$L_2$ through $L_n$ = 0

$G_1 = (0.80) \times (0.98) \times (0.97) = 0.76048$

$G_2 = (0.80) \times (0.98) \times (1 - 0.95) \times (1 - 0.96) = 0.001568$

$G_3$ through $G_K$ = 0

$\Delta_1 = 1$

$\Delta_2 = 1$

$\Delta_3$ through $\Delta_K$ = 0

$\Delta = 1 - L_1 = 1 - 0.0392 = 0.9608$

$$\text{Reliability} = \frac{(0.76048 \times 1) + (0.001568 \times 1)}{(0.9608)} = 0.7931391$$

Reliability = 0.793

For n = 2, RB = 0.80 + 0.14 = 0.94

With the software reliability model,

$L_1 = (0.94) \times (0.98) \times (1 - 0.95) = 0.04606$

$L_2$ through $L_n$ = 0

$G_1 = (0.94) \times (0.98) \times (0.97) = 0.893564$

$G_2 = (0.94) \times (0.98) \times (1 - 0.95) \times (1 - 0.96) = 0.0018424$

$G_3$ through $G_K$ = 0

$\Delta_1 = 1$

$\Delta_2 = 1$

$\Delta_3$ through $\Delta_K$ = 0

$\Delta = 1 - L_1 = 1 - 0.04606 = 0.95394$

$$\text{Reliability} = \frac{(0.893564 \times 1) + (0.0018424 \times 1)}{(0.95394)} = 0.93864017$$

Reliability = 0.939

Discussion of the Results

Table F-5 compares the RB reliability values and the overall software reliability values that are obtained by using n or fewer than n alternates in this example.

TABLE F-5. COMPARISON OF RELIABILITY VALUES WHEN FEWER THAN n ALTERNATES ARE USED

| Number of Alternates Used | Recovery Block Reliability Value | Overall Software Reliability Value |
|---|---|---|
| n = 1 | 0.80 | 0.793 |
| n = 2 | 0.94 | 0.939 |
| n = 3 | 0.994 | 0.995 |

BIBLIOGRAPHY


Anderson, T., and J. Knight, "Software Fault Tolerance for Real-Time Avionics Systems," AGARD Conference Proceedings No. 330, Software for Avionics, September 1982.

Eckhardt, D. E. Jr., and L. D. Lee, A Theoretical Basis for the Analysis of Redundant Software Subject to Coincident Errors, NASA Langley Research Center, NASA Technical Memorandum 86369, January 1985.

Gephart, L. S., et al., Software Reliability: Determination and Prediction, Technical Report AFFDL-TR-78-77, Air Force Flight Dynamics Laboratory, Air Force Systems Command, Wright-Patterson Air Force Base, OH, June 1978.

Goel, A. L., A Guidebook for Software Reliability Assessment, Technical Report RADC-TR-83-176 (AD A139240), Rome Air Development Center, Air Force Systems Command, Griffiss Air Force Base, NY, August 1983.

Hecht, H., "Fault-Tolerant Software," IEEE Transactions on Reliability, Vol. R-28, No. 3, pp. 227-232, August 1979.

Hitt, E. F., J. J. Webb, and M. S. Bridgman, Comparative Analysis of Fault-Tolerance Software Design Techniques, Prepared Under Contract Number NAS1-17412, February 15, 1984.

Krauson, S. S., and D. R. Baker, "Software Reliability Enhancement Techniques and Assessment Method for Embedded Computer Systems," IEEE 1982 National Aerospace and Electronics Conference, NAECON 1982, May 1982.

Larsen, W., et al., "An Overview of the Digital Avionics Assessment Activities Being Conducted by the Federal Aviation Administration at NASA-Ames Research Center," Proceedings of the AIAA/IEEE 6th Digital Avionics Systems Conference, December 1984.

Prater, S. A., E. F. Hitt, and D. Eldredge, Software Dependability Assessment Methods, DOT/FAA/CT-86/27, FAA Technical Center, Atlantic City Airport, NJ, November 1986.

Ramamoorthy, C. V., and F. B. Bastani, "Software Reliability - Status and Perspectives," IEEE Transactions on Software Engineering, Vol. SE-8, No. 4, 354-371, July 1982.

Scott, R. K., J. W. Gault, and D. F. McAllister, "Modeling Fault Tolerant Software Reliability," Proceedings of the Third Symposium on Reliability in Distributed Software and Database Systems, 1983.

Scott, R. K., J. W. Gault, and D. F. McAllister, "Fault Tolerant Software Reliability Modeling," IEEE Transactions on Software Engineering, 1987, Vol. SE-13, No. 5, pp. 582-592, May 1987.

_____, et al., "Investigating Version Dependence in Fault-Tolerant Software," AGARD Conference Proceedings No. 361, Design for Tactical Avionics Maintainability, May 1984.

Shinners, S. M., Modern Control System Theory and Application, Addison-Wesley Publishing Company, Inc., U.S.A., 1978.

Shooman, M. L., "Software Reliability: A Historical Perspective," IEEE Transactions on Reliability, Vol. R-33, No. 1, pp. 48-54, April 1984.

## GLOSSARY

ERROR. A state of the system which (in the absence of any corrective action by the system) could lead to a failure that would not be attributed to any event subsequent to the error. (More accurately known as an erroneous state.)

FAILURE. The situation when the external behavior of a system does not conform to that prescribed by the system specification.

FAULT. The adjusted cause of error.

HAZARD FUNCTION. The conditional probability that a fault is exposed in the interval t to $\Delta t$ given that the fault did not occur prior to time t.

VOTING PROCEDURE. An algorithm included in fault tolerant software which uses the consensus RB method. It compares outputs of the n independent versions and determines which outputs are correct by identifying agreements among two or more versions.

## ACRONYMS AND ABBREVIATIONS

| | |
|---|---|
| AFFDL | Air Force Flight Dynamics Laboratory |
| AGARD | Advisory Group for Aerospace Research and Development |
| cdf | Cumulative Density Function |
| DOD | Department of Defense |
| DOT | Department of Transportation |
| FAA | Federal Aviation Administration |
| MLE | Maximum Likelihood Estimates |
| MTTF | Mean Time to Failure |
| NHPP | Non-Homogeneous Poisson Process |
| NVS | N-version Software |
| pdf | Probability Density Function |
| RADC | Rome Air Development Center |
| RB | Recovery Block |

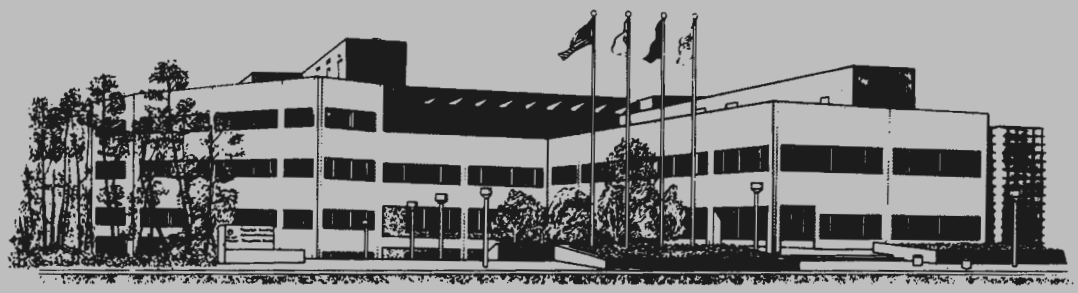
COPY — 1

U.S. Department of Transportation
Federal Aviation Administration

DOT/FAA/CT-88/10

# HANDBOOK-VOLUME II
# DIGITAL SYSTEMS VALIDATION

## CHAPTER 9
## FAULT TOLERANT SOFTWARE



**PREPARED BY:**

COMPUTER RESOURCE MANAGEMENT, INC.
950 HERNDON PARKWAY, SUITE 360
HERNDON, VIRGINIA 22070

**PREPARED FOR:**

FEDERAL AVIATION ADMINISTRATION
TECHNICAL CENTER
ATLANTIC CITY INTERNATIONAL AIRPORT, NEW JERSEY 08405

FAA WJH Technical Center
*00021057*

NOTICE

This document is disseminated under the sponsorship of the U.S. Department of Transportation in the interest of information exchange. The United States Government assumes no liability for the contents or use thereof.

The United States Government does not endorse products or manufacturers. Trade or manufacturers' names appear herein solely because they are considered essential to the objective of this report.

TABLE OF CONTENTS


| Section | | | Page |
|---|---|---|---|
| 1. | INTRODUCTION | | 9-1 |
| | 1.1. | Motivation for Fault Tolerant Software | 9-1 |
| 2. | PARTIAL SOFTWARE FAULT TOLERANCE TECHNIQUES | | 9-5 |
| | 2.1. | Robustness | 9-5 |
| | 2.2. | Rollback | 9-6 |
| | 2.3. | Issues in Partial Fault Tolerance | 9-6 |
| 3. | N-VERSION SOFTWARE | | 9-9 |
| | 3.1. | Description | 9-9 |
| | 3.2. | Concerns | 9-11 |
| | | 3.2.1. The Software Specification | 9-11 |
| | | 3.2.2. Independence of Coding Faults | 9-11 |
| | | 3.2.3. Run-time Environment | 9-12 |
| | | 3.2.4. Recovery of Failed Versions | 9-13 |
| | | 3.2.5. Maintenance and Upgrades | 9-13 |
| 4. | RECOVERY BLOCKS | | 9-15 |
| | 4.1. | Description | 9-15 |
| | 4.2. | Critical Issues | 9-21 |
| | | 4.2.1. Acceptance Tests | 9-21 |
| | | 4.2.2. Run-time Environment | 9-22 |
| | | 4.2.3. Independence of the Alternate Routine | 9-22 |
| 5. | FAULT AVOIDANCE VERSUS FAULT TOLERANCE | | 9-25 |
| 6. | INTEGRATION OF FAULT TOLERANCE TECHNIQUES | | 9-29 |
| | 6.1. | Coverage | 9-29 |
| | 6.2. | Retry Requirements | 9-29 |
| | 6.3. | Cost and Complexity | 9-29 |
| 7. | CONCLUSION | | 9-33 |
| BIBLIOGRAPHY | | | 9-35 |
| GLOSSARY | | | 9-37 |
| ACRONYMS AND ABBREVIATIONS | | | 9-38 |

## LIST OF ILLUSTRATIONS


| Figure | | Page |
|---|---|---|
| 3.1-1 | A 4-VERSION SOFTWARE IMPLEMENTATION FOR FLIGHT CONTROL | 9-10 |
| 4.1-1 | BASIC RECOVERY BLOCK | 9-16 |
| 4.1-2 | RECOVERY BLOCK FOR REAL-TIME APPLICATIONS WITH A SINGLE WATCHDOG TIMER | 9-18 |
| 4.1-3 | A REAL-TIME RECOVERY BLOCK WITH TWO WATCHDOG TIMERS | 9-20 |
| 4.1-4 | DISTRIBUTED RECOVERY BLOCK | 9-21 |


## LIST OF TABLES


| Table | | Page |
|---|---|---|
| 6-1 | SOME RELEVANT CHARACTERISTICS OF FAULT TOLERANCE TECHNIQUES | 9-30 |

# 1. INTRODUCTION

As digital systems for flight control, engine control, power distribution, and other flight critical functions have increased in sophistication and complexity, the need for fault tolerant software to overcome software reliability problems has become more apparent. This chapter describes fault tolerant software techniques and concerns in certification and validation. Section 1 explains why traditional software development methods are inadequate and where the use of fault tolerant software is required. Sections 2 through 4 discuss the three major categories of software fault tolerance: partial fault tolerance, N-version software, and recovery blocks. Section 5 describes the tradeoffs and considerations in fault tolerant versus non-fault tolerant software, and section 6 discusses the combination and integration of several fault tolerance techniques.

## 1.1. Motivation for Fault Tolerant Software

At present, the most common approach for development of critical real-time software can be termed fault avoidance, i.e., attempting to prevent any software faults in the final delivered product through disciplined software development practices, testing, and Independent Verification and Validation (IV&V).

However, fault avoidance methods have their limitations. Most test methodologies used in verification do not have well-defined termination criteria; i.e., it is usually difficult to decide when enough testing has been performed. Investigations of software test strategies (Dijkstra, 1972, Howden, 1978, and Miller, 1981) have shown that fault densities of approximately 1 per 1000 lines of source code remain even after extensive testing. The limitations of testing were capsulized in the famous saying of E.W. Dijkstra, "Program testing can be used to show the presence of bugs but never their absence." IV&V limitations include:

- The program specification on which verification occurs is generally incomplete because it cannot include enough end-to-end test cases, exception conditions, and comprehensive run time attributes, such as response time or memory utilization.

- Hardware development is frequently conducted in parallel with the software development. IV&V activities are not carried out on the processors on which the operational code will be executed.

- Changes in the system requirements, environments, or the program itself result in the need to perform extensive retesting. The amount of regression testing necessary to ensure verification after changes is uncertain and is often dictated by cost or schedule rather than technical considerations.

Such limitations prompted the developers of Bell Laboratories' No. 4 ESS (Electronic Switching System) to conclude that it

> "...is not technically or economically feasible to detect and fix all software problems in a system as large as the No. 4 ESS. Consequently, a strong emphasis has been placed on making it sufficiently tolerant of software errors to provide successful operation and fault recovery in an environment containing software problems." (Davis and Giloth, 1981)

The No. 4 ESS program referred to in this quotation occupies slightly over two million words of computer memory. Although current flight control and management software is typically only half this size, increasing use of higher order languages, the advent of the "glass cockpit," fly-by-wire control systems, full authority digital engine controllers, and other trends point in the direction of increasing software size and complexity. As these applications become more sophisticated, the need for fault tolerance to achieve higher software reliability is correspondingly greater. However, fault tolerance provisions use resources; therefore, their practical use will require tradeoffs between the need for fault tolerance and the cost incurred by its implementation. Resource utilizations will usually consist of the following components:

- Software development: Staff resources required for the additional code plus corresponding increases in documentation and test.
- Performance penalty: Increase in execution time due to tests and alternate versions that have been incorporated in the program.
- Storage expansion: Additional memory required for storage of tests, alternate versions, alternate routines (for recovery blocks), and recovery cache provisions.
- Higher maintenance costs: Additional costs due to both the inherent increase in the amount of code and the need for increased testing after changes to ensure that the fault tolerance provisions are still intact.

Thus, it is impractical to apply fault tolerance provisions throughout the entire code. Instead, they should be confined to critical modules and components. Another technique to reduce the cost of fault tolerance provisions is to generate a highly "robust" kernel that can then be utilized to direct the error detection and exception handling in other segments of the program. This kernel would include fault tolerance provisions. The "nucleus" of this kernel would contain a small amount of very highly verified software that could be used for fault detection and recovery of other portions of the kernel which, in turn, would be responsible for recovery of the balance of the system.

The discussion of software fault tolerance techniques in the following sections proceeds in the order of increasing effectiveness and computing resource requirements. Only the most effective and therefore costly fault tolerance techniques should be utilized where a failure can cause complete termination of further program execution. Where failure can only cause faulty displays which

will be readily recognized by the operator who has continued ability to call for the display of other screens, only the less costly measures will be considered.

## 2. PARTIAL SOFTWARE FAULT TOLERANCE TECHNIQUES

Partial software fault tolerance techniques have the capability to minimize rather than totally mask a fault. They detect and recover from faults without resorting to an alternate, redundant version of the function in order to produce an acceptable result. Thus, partial software fault tolerance techniques are nonredundant fault tolerance provisions. The two general classes are:

- Robustness: Substitution of an alternate value and continuation of execution if a software fault is detected.
- Rollback: Retrying the calculation in the event that a failure is detected, under the assumption that some external condition may have changed thereby resolving the anomaly.

### 2.1. Robustness

The Data Analysis Center for Software in Rome, New York, defined Robustness as the ability of the code to perform despite some violation of the assumptions in its specifications (DACS, March 1979). An example is the proper handling of out-of-range inputs without degrading the performance of functions not dependent on that input. Under such conditions, a new input may be requested (particularly if it can be supplied by a human operator), the last acceptable value can be used, or a predefined default can be assigned. In all cases, a program flag is set to notify the operator of a program exception state and to facilitate handling of the exception condition by other program elements.

Most of the techniques described as "self-checking" software are included in this definition. Self-checking features include (Yau and Cheung, 1975):

- Functionality: Range or reasonableness checks on the output.
- Control sequences: Setting an upper limit on loop iterations.
- Input data: Parity and type checks.

A distinctive feature of robustness is the protection it provides against predefined causes of software problems. An advantage of this specificity is that errors are usually detected before they can contaminate related programs or data sets. It can therefore be referred to as a technique of forward recovery. Robustness cannot usually be depended on to provide complete protection against faulty algorithms or implementations because of the possibility of faults for which no checks were incorporated.

## 2.2. Rollback

Rollback consists of the following three elements: an error detection mechanism, a protective action, and a recovery cache.

The recovery cache is the specific feature which distinguishes rollback from robustness. Error detection mechanisms can be input or output assertion checks, accounting tests, range checks, reasonableness tests, hardware alarms (e.g., divide by zero, overflow), and timeout interrupts. When the detection mechanism senses an exception condition, it usually sets a flag to notify the operator and to facilitate handling of the exception condition by using other programs. The usual protective actions are rollback to the beginning of the affected module or restart of the entire program. The recovery cache contains the data necessary for reexecution of the program or segment. During execution the program writes to a temporary memory section. The newly written values are accepted as permanent, replacing those in the recovery cache after the program has passed all checks. This methodology also prevents failures in one section of the code from contaminating other data. For these reasons, rollback has also been referred to as fault containment (Hecht and Hecht, 1986).

Rollback is effective when the failure is triggered by a temporary and unusual data value or computer control state. As the name implies, it is primarily aimed at limiting the propagation of faulty data. Rollback is a broader technique than robustness, but it does not always assure that the program will continue to run after an error has been detected. If the failure-inducing condition is short lived, the program can be expected to resume normal operation after a rollback or restart or in the next iteration. However, if the condition persists, the restoration of service will not be immediate or automatic. Because the technique depends on a recovery cache, it is a backward recovery technique.

## 2.3. Issues in Partial Fault Tolerance

Three issues to consider in the course of certification of partial fault tolerance provisions for flight critical systems are verification, performance, and coverage.

One concern of verification of robustness provisions relates to the substituted value. First, is it available, particularly when the source of that value is a real-time input such as a sensor or an operator? Second, does the substitute value allow a resumption of execution without any further downstream processing anomalies under all circumstances?

Another concern is the effectiveness of the fault detection provisions. These provisions, typically assertions in the code, must work for all the anticipated conditions and must not spuriously trigger the substitution when it is not called for. Because robustness can be relied on only for anticipated fault conditions, additional system-level analyses such as Failure Modes and Effects Analyses (FMEAs) and fault tree analyses should be conducted to ensure that major problems have not been overlooked.

The verification issues also apply to rollback. An additional concern is the reliability of the recovery cache and ensuring that data are not corrupted. The recovery cache mechanism typically involves portions of the operating system, memory management scheme (that may include hardware as well as software), and application software procedures that access the cache.

For code to be robust, the substitute value must be available within the time frame required. If this value is contained in the processor memory, then it will be available as required. However, if input from a sensor, operator, or external computer is required, then this input must occur within the required time frame, or a secondary source for substitute values must exist.

For rollback and retry, the recovery action is to restart the calculation. If the calculation is time-critical, this strategy can be problematic. In considering the use of rollback and retry, the designer and verifier must be aware of performance and response time requirements. Reasonable assurance is needed that these requirements can be met even if several retries are necessary before a result is produced. In some cases, an upper limit on the number of retries may be necessary, and a default or other value used when this upper limit is reached. A secondary concern is that nested rollback and retry provisions may cause a propagation of retries (Hitt and Prater, 1987) which could in turn result in a major system processing anomaly.

Coverage is the probability that when a fault occurs, it will be detected and recovery from the fault will be successful. Partial fault tolerance provisions provide error detection and recovery capabilities primarily for anticipated faults. An overriding question is the adequacy of fault detection mechanisms and recovery provisions. In the case of a sensor-reading, conversion, and display task which may be repeating several times per second, the argument for adequacy is fairly clear; even if the provisions fail, new data and a new display will be generated immediately. However, in other cases, e.g., in the scheduling or dispatching portions of an operating system, they may not be.

# 3. N-VERSION SOFTWARE

N-version software, also referred to as multiversion software, relies on voting as the fault tolerance mechanism. Section 3.1 describes the technique. Section 3.2 lists issues of concern to those involved with the validation and certification of this software in flight critical systems.

## 3.1. Description

N-version software uses several independently coded versions of a specified function running in parallel (at least conceptually) (Avizienis and Kelly, 1984). Their output is voted upon, and the majority answer is used as the result. The hardware analogy is passive redundancy, i.e., having n components in parallel with a voter circuit that processes all outputs. The basic strength of the approach is that the code need not be absolutely correct. If enough versions of a critical function have been written, high reliability can be achieved through logical redundancy just as hardware reliability is achieved with physical redundancy.

Figure 3.1-1 shows how a quadruple-version program might be implemented for a quadruple (hardware) redundant flight control system based on work done at Lockheed Georgia (Mulcare and Barton, 1987). Each task would run on a different processor. Input data are placed in one or more buffers. (There may be a single buffer if the four processors can access a global memory). The operating system initiates the execution of the four tasks. As the results are completed, they are passed onto the voting task which assembles the output and determines the majority value. In the quad-redundant configuration shown here, the voter must also have an algorithm for breaking 2-2 ties. For diagnostic purposes the voting task also tracks any disagreements; if a consistent pattern emerges, it notifies the operating system. The operating system then takes an appropriate action, e.g., performing diagnostics and configuring out a processor if versions running on the processor are consistently outvoted. In flight control, where mission lengths typically do not exceed more than 20 hours, a more prudent approach may be to simply retire the processor.

There have been many research projects on N-version programming in addition to the work cited above (Avizienis and Kelly, 1984, and Knight and Leveson, 1986). Two-version programs with a comparator rather than a voter have been certified for flight control and are being used on passenger aircraft. The Airbus A310 slat and flap control systems (Martin, 1982, and Hills, 1985), the Boeing 737-300, 757, and 767 (Yount, 1984), and the Airbus A320 flight control system are all operational examples of 2-version software. For these applications, the primary benefits have been as follows (Yount, 1984):

- More efficient testing: A side-by-side comparison of the two versions aids in the identification of anomalous result.

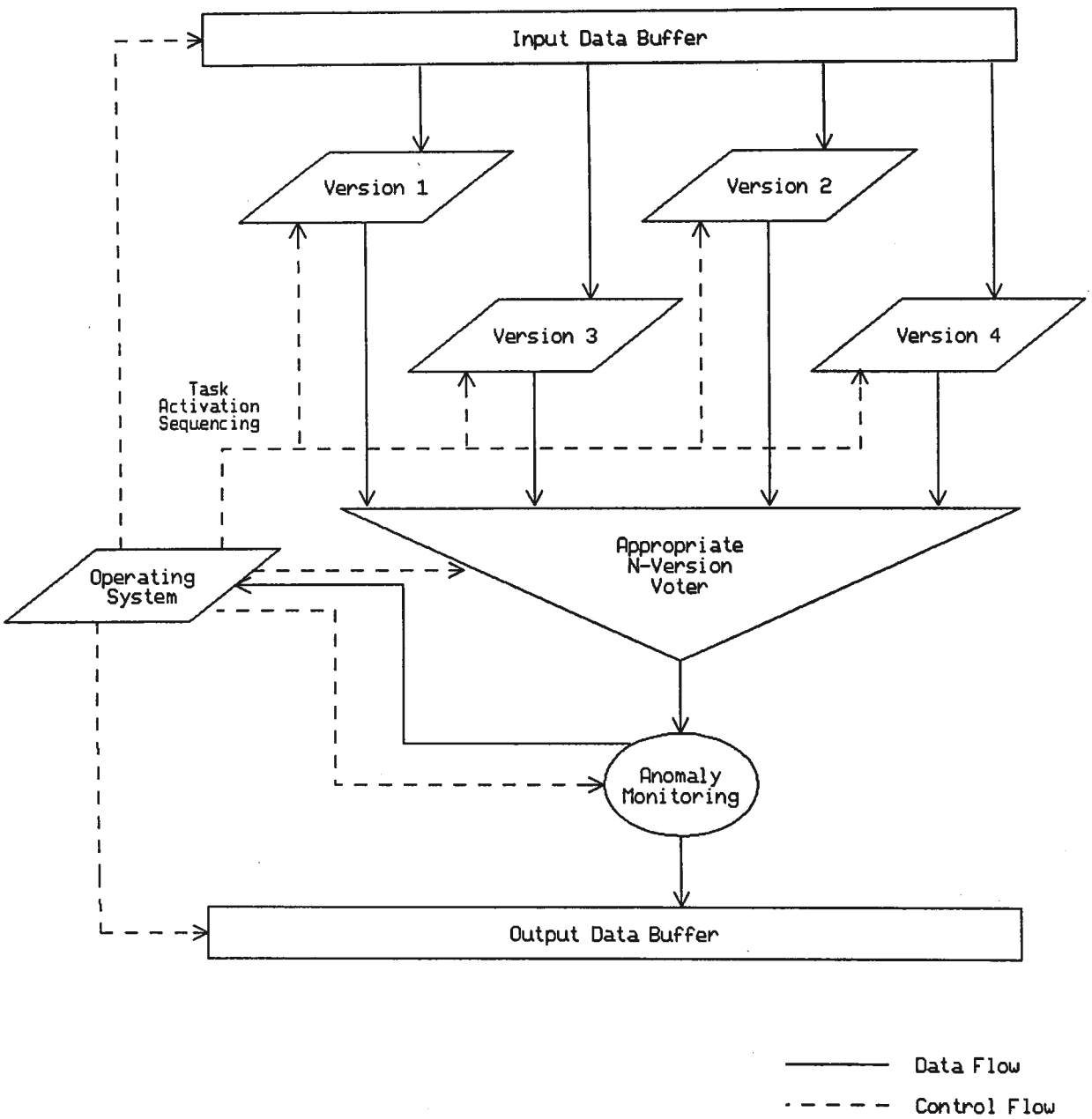



FIGURE 3.1-1. A 4-VERSION SOFTWARE IMPLEMENTATION FOR FLIGHT CONTROL

- Easier certification: The FAA gave tangible credit to Boeing for the use of diverse software.

- Acceptable system reliabilities: With data gathered from 15 airlines, the design Mean Time Between Failures (MTBF) of more than 10,000 hours for the slat and flap control systems was exceeded by approximately 30 percent.

N-version software is a fully fault tolerant technique that provides forward recovery. If a minority of versions are faulty, then the voting process causes their outputs to be disregarded, i.e., masked, and execution proceeds without any delay. This behavior is in contrast to rollback and retry which was discussed earlier or the recovery block which will be described next.

### 3.2. Concerns

Certification of N-version programs for flight critical systems entails a number of significant issues, including software specification, independence of coding faults, the execution environment, recovery of failed versions, and maintenance of multiple versions.

#### 3.2.1. The Software Specification

In conventional software development environments, many misunderstandings that a programmer may have about the application are cleared up in the course of interaction with other members of the development team. However, because the normative N-version programming framework calls for totally separate development efforts, this important but informal supplement to the normal documentation is lost. Therefore, the software specification in N-version programming is of critical importance. A flawed specification will lead to an unreliable program. Therefore, the specification must be complete, consistent, and correct.

Avizienis and Kelly (1984) suggested that a formal specification language be used. Although such a language can reduce ambiguity, it is more difficult for end-users and system-level developers to verify that the specification is a complete and correct statement of what was wanted. Many specification languages are as complicated--or even more complicated--than programming languages. Hence, even if the specifications are complete and correct, they may not be understood by all development team members and outside specialists unless they have specific training in the particular specification language being used.

#### 3.2.2. Independence of Coding Faults

The fundamental assumption of N-version programming is that coding defects are random and uncorrelated. If this assumption is correct, the likelihood of the same fault existing in all or most modules is very low. Unfortunately, this assumption has not been established by experience. A major problem is that a difficult requirement will result in a greater likelihood that all programmers will make the same type of mistake even if the developments are totally isolated from each other. These mistakes, which have been termed correlated errors or faults, are often subtle and difficult to find.

One example of the correlated fault is a proposed recent airworthiness directive issued by the FAA for the 737-300 flight control system noted earlier. There have been reports of at least 18 uncommanded flight control changes. These uncommanded changes apparently occur once every several thousand times that an aircrew resets the target altitude in the autopilot's altitude select window (Aviation Week, 1988). A second example of a correlated error was that the hardware implementations of floating point arithmetic on the VAX 11/780 computer did not allow for adequate precision for some cases in an experiment performed at the University of Virginia and the University of California, Irvine, in 1985 (Knight and Leveson, 1986).

The problem of independence among versions is compounded by the fact that all versions must produce output in exactly the same format, at the same rate, and with the same precision in order for the voting algorithm to be kept simple and reliable.

One way to relieve some of these drawbacks is the notion of directed N-version design. Instead of relying totally on independence to produce the required degree of diversity, specifications for the different versions could be written to include some algorithmic constraints. These constraints might include the order of operations, specific numerical techniques that may or may not be used, or specified differences in techniques, e.g., table look-up versus direct calculation.

### 3.2.3. Run-time Environment

As figure 3.1-1 shows, N-version programming involves more than the n programs and a voter. The run-time environment for execution of this software in a flight control system must include provisions for:

- Inputting identical data to all versions.
- Synchronizing all versions; i.e., in order to perform a valid vote, each version of the software must complete within a certain time interval, and all results must be from the current execution frame or time slice.
- Collecting results for input to the voter.
- Routing the results of the voter to the appropriate destination, e.g., an actuator motor or display.
- Data recording and diagnostics so that consistently disagreeing versions--and possibly their associated processors if they run in a multiprocessing environment--are configured out of the operational system for the duration of the mission.

The run-time environment is critical to the correct functioning of the technique and must be protected. If the software and hardware components of the environment are incorrectly designed or become corrupted during a flight an unrecoverable system failure will probably occur.

### 3.2.4. Recovery of Failed Versions

If the application requires awareness of past events; e.g., for integration or averaging, filtering, or target tracking, a failed version must have its data structures updated before participating in subsequent votes. The nature of the updating mechanism, techniques for performing the update, and synchronization techniques are application dependent. However, their correctness is critical for realizing the potential reliability benefits through restoration of a failed resource.

### 3.2.5. Maintenance and Upgrades

Questions of maintenance affect errors in a single version and across all versions. In a single version, there is a question of whether a known bug should be altered if it has not been found in other versions. Economic considerations, configuration management problems, and the concern that fixing a known bug can introduce additional unknown problems must all be addressed before a decision is made.

Management of a version-wide modification presents other problems. If the same maintenance coder modifies all versions of the software, independence is lost. On the other hand, it is impractical to have four or five different coders maintaining different versions. The practices of the developer in handling this problem can have a significant impact on the reliability of the software once the development is concluded.

# 4. RECOVERY BLOCKS

Recovery blocks are a backward recovery technique which incorporate explicit error detection and recovery provisions. In this regard, they are related to the rollback technique described in section 2. However, they include redundant logic in the form of an alternate routine which is used if an error is detected. They are therefore fully fault tolerant. Section 4.1 describes the basic recovery block and its variations. Section 4.2 lists concerns related to their validation in flight-critical software.

## 4.1. Description

Recovery blocks were first described by B. Randell of the University of Newcastle in 1975 (Randell, 1975). The three elements of the basic recovery block are:

- The primary routine which implements a given function or program specification.
- The acceptance test which determines that the requirements of the function or specification have been met after each execution of the primary routine.
- One or more alternate routines which perform the same function as the primary routine using an independent algorithm. The alternate routines are invoked if the acceptance test condition is not met.

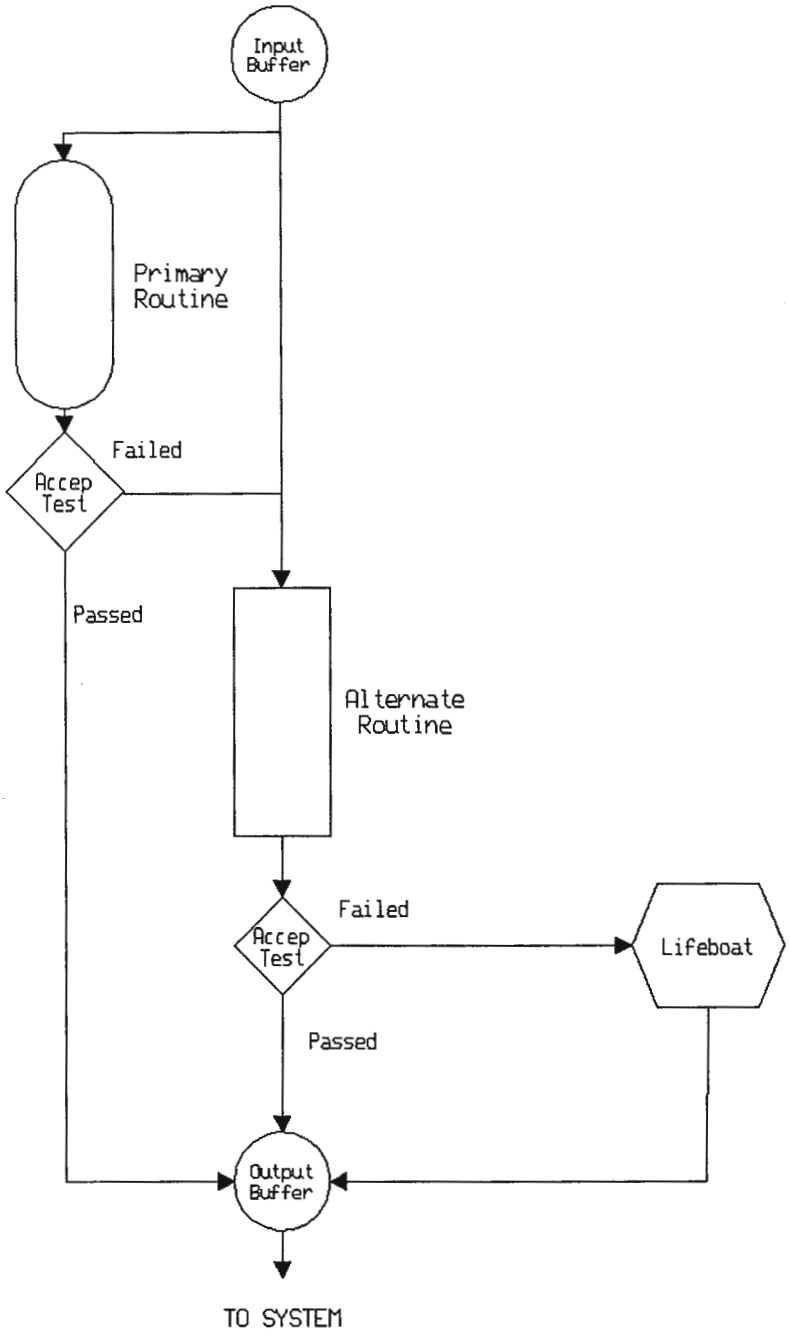
The basic structure of a recovery block with only a single alternate routine is shown in figure 4.1-1. Randell stated it formally as:

```
Ensure AT

        By P
        Else by Q

Else Abort
```

Where AT is the acceptance test, P is the primary routine for a given function, Q is the alternate, and the Abort condition invokes a higher level recovery procedure. When more than one alternate is provided, additional 'Else by...' statements (i.e., Else by R, Else by S, etc.) follow 'Else by Q'. Until the acceptance test is passed, the input data are protected in their original state by a recovery cache in a manner similar to the fault containment structure described in section 2.

Important features of the recovery block are:

- Software outside the recovery block is unaffected by which routine (primary or alternate) furnished the satisfactory result.




FIGUR E 4.1-1. BASIC RECOVERY BLOCK

- The alternate routine is executed only upon a failure of the primary routine. Thus, the alternate can use inefficient but highly failure-resistant code; this avoids a run-time penalty under normal conditions.

- The recovery block allows for a subsequent retry of the primary routine. Software failures result from the inability of the program to process a specific set of input data when in a certain computer state. Therefore, it is undesirable to make a permanent switch to the backup module after only an isolated failure in the primary routine. Returning to the primary may be desirable because of greater efficiency, versatility, or confidence. A simple diagnostic procedure can be used to switch the roles of the primary and backup modules when repeated failures of the primary occur.

- Failures of individual routines are visible from the state of a flag set by the acceptance test AT. Therefore, online monitoring and testability are readily incorporated into the recovery block. The technique also permits capture of the data associated with each failure. Hence, complete diagnostic information for later improvements of software can be made available.

In order to implement recovery blocks effectively, the primary routine, alternate routine, and acceptance test must be deliberately independent. This approach contrasts with N-version programs where independence is usually assumed to be implicit by virtue of the independent development of the versions. Deliberate independence can be achieved in a variety of ways. In mathematical and logical operations, the basic commutative, associative, and distributive properties can be used to change the order of calculation. In systems which control physical processes, it may be possible to derive a result from several different physical inputs.

In a demonstration of recovery blocks for use in flight navigation (Hitt and Prater, 1987), independence was achieved by removing some of the functionality in the primary routine. However, because no alternative to the fundamental form of the navigation equations was available, an approximation of the sine and cosine functions was used in the alternate version; the primary relied on library trigonometric routines in the Ada compiler that was being used.

Acceptance tests can often be designed to check for a plausible result rather than a correct result. For example, data processed from sensors and sent out to actuators can be checked for continuity; i.e., the difference between the value of a previous iteration and the current value should be less than some stated limit.

An additional consideration for real-time applications is to ensure that computations are completed in the required time. One possible approach is the inclusion of a "watchdog timer" which forces entry into a recovery condition if the acceptance criteria are not met within a given time. The structure of such a recovery block is shown in figure 4.1-2.

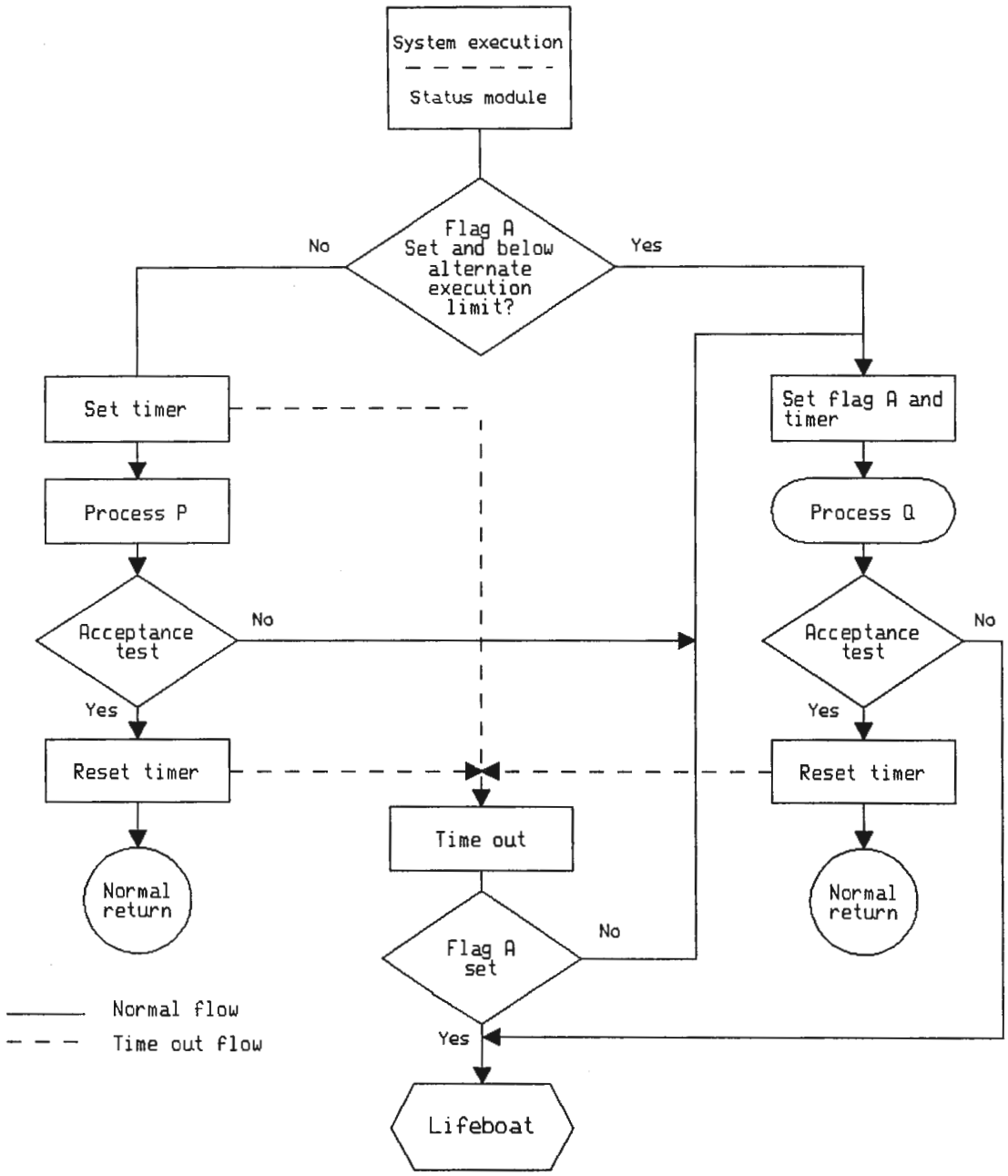



FIGURE 4.1-2. RECOVERY BLOCK FOR REAL-TIME APPLICATIONS WITH A SINGLE WATCHDOG TIMER

Before entering the recovery block, calls to the primary and alternate routines are prepared, and the allocated execution time (including an allowance for the acceptance test) is loaded into the watchdog timer. The primary call is then executed and the result of the processing is subjected to the acceptance test. If passed, a normal return results. If not passed or if the watchdog timer interrupts the primary process, the alternate routine is executed and the acceptance test is run on its results. Once again, a normal return will take place if the results pass the acceptance test. However, if the alternate routine also does not produce an acceptable result or if an alternate routine time-out occurs, then an abort-return results and recovery must take place on a higher level.

Figure 4.1-3 (Hitt and Prater, 1987) shows the flow chart for a recovery block implemented with two watchdog timers: The first is for the "minor frame," i.e., the time allotted for execution of the total recovery block. The second is for each version of the software. The function of this second watchdog timer is similar to the single watchdog timer example described in the previous paragraph.

An additional variation on recovery blocks is nesting. In a nested recovery block, a failure of the lower level primary and alternate would cause a return to the higher level recovery block with a failure indication. The higher level recovery block would then invoke its alternate higher level routine. The advantage of such a structure is that it provides additional software redundancy, and that a single higher level alternate routine may substitute for a number of secondary or tertiary alternates on a lower level. The primary cost is a performance penalty because rollback to an earlier stage in execution must occur to effect a recovery.

When both the primary and alternate routines are implemented on a single processor, the performance penalty entailed by rolling back and restarting the calculation with an alternate routine using data in the recovery cache may be unacceptable. However, because most future digital flight control designs will have several processors, it is possible to use a Distributed Recovery Block (DRB) (Kim 1984) as shown in figure 4.1-4. In the DRB, each processor contains its own copy of the primary and alternate. The processor in control of the system executes the primary routine and the backup processor(s) execute(s) the alternate routine(s) in parallel. All processors run the same acceptance test. Thus, if the primary processor fails, the backup processor's results are already available with no time lost for the retry. The use of different processors with different versions of the software unifies both hardware and software fault tolerance into a single framework of distributed systems fault tolerance. This integration provides protection against a broader scope of problems (i.e., both hardware and software failures) and has significant performance advantages. Additional research should be pursued on issues such as identifying the actions to be taken following a failure and integrating hardware status checks into the DRB (Hecht, 1988).

There is little data available on the effectiveness of recovery blocks in operational systems. However, early results indicate that increases in software MTBF on the order of two to three times could be achieved in an academic setting (Anderson, et al., 1985). The authors suggest that significantly higher

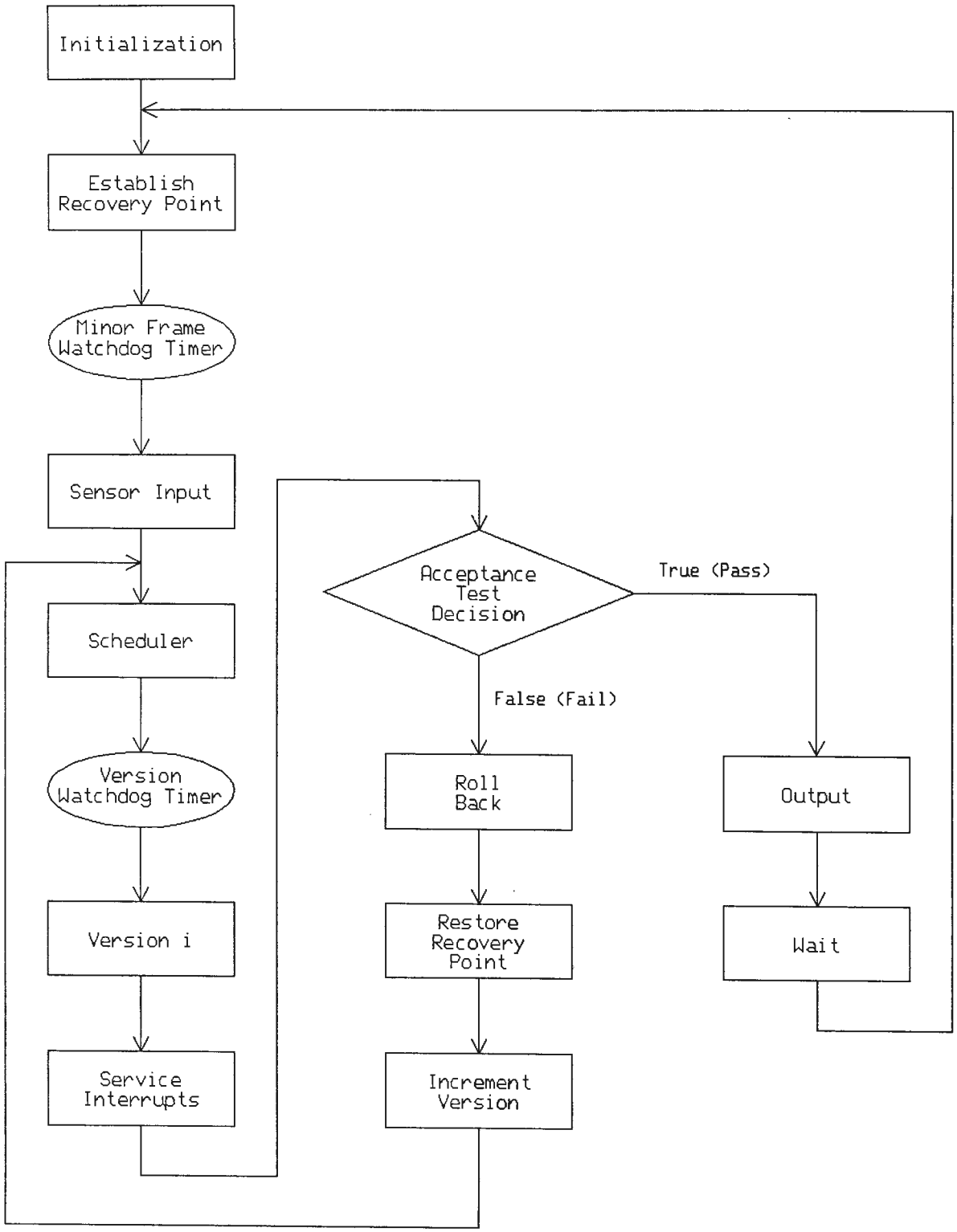



FIGURE 4.1-3. A REAL-TIME RECOVERY BLOCK WITH TWO WATCHDOG TIMERS

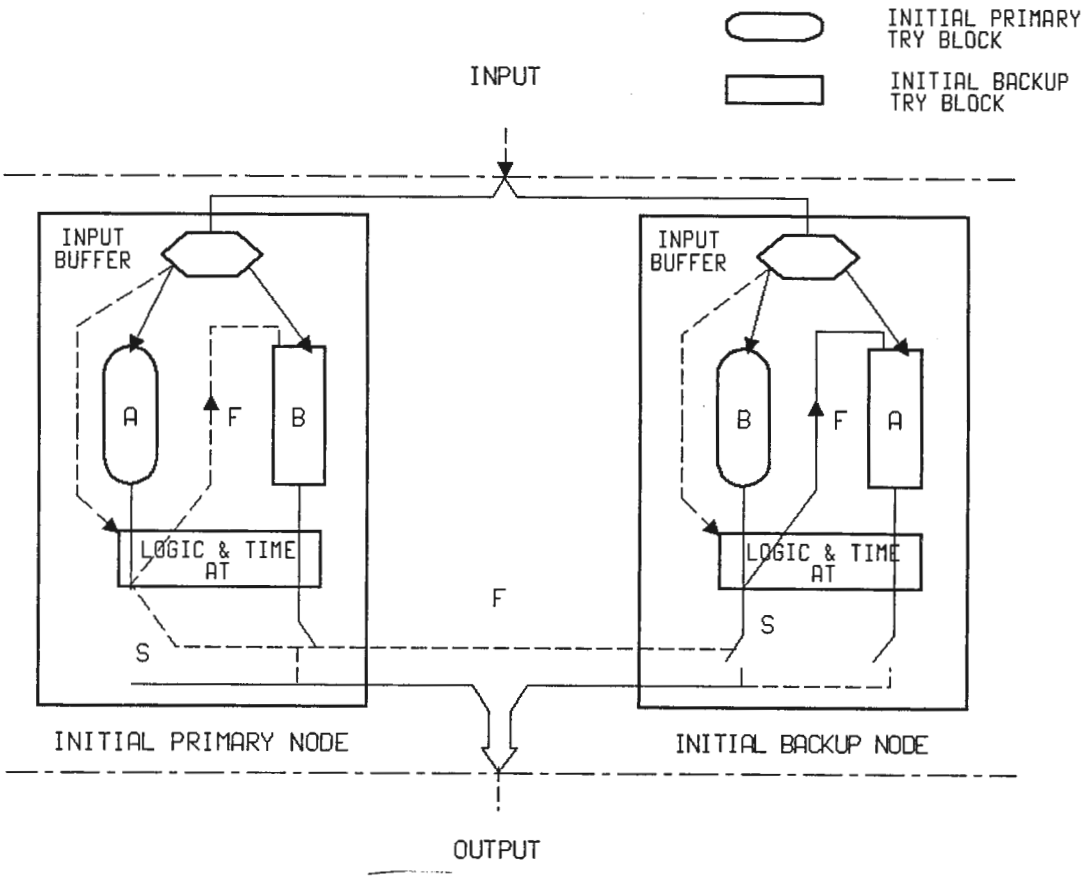



FIGURE 4.1-4. DISTRIBUTED RECOVERY BLOCK

reliability increases can be expected in a conventional software production setting; system-unique factors were cited as underestimating the true effects of software fault tolerance in this particular experiment.

## 4.2. Critical Issues

Critical issues for recovery blocks used in flight critical software include the correctness of the acceptance test, reliability of the underlying operating environment, and independence of the alternate routine.

### 4.2.1. Acceptance Tests

The acceptance test is a single point of failure in the recovery block. The two possible failures are: First, that the acceptance test accepts an incorrect result. Second, that the acceptance test rejects a correct result. The consequences of both failures can be serious. For example, failure to detect erroneous navigation data may result in an aircraft flying into the flight path

of another aircraft, an unauthorized airspace, or a military conflict area. The acceptance test's spurious rejection of correct results by both the primary and the alternate routine will result in a cessation of computation and a total failure of the recovery block.

The most effective means of minimizing this risk is fault avoidance in the acceptance test. Specifically, the acceptance tests should be very simple and amenable to thorough verification. The need to create simple, verifiable acceptance tests should be considered when designing the construction and modularization of this system. Further considerations in the use of fault avoidance versus fault tolerance are listed in section 5.

A second method of reducing the risk of defective acceptance tests is through nesting. Mitigation of the non-detection risk occurs because a faulty result may cause an acceptance test failure in subsequent calculations. This, in turn, can cause a rollback to an alternate routine which resumes the calculation using data unaffected by the fault. Mitigation of the spurious rejection risk occurs because the alternate path will use a different set of acceptance tests than was used on the primary path. However, there is a significant performance penalty in nesting. Extensive rollback can cause response time requirements to be missed. This performance penalty may not be acceptable in unstable aircraft because an unrecoverable situation can occur if the control system does not react in time.

#### 4.2.2. Run-time Environment

As is the case for all other software fault tolerance techniques, there is a dependence on the underlying operating system and run-time provisions. Concerns for recovery blocks are: maintenance of the Recovery Cache, invocation of the Alternate Routine, and maintenance of the Watchdog Timer.

Where the mechanisms for these functions are sufficiently simple, they can be subject to exhaustive verification and considered to be part of the protected, unalterable, "hard core" of the operating system. In other cases, these mechanisms can in and of themselves be made fault tolerant and rely on simpler mechanisms contained within an operating system hard core.

#### 4.2.3. Independence of the Alternate Routine

The type of degree of independence required in the alternate routine is determined largely by the nature of the fault classes which are anticipated. In the recovery block implementation for the navigation system cited above, independence was achieved by avoiding use of library procedures in the alternate routine. This measure ensures that precision or other subtle problems in the libraries will not affect the result. Because the alternate used the same equations as the primary, however, the developers needed to ensure that these equations would not be a source of error.

In other cases, it may be acceptable to use the same underlying library routines in both versions while using different equations or algorithms. This measure is appropriate if the developers are concerned about the appropriateness of the algorithms. A third measure is to use different languages and different

libraries for the primary and alternate. The availability of a reliable linking and calling mechanism alleviates the concerns of defects in libraries and compilers.

Definition of the degree of independence should be based on system-level reliability analyses performed as a part of the design process. During the validation of flight control systems, the analyses, the independence requirements derived from these analyses, and conformance with these requirements must be performed.

## 5. FAULT AVOIDANCE VERSUS FAULT TOLERANCE

Fault tolerant software provisions have costs in terms of development time, processing overhead, and recovery time. In addition, these provisions rely on underlying systems software or firmware in order to function properly. Thus, fault tolerance is not a substitute for fault avoidance, i.e., traditional software development practices designed to eliminate coding defects. These practices include:

- Software Development Process Controls including formal requirements generation, design methodologies, structured programming reviews, inspections, and configuration management.

- Software Testing at the unit, Computer Software Component (CSC), and Computer Software Configuration Item (CSCI) level.

- IV&V of deliverables, procedures, practices, and standards.

Ideally, stringent controls during the software development process can ensure that the entire project team works from a single design, uses well-defined interfaces, follows consistent variable naming and typing conventions, and develops adequate documentation. However, while such control can prevent faults due to a lack of communication or coordination among various team members, it cannot guarantee that subtle (or not so subtle) flaws in the specification or detail design will not result in faulty code. These design flaws are most likely to occur when timing, synchronization, exception handling, input/output, or interprocessor communication are involved.

Software testing can occur at the unit, component, and configuration item level. Strategies used in testing include branch testing, path testing, testing for singularities of critical variables, independent testing, and use of automated test tools. An important test methodology that has not received much attention in the literature is stress testing, which is deliberately exposing the program to faulty data sets or higher workloads than called for in the specification. Although still subject to the general limitations of testing, stress tests will frequently expose performance deficiencies which are not addressed by other strategies.

Code verification compares the detailed design with the actual code and devises tests and inspections that indicate compliance. Validation is done over a larger scope of system development, e.g., comparison of the delivered code with the A-specification or a lower level requirements document. For large real-time systems in critical applications, it is current practice to have a separate organization perform the IV&V. Usually, such IV&V activities are performed using standard procedures and sampling methods that are intended to provide a reasonable but by no means complete demonstration that the code fulfills its requirements.

Examples of where fault avoidance measures are either necessary or more efficient than real-time fault tolerance provisions include:

- Critical operating system software: Software that manages the invocation of fault-tolerant tasks; i.e, queues, buffers, interrupts, and timers must function flawlessly in order for the fault tolerance provisions to work effectively. Modules which perform these functions are generally large and may be amenable to the inclusion of fault tolerance provisions. However, there is still the need for total verification of these fault detection, isolation, and recovery provisions. Normal verification techniques described above are unsuitable for these critical code sections. More intensive techniques such as the Enhanced Condition Table (Tai, Hecht, and Hecht, 1987) together with extensive testing on the target system are generally needed.

- Constants: Faults in constants embedded in the code (e.g., size of memory or number of mailboxes for interprocessor communication) cannot easily be detected by fault tolerance provisions but can be readily found during pre-operational verification activities. System-level reliability analyses such as FMEAs and fault-tree analyses can be used to identify critical constants which must be verified. Utility programs or test tools can be built to perform this verification automatically or semi-automatically.

- Simple sections of code: While verification of entire real-time software systems is impractical, it may be possible to totally verify certain small sections of code--particularly if their input domains and operational environments are well-defined, and there are no timing or communication dependencies. Even when fault tolerance provisions can be defined, run time penalties and the need for additional exception handling frequently make them less desirable alternatives to fault avoidance.

- Static data: Many initiating events in software failures are related to input data. While much of the data in a flight control system is dynamic, there are significant portions which are static, e.g., geographical positions and aircraft parameters. As was the case for constants embedded in the code, verification and automated tools can be an effective and efficient means of detecting and correcting faults in such data.

On the other hand, fault avoidance provisions cannot be relied upon to provide complete coverage when the state of the system cannot be determined in advance. Examples include:

- Interprocessor communication: Requirements for interprocessor timing, sequencing, and message processing (whether on the same bus or from another network node) are generally not totally predictable during the design phase. Thus, it is generally impossible to develop a complete set of test cases that demonstrate correct functioning of the communication and message processing software. Under these circumstances, fault containment is a better approach. Data from interprocessor communication can be checked for integrity using error coding techniques, for reasonableness using range

checking or wraparound tests, and for security using various encoding schemes. Failure to pass these tests can cause a retry, substitution of default values, or use of previous values.

- Input/output: As was the case previously, the timing, content, and format of incoming messages cannot be predicted with sufficient certainty that an exhaustive set of test vectors can be produced. Under such conditions, online assertion checking such as that contained in fault containment or recovery blocks will constitute a more reliable approach.

- Critical response times: When it is difficult or impossible to ensure that all response time requirements will be met, a recovery block (which includes a watchdog timer as part of the acceptance test) is generally necessary.

# 6. INTEGRATION OF FAULT TOLERANCE TECHNIQUES

The previous sections have described critical issues and limitations associated with specific techniques. However, many of these limitations can be overcome by judicious combination of different fault tolerance techniques and the use of fault avoidance. Table 6-1 lists considerations that affect the choice of methods. The following subsections discuss the specific considerations in greater detail.

## 6.1. Coverage

Most robustness provisions address specific fault types, such as data out of range or improper arguments passed to a program. Fault containment and recovery blocks can use reasonableness or other types of acceptance tests which provide much broader scope and do not require prior knowledge of the expected faults. N-version programming provides the broadest coverage because its only limits are the common assumptions used in the specification or the design of all versions.

## 6.2. Retry Requirements

Retry of the same program always entails a possibility that the failure will be repeated in subsequent trials. Retry also impacts timing and throughput requirements. As long as no errors are encountered, the additional execution time is expected to be small; it is limited only to the assertion check at the end of the routine. However, when errors are encountered, the execution time may increase considerably--perhaps beyond the acceptable limits for time-critical applications. Because of the rarity of exception cases, average execution time is hardly affected by this consideration.

Failures covered by rollback can be resolved by assignment of default values or similar forms of forward recovery. Thus, retry is not always necessary using fault containment or robustness techniques.

By way of contrast, recovery blocks always use a new program if there is a failure. Similarly, use of N-version code does not require a retry because the final output is determined by a majority vote. Thus, for critical timing requirements or where fully fault tolerant software is necessary, the preferred techniques are N-version software or recovery blocks.

## 6.3. Cost and Complexity

Cost factors are related to the introduction of additional code including the development and maintenance expense of additional code and resources utilized during execution. Additional code introduces complexity which in turn increases the likelihood of added faults. Because robustness and rollback require

relatively little additional code, fault tolerant development costs represent only a minor fraction of the total cost for the delivered code. So long as retry is acceptable, these two techniques also impose the lowest overhead.

TABLE 6-1. SOME RELEVANT CHARACTERISTICS OF FAULT TOLERANCE TECHNIQUES

| Techniques | Coverage | Retry[1] | Cost and Complexity |
|---|---|---|---|
| Robustness | Narrow (specific to anticipated faults) | No (default or alternate is used) | Generally low |
| Rollback | Can be broad (determined by assertion) | Usually (default may be used) | Low to medium |
| Recovery Block | Broad (determined by acceptance test) | No (handled in alternate routine) | High |
| N-version software | Very broad (limited only by assumptions in the specifications common design assumptions | No (N-version run once) | High to very high |

[1]Whether a retry of the same code which induced the fault initially is necessary.

When the recovery block is used, at least three independent sections of code (the primary routine, the alternate routine, and the acceptance test) have to be designed, coded, tested, and maintained. A considerable increase in cost and complexity is therefore incurred. In most cases, the acceptance test is small relative to the code being tested and hence, run-time overhead is low. However, the alternate routine may be significantly less efficient than the primary routine when an error condition occurs, and repeated execution of the entire recovery block (i.e., primary, acceptance test, and alternate routine) may result in a major throughput penalty. The likelihood of such an occurrence should be assessed as part of the design process.

In N-version software cost can be even higher because (1) multiple versions must be developed separately and (2) special development of specifications may be necessary to ensure that programmers can work independently and produce

compatible results. During run-time, throughput is limited to the slowest of the routines. If the technique is practiced using true independence, it may not be possible to know the extent of this run time penalty until after completion of the coding when it may be too late. This run-time risk must be assessed as part of the software design because it may ultimately be the major cost driver in the development.

A major cost common to all fault tolerant techniques is verification during both development and maintenance. The verification of recovery provisions is generally difficult, and the major function of fault tolerance is to provide for recovery. For fault containment and recovery blocks, the goal of verification is to ensure that the acceptance test correctly identifies all error conditions and does not spuriously identify an acceptable result as a failure. For N-version programming, verification must be performed on all versions of the routine as well as on the voter for all possible results. During maintenance, thorough testing is necessary in order to ensure that coding changes do not defeat fault tolerance measures that are already in place.

Because of the high cost associated with fault tolerant software development, maintenance, and verification, its use should be restricted to functions critical to the continued safe operation of the system.

## 7. CONCLUSION

In order to achieve the very high reliability required in flight-critical systems, traditional software development methods are insufficient. Fault tolerance provisions can provide the additional reliability by detecting and recovering from faults occurring within the software before they affect aircraft operation and safety. However, in order to be effective, decisions on which fault tolerant techniques to use, where they are to be applied, and how they are to be verified must be made early in the design process and in the framework of the appropriate system-level analyses such as FMEA and Fault Tree Analysis.

The proper choice of fault avoidance versus fault tolerance, and judicious application of the appropriate fault tolerance techniques will result in a maintainable, cost effective, reliable, and high performance system. Those involved in the validation and certification of flight-critical digital systems should ensure that these analyses have been performed, and that they take into account the strengths and weaknesses of the techniques that have been described in this chapter.

BIBLIOGRAPHY


Anderson, T., et al., Software Fault Tolerance: An Evaluation, University of Newcastle upon Tyne, Technical Report No. 202, September, 1985.

Avizienis, A. and J. P. J. Kelly, "Fault Tolerance by Design Diversity: Concepts and Experiments," Computer, Vol. 17, No. 8, pp. 67-80, August 1984.

Chen, L. and A. Avizienis, "N-Version Programming: A Fault Tolerance Approach to Reliability of Software Operation," Digest of Papers, FTCS 8, pp 3-9, June 1978.

Data and Analysis Center for Software (DACS), "Quantitative Software Models," DACS SRR-1, March 1979.

Davis, E. A. and P. K. Giloth, "No. 4 ESS: Performance Objectives and Service Experience," Bell System Technical Journal, Vol. 60, No. 6, pp. 1203-1224, August, 1981.

Dijkstra, E. W., "Notes on Structured Programming," Structured Programming, Academic Press, New York, 1972.

Harvey, P. R., Fault Tree Analysis of Software, Thesis for the Master of Science in Information and Computer Science, University of California at Irvine, 1982.

Hecht, H. and M. Hecht, "Use of Fault Trees in the Design of Recovery Blocks," Proceedings 1982 Fault Tolerant Computing Symposium (FTCS-12), Santa Monica, CA, June, 1982.

Hecht, H. and M. Hecht, "Fault Tolerant Software," Fault Tolerant Computing, D. K. Pradhan, ed., Prentice Hall, Englewood Cliffs, NJ, 1986.

Hecht, M., "Extended Distributed Recovery Blocks," Proceedings of the Annual National Joint Conference on Software Quality and Reliability, National Security Industrial Association, Arlington, VA, March, 1988.

Hills, A. D., "Digital Fly-by-Wire Experience," Proceedings AGARD Lecture Series No. 143, October, 1985

Hitt, E. F. and S. A. Prater, Navigation Recovery Block Design Description, Federal Aviation Administration Technical Center, DOT/FAA/CT-87-15, Atlantic City, 1987.

Howden, W. E., "An Evaluation of the Effectiveness of Symbolic Testing," Software-Practice and Experience, Vol. 8, pp. 381-397, John Wiley & Sons, 1978.

Kelley, J. P. J., "A Specification Oriented Multi-Version Software Experiment," Proceedings of the 13th International Conference on Fault Tolerant Computing (FTCS-13), Milan, Italy, 1983.

Kim, K. H., "Distributed Execution of Recovery Blocks: An Approach to Uniform Treatment of Hardware and Software Faults," Proceedings of the 4th Annual International Conference on Distributed Computing Systems, pp. 526-532, IEEE Cat. No. 84CH2149-3, May, 1984.

Knight, J. and N. Leveson, "An Experimental Evaluation of the Assumption of Independence in Multiversion Programming," IEEE Transactions Software Engineering, Vol. SE-12, No. 1, pp. 96-109, January, 1986.

Martin, D. J., "Dissimilar Software in High Integrity Applications in Flight Control," Proceedings AGARD CPP-330, September, 1982.

Miller, E. F., Jr. et al., "Application of Structural Quality Standards to Software," Software Eng. Stand. Appl. Workshop, IEEE Cat. 81CH1663-7, pp. 51-57, July 1981.

Mulcare, D. B. and L. A. Barton, N-Version Software Demonstration for Digital Flight Controls, Federal Aviation Administration Technical Center, DOT/FAA/CT-86/33, April, 1987.

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

Tai, A., M. Hecht, and H. Hecht, "Enhanced Condition Table Method for Verification of Critical Software," Proceedings of the 11th Annual International Computer Software and Applications Conference (COMPSAC87), IEEE Computer Society Press, pp. 317-323, October, 1987.

Yau, S. S. and R. C. Cheung, "Design of Self-Checking Software," Proceedings 1975 International Conference on Reliable Software, IEEE Cat. 75CH0940-7CSR, pp. 450-475, April 1975.

Yount, L. J., "Architectural Solutions to Safety Problems of Digital Flight Critical Systems for Commercial Transports," Proceedings AIAA/IEEE Digital Avionics Systems Conference, Long Beach, CA, December 1984.

## GLOSSARY

A-SPECIFICATION. The highest level specification typically produced by the contracting organization to define a system (see MIL-STD-1521).

BACKWARD RECOVERY. Restoration of the system to some previous known correct state and restarting the computation from that point.

COVERAGE. The probability that when a fault occurs, it will be detected and recovery from the fault will be successful.

FAULT AVOIDANCE. The attempt to prevent any software faults in the final delivered product through disciplined software development practices, testing, and IV&V.

FAULT CONTAINMENT. The capacity of a system to prohibit errors and/or failures from propagating from the source throughout the system.

FAULT TOLERANCE. The capability to endure errors and/or failures without causing total system failure.

FAULT TOLERANCE. Software which continues to operate satisfactorily in the presence of faults.

FORWARD RECOVERY. Restoration of the system to a consistent state by compensating for inconsistencies found in the current state so that the system may continue processing.

GLASS COCKPIT. Advanced state-of-the-art electronic displays utilizing flat panel and/or cathode ray tube display technology for cockpit instrumentation.

RECOVERY CACHE. The location used to preserve input values until the outputs resulting from them have been accepted.

ROBUSTNESS. The ability of the code to perform despite some violation of the assumptions in its specifications usually via substitution of an alternate value and continuation of execution if a software fault is detected.

ROLLBACK. Retrying the calculation in the event that a failure is detected, under the assumption that some external condition may have changed thereby resolving the anomaly.

## ACRONYMS AND ABBREVIATIONS

| | |
|---|---|
| AIAA | American Institute for Aeronautics and Astronautics |
| CSC | Computer Software Component |
| CSCI | Computer Software Configuration Item |
| DRB | Distributed Recovery Block |
| ESS | Electronic Switching System |
| FMEAs | Failure Modes and Effects Analyses |
| IEEE | The Institute for Electrical and Electronics Engineers, Incorporated |
| IV&V | Independent Verification and Validation |
| MTBF | Mean Time Between Failures |

U.S. Department of Transportation
Federal Aviation Administration

DOT/FAA/CT-88/10

# HANDBOOK-VOLUME II
# DIGITAL SYSTEMS VALIDATION

## CHAPTER 10
## LATENT FAULTS

PREPARED BY:

COMPUTER RESOURCE MANAGEMENT, INC.
950 HERNDON PARKWAY, SUITE 360
HERNDON, VIRGINIA 22070

PREPARED FOR:

FEDERAL AVIATION ADMINISTRATION
TECHNICAL CENTER
ATLANTIC CITY INTERNATIONAL AIRPORT, NEW JERSEY 08405

NOTICE

This document is disseminated under the sponsorship of the U.S. Department of Transportation in the interest of information exchange. The United States Government assumes no liability for the contents or use thereof.

The United States Government does not endorse products or manufacturers. Trade or manufacturers' names appear herein solely because they are considered essential to the objective of this report.

COPY 1

# ERRATA

Report No. DOT/FAA/CT-88/10

DIGITAL SYSTEMS VALIDATION HANDBOOK - VOLUME II

February 1989

U.S. DEPARTMENT OF TRANSPORTATION
FEDERAL AVIATION ADMINISTRATION
TECHNICAL CENTER
ATLANTIC CITY INTERNATIONAL AIRPORT, NJ 08405

Replace the following pages in Chapter 10:

10-13, 10-14, 10-15, 10-16, 10-17, 10-18, 10-19, 10-20

Released October 1989

(Henceforth, we will assume that there is only a single auxiliary program. If more than one auxiliary program were involved, the complexity of the model would increase.)

The parameters of the model are not independent. In fact,

$$p_1 + p_2 + p_3 = 1$$

$$p_2 = (1 - p_1)\left[\frac{as}{as + ds}\right]$$

$$p_3 = (1 - p_1)\left[\frac{ds}{as + ds}\right]$$

From the results of the FIIS and previous experiments it was expected to obtain estimates of $p_1$, $p_3$, $e_\alpha$, and $e_\beta$. The parameters $p_2$, as, and ds would be obtained from typical FCS scenarios (e.g., duration of and time between initiation of auxiliary programs).

3.2. Parameter Estimates

- $p_1$: From the Bendix study, $.85 < p_1$; from FIIS, $.96 < p_1 < .98$
- $p_2$: It is estimated that $p_2 = 0$, approximately
  (With $p_2 = 0$, $1 - p_1$ = proportion of latent faults)
- $p_3$: From the Bendix study, $p_3 < .15$; from FIIS, $.02 < p_3 < .04$
- $e_\alpha$: From the FIIS study, $e_\alpha > 7200$/hour
- $e_\beta$: From the FIIS study, $e_\beta > 900$/hour
- as: From FCS scenarios, .01/hour < as < 10/hour
- 1/ds: From FCS scenarios, .01 hours < 1/ds < 1 hour
- f: The current range is .001/hour < f < .0001/hour

3.3. Potential Criticality of Latent Faults

In order to assess the effects of latent faults, we present two examples of conventional FCSs which could be vulnerable to the accumulation of latent faults. The examples noted are Triplex FCS (section 3.4) and Quadruplex FCS (section 3.5). The scenario is one in which latent faults, when activated, lead to a rapid and unexpected loss of components. Such a condition can arise in a Quadruplex FCS, where for example, in time, three out of the four lanes develop latent faults. Eventually, due to a single source of excitation, such as the callup of an outer-loop program, the faults become active. Even assuming, as we do, that the time interval between the successive activation of the faults (referred to, hereafter, as "avalanching latent faults") is sufficient to allow the comparators to detect and isolate each fault in turn, the result is a loss

of the three lanes in quick succession. It is our contention that the probability of such an event may not be insignificant.

3.4. Example 1 - Triplex FCS

The conventional reliability analysis, which assumes that there are no faults at the start of each flight, would conclude that the probability of loss of the FCS function in a flight of one hour is $3f^2$.

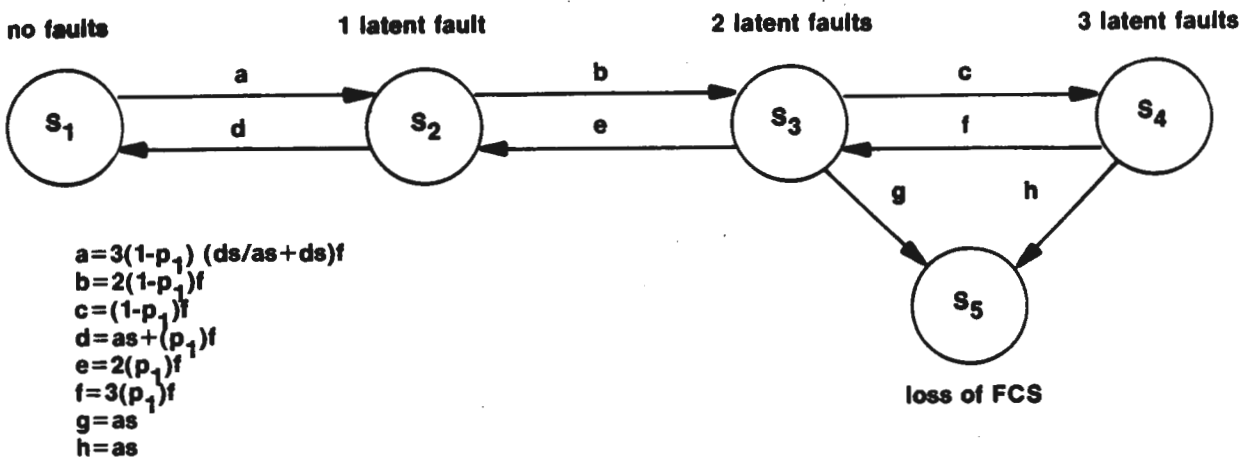



FIGURE 3.4-1. MARKOV MODEL/TRIPLEX FCS WITH LATENT FAULTS

A Markov reliability model of the FCS, representing the effects of avalanching latent faults, is shown in figure 3.4-1. The following assumptions are implicit:

- Loss of control is due, exclusively, to avalanching latent faults.
- Faults are exponentially distributed.
- The time between the successive activation of multiple latent faults is small relative to mission time.
- $\alpha$-faults (i.e., faults active at their occurrence) are detected immediately as they occur, at which time the entire lane is replaced. Thus, existing latent faults are effectively repaired.

- Excitation of latent faults is assumed to be correlated across lanes (i.e., a single excitation activates latent faults in different lanes).

Referring to figure 3.4-1:

- State $S_1$ represents the condition that all lanes are fault-free.
- State $S_2$ has a latent fault in one lane.
- State $S_3$ has a latent fault in two lanes.
- State $S_4$ has a latent fault in three lanes.
- State $S_5$ represents loss of the FCS function.

The transition rates are:

$$a = 3(1-p_1)f\left[\frac{ds}{as + ds}\right]$$

$$b = 2(1-p_1)f$$

$$c = (1-p_1)f$$

$$d = as + (p_1)f$$

$$e = 2(p_1)f$$

$$f = 3(p_1)f$$

$$g = as$$

$$h = as$$

It is noted that the backward transitions (d, e, and f) are the result of repairs due to the detection of $\alpha$-faults.

Let $x_1$, $x_2$, $x_3$, $x_4$, and $x_5$ denote the occupancy probabilities of states $S_1$, $S_2$, $S_3$, $S_4$, and $S_5$, respectively. From the figure we obtain the following differential equations:

$$sX_1 - x_1(0) = -aX_1 + dX_2$$

$$sX_2 = aX_1 - (b+d)X_2 + eX_3$$

$$sX_3 = bX_2 - (c+e+g)X_3 + fX_4$$

$$sX_4 = cX_3 - (f+h)X_4$$

$$sX_5 = gX_3 + hX_4$$

with initial conditions, $x_1(0) = 1$, $x_2(0) = x_3(0) = x_4(0) = x_5(0) = 0$. $X_1$, $X_2$, $X_3$, $X_4$, and $X_5$ denote the Laplace transforms of $x_1$, $x_2$, $x_3$, $x_4$, and $x_5$, respectively, and s denotes the Laplace operator.

Figure 3.4-2 shows the mean failure rate (MFR) (see appendix A), for a range of values of $1-p_1$ and f. The parameters $p_2$ and ds were fixed at the values $p_2 = 0$ and ds = 10/hour. Sensitivity studies indicate that the MFR is insensitive to values of ds between 1 and 10 per hour.

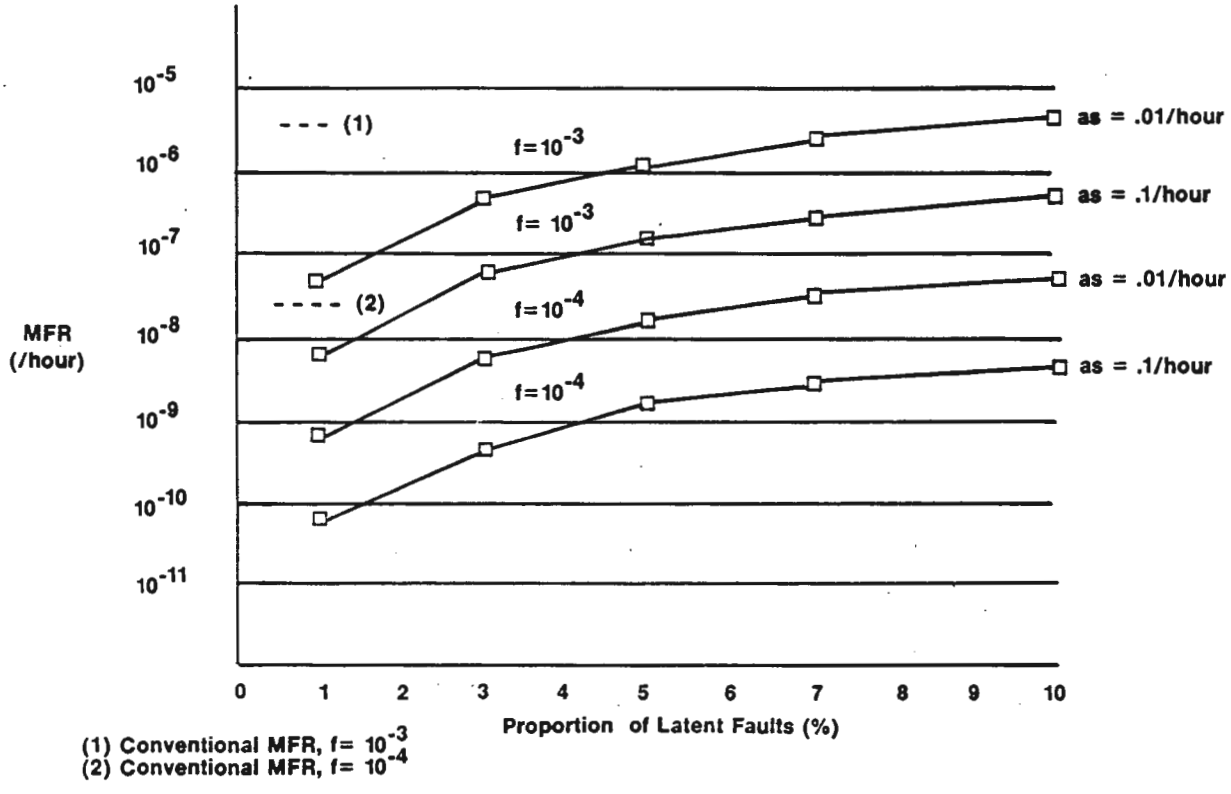



FIGURE 3.4-2. MEAN FAILURE RATE/TRIPLEX FCS

These results are summarized in tables 3.4-1 and 3.4-2 for lane failure rates ($10^{-3}$/hour and $10^{-4}$/hour, respectively). It is noted that the corresponding MFRs, based on conventional reliability analysis, are $3x10^{-6}$/hour and $3x10^{-8}$/hour, respectively.

TABLE 3.4-1. MFR, TRIPLEX FCS, $f = 10^{-3}$/HOUR

| as(/hour) | $1-p_1$(%) | MFR(/hour) |
|---|---|---|
| .1 | 10 | $5.76\times10^{-7}$ |
| .1 | 7 | $2.82\times10^{-7}$ |
| .1 | 5 | $1.44\times10^{-7}$ |
| .1 | 3 | $5.19\times10^{-8}$ |
| .1 | 1 | $5.77\times10^{-9}$ |
| .01 | 10 | $4.47\times10^{-6}$ |
| .01 | 7 | $2.20\times10^{-6}$ |
| .01 | 5 | $1.13\times10^{-6}$ |
| .01 | 3 | $4.07\times10^{-7}$ |
| .01 | 1 | $4.53\times10^{-8}$ |

TABLE 3.4-2. MFR, TRIPLEX FCS, $f = 10^{-4}$/HOUR

| as(/hour) | $1-p_1$(%) | MFR(/hour) |
|---|---|---|
| .1 | 10 | $5.92\times10^{-9}$ |
| .1 | 7 | $2.90\times10^{-9}$ |
| .1 | 5 | $1.48\times10^{-9}$ |
| .1 | 3 | $5.33\times10^{-10}$ |
| .1 | 1 | $5.92\times10^{-11}$ |
| .01 | 10 | $5.81\times10^{-8}$ |
| .01 | 7 | $2.85\times10^{-8}$ |
| .01 | 5 | $1.45\times10^{-8}$ |
| .01 | 3 | $5.23\times10^{-9}$ |
| .01 | 1 | $5.82\times10^{-10}$ |

3.4.1. Mean Failure Rate Approximation

An approximate value of MFR can be obtained as follows:

On the average, loss of control will occur if two latent faults occur in a time interval of length 1/as hours and no $\alpha$-faults occur in either of the lanes containing the latent faults. If q denotes this probability, then

$$q \approx 6[(1-p_1)f/as]^2[1 - 2(p_1)f/as]$$
$$\approx 6(1-p_1)^2f^2(1/as)^2 \qquad (1)$$

Thus, the Mean Time to Failure (MTTF) is:

$$MTTF \approx (1/as)[\ q + 2(1-q)q + 3(1-q)^2q + \ldots\ ] \quad (2)$$
$$= (1/as)/q$$

$$MFR \approx q(as) \quad (3)$$
$$\approx 6(1-p_1)^2f^2/as$$

3.4.2. Summary of Example 1

- The relative effect of latent faults is conveniently described by the ratio

$$R = \frac{\text{probability of loss of control with latent faults}}{\text{probability of loss of control by conventional analysis}}$$

  For values of $R \ll 1$, the effects of latent faults are insignificant. From the tables, the maximum value of R is .97 (for $f = 10^{-4}$, $as = .01$, $p_1 = .9$).

- Relative to the conventional MFR, a Triplex FCS does not appear to be especially vulnerable to latent faults; in the worst case, survivability is reduced by a factor of two.

- MFR is directly proportional to $(1-p_1)^2$.

- MFR is directly proportional to 1/as, the "time on risk."

- Survivability is adversely affected by the proportion of latent faults, $1 - p_1$, although for the range of parameters selected, this effect is not significant.

3.5. Example 2 - Quadruplex FCS

The conventional reliability analysis would indicate that probability of loss of the FCS function in a flight of one hour = $4f^3$.

A Markov reliability model of the FCS is shown in figure 3.5-1. The assumptions are the same as for the triplex system.

Referring to figure 3.5-1:

- State $S_1$ represents the condition that all lanes are fault-free.

- State $S_2$ has a latent fault in one lane.

- State $S_3$ has a latent fault in two lanes.

- State $S_4$ has a latent fault in three lanes.

- State $S_5$ has a latent fault in four lanes.

- State $S_6$ represents loss of the FCS function.

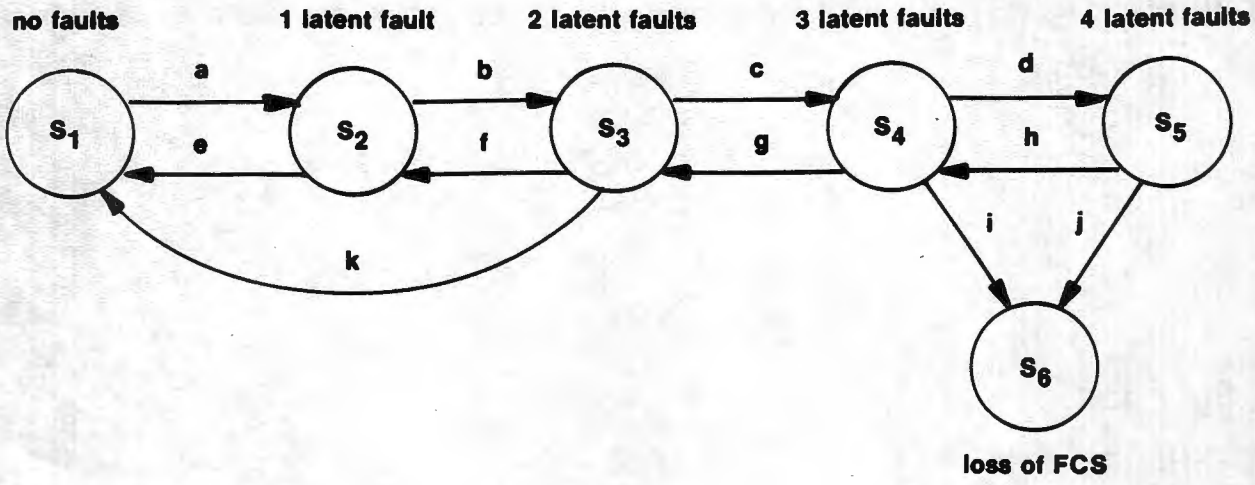



FIGURE 3.5-1. MARKOV MODEL/QUADRUPLEX FCS WITH LATENT FAULTS

The transition rates are:

$$a = 4(1-p_1)f\left[\frac{ds}{as + ds}\right]$$

$$b = 3(1-p_1)f$$

$$c = 2(1-p_1)f$$

$$d = (1-p_1)f$$

$$e = as + (p_1)f$$

$$f = 2(p_1)f$$

$$g = 3(p_1)f$$

$$h = 4(p_1)f$$

$$i = as$$

$$j = as$$

$$k = as$$

Again, it is noted that the backward transitions (e, f, g, h, and k) are the result of repairs due to the detection of $\alpha$-faults.

From the figure we obtain the following differential equations:

$$sX_1 - x_1(0) = -aX_1 + eX_2 + kX_3$$

$$sX_2 = aX_1 - (b + e)X_2 + fX_3$$

$$sX_3 = bX_2 - (c + f + k)X_3 + gX_4$$

$$sX_4 = cX_3 - (d + g + i)X_4 + hX_5$$

$$sX_5 = dX_4 - (h + j)X_5$$

$$sX_6 = iX_4 + jX_5$$

with initial conditions, $x_1(0) = 1$, $x_2(0) = x_3(0) = x_4(0) = x_5(0) = x_6(0) = 0$. $X_1$, $X_2$, $X_3$, $X_4$, $X_5$, and $X_6$ denote the Laplace transforms of $x_1$, $x_2$, $x_3$, $x_4$, $x_5$, and $x_6$, respectively.

Figure 3.5-2 shows the MFR for a range of $1-p_1$ and f with $p_2$ and ds again fixed at $p_2 = 0$, ds = 10/hour. These results are summarized in tables 3.5-1 and 3.5-2 for lane failure rates, $10^{-3}$/hour and $10^{-4}$/hour, respectively.

It is noted that the corresponding MFRs, based on conventional reliability analysis, are $4 \times 10^{-9}$/hour and $4 \times 10^{-12}$/hour, respectively.

TABLE 3.5-1. MFR, QUADRUPLEX FCS, F = $10^{-3}$/HOUR

| as(/hour) | $1-p_1$(%) | MFR(/hour) |
|---|---|---|
| .1 | 10 | $2.23 \times 10^{-9}$ |
| .1 | 5 | $2.80 \times 10^{-10}$ |
| .1 | 3 | $6.04 \times 10^{-11}$ |
| .1 | 1 | $2.24 \times 10^{-12}$ |
| .01 | 10 | $1.37 \times 10^{-7}$ |
| .01 | 5 | $1.73 \times 10^{-8}$ |
| .01 | 3 | $3.75 \times 10^{-9}$ |
| .01 | 1 | $1.39 \times 10^{-10}$ |

TABLE OF CONTENTS


| Section | | Page |
|---|---|---|
| 1. | INTRODUCTION AND HISTORY | 10-1 |
| | 1.1. Bendix Study | 10-1 |
| | 1.2. Charles Stark Draper Laboratories Study | 10-2 |
| | 1.3. Results | 10-3 |
| | 1.4. Conclusions | 10-4 |
| 2. | OVERVIEW OF THE FIIS EXPERIMENTS | 10-5 |
| | 2.1. Objectives | 10-5 |
| | 2.2. Test Ground Rules and Procedures | 10-5 |
| | 2.2.1. Open-loop Scenario | 10-5 |
| | 2.2.2. Closed-loop Scenario | 10-6 |
| | 2.2.3. Self-Test | 10-6 |
| | 2.2.4. Test Conditions | 10-6 |
| | 2.3. Results | 10-7 |
| | 2.4. Summary of the FIIS Experiments | 10-8 |
| | 2.5. Conclusions of the FIIS Experiments | 10-9 |
| 3. | FUNCTION CRITICALITY | 10-11 |
| | 3.1. Single-Fault Model | 10-11 |
| | 3.2. Parameter Estimates | 10-13 |
| | 3.3. Potential Criticality of Latent Faults | 10-13 |
| | 3.4. Example 1 - Triplex FCS | 10-14 |
| | 3.4.1. Mean Failure Rate Approximation | 10-17 |
| | 3.4.2. Summary of Example 1 | 10-18 |
| | 3.5. Example 2 - Quadruplex FCS | 10-18 |
| | 3.5.1. Mean Failure Rate Approximation | 10-22 |
| | 3.5.2. Summary of Example 2 | 10-22 |
| | 3.6. Self-Test | 10-22 |
| 4. | UNRESOLVED ISSUES | 10-23 |
| 5. | SUPPLEMENTAL WORKED EXAMPLE | 10-25 |
| | 5.1. FCS Configuration | 10-25 |
| | 5.2. Unsafe Condition | 10-26 |
| | 5.3. Repair of Latent Faults | 10-26 |

TABLE OF CONTENTS (Continued)


| Section | | Page |
|---|---|---|
| 5.4. | Markov Reliability Model | 10-26 |
| | 5.4.1. Model Parameters | 10-28 |
| 5.5. | Effects of Latent Faults on Survivability | 10-28 |
| 5.6. | The Conventional Reliability Analysis | 10-29 |
| 5.7. | Numerical Example | 10-30 |
| 5.8. | Summary of Example | 10-30 |
| 6. | CONCLUSIONS | 10-31 |
| APPENDIX | | |
| | A - MEAN FAILURE RATE | 10-33 |
| BIBLIOGRAPHY | | 10-35 |
| GLOSSARY | | 10-36 |
| ACRONYMS AND ABBREVIATIONS | | 10-37 |

## LIST OF ILLUSTRATIONS


| Figure | | Page |
|---|---|---|
| 3.1-1 | SINGLE-FAULT MODEL | 10-12 |
| 3.4-1 | MARKOV MODEL/TRIPLEX FCS WITH LATENT FAULTS | 10-14 |
| 3.4-2 | MEAN FAILURE RATE/TRIPLEX FCS | 10-16 |
| 3.5-1 | MARKOV MODEL/QUADRUPLEX FCS WITH LATENT FAULTS | 10-19 |
| 3.5-2 | MEAN FAILURE RATE, QUADRUPLEX FCS | 10-21 |
| 5.1-1 | DUAL, FAIL SAFE FLIGHT CONTROL SYSTEM | 10-26 |
| 5.4-1 | MARKOV MODEL/DUAL, FAIL SAFE, FLIGHT CONTROL SYSTEM | 10-27 |


## LIST OF TABLES


| Table | | Page |
|---|---|---|
| 2.3-1 | OPEN-LOOP, UNWEIGHTED FAULTS | 10-7 |
| 2.3-2 | CLOSED-LOOP, UNWEIGHTED FAULTS | 10-8 |
| 2.3-3 | SUMMARY OF BASELINE/AUXILIARY FAULTS | 10-8 |
| 3.4-1 | MFR, TRIPLEX FCS, $f = 10^{-3}$/HOUR | 10-17 |
| 3.4-2 | MFR, TRIPLEX FCS, $f = 10^{-4}$/HOUR | 10-17 |
| 3.5-1 | MFR, QUADRUPLEX FCS, $F = 10^{-3}$/HOUR | 10-20 |
| 3.5-2 | MFR, QUADRUPLEX FCS, $F = 10^{-4}$/HOUR | 10-21 |

# 1. INTRODUCTION AND HISTORY

The reliability of a flight control system (FCS) is achieved by a combination of (1) reliable components, (2) redundancy, and (3) the ability of the FCS to detect, isolate, and recover from faults. All highly reliable FCS designs incorporate some form of comparison-monitoring to detect faults. When the fault, or rather, the error produced by the fault, reaches a comparator it is immediately detected, at which time the isolation and recovery mechanisms are invoked.

Experience has shown that comparison-monitoring is an effective and reliable means of detecting faults. Moreover, because a comparator only detects a fault when the fault produces an error, it retains the faulty component for as long as the component functions correctly. In the interim, the fault may remain latent for long periods of time before it finally produces a malfunction. To detect these latent faults, some form of self-test is employed, either in a preflight or maintenance test or, during flight, in background.

With respect to latent faults, the critical issues are:

- What proportion of faults are latent?
- What mechanism activates a latent fault?
- What are the effects, if any, of latent faults on aircraft survivability?
- If latent faults are a contributing factor to reduced survivability, what level of coverage is required to detect and eliminate them?

Prior to the Fault Insertion and Instrumentation System (FIIS) experiments (the results and implications of which comprise the subject matter of this tutorial), most of the published information on data latency in digital FCSs was obtained from two important studies conducted independently by Bendix (McGough and Swern, 1983) and by the Charles Stark Draper Laboratories (CSDL) (Lala and Smith, 1983).

## 1.1. Bendix Study

The purpose of this study was to estimate detection coverage of several candidate self-test programs and to estimate fault latency in a conventional digital FCS. Fault injection experiments were conducted on a software-simulated version of the Bendix BDX-930 bit-sliced flight control computer (FCC), which featured the 2901A arithmetic logic unit (ALU) as the principal processing element. The simulated components consisted of the central processing unit (CPU), micromemory, program memory (ROM), and random access memory (RAM). The input/output (I/O) circuitry was not simulated. Sensor inputs were randomly generated and deposited in RAM at the start of each computational frame. Each

device of the CPU was represented by a gate-equivalent circuit and stuck-at faults were injected at the gate nodes, something not possible in actual hardware fault testing. The micromemory was represented functionally and faults were simulated by reversing the logic states of single bits.

The ROM and RAM memories were also represented functionally but no faults were injected into these devices. Faults were randomly selected and injected, one at a time. Fault selection was weighted in proportion to the failure rate of the device (i.e., the average number of faults injected into a particular device was proportional to the device failure rate). The simulation technique was "parallel mode," which allowed for the simultaneous simulation of up to 32 computers, one of which was always the non-failed version. This made it possible to compare the responses of the failed and non-failed computers at any time in the compute cycle and at any device.

Although all devices were simulated at the gate-level, faults were injected at both the pin-level and gate-level in order to determine differences in detection coverage between the two fault types.

Two types of experiments were conducted, depending upon the method of detection:

- To determine the effectiveness of comparison-monitoring, the computed outputs of a failed and non-failed computer were compared at the end of each frame. Any discrepancy was defined as a "detected failure". These comparisons were performed by the simulator executive and did not involve the detection mechanisms that would normally be resident in the FCCs. Each computer executed the same flight control program, which consisted of the inner loops of a high performance aircraft. In order to reduce the simulation time, a fault run was terminated after detection or after eight repetitions, whichever occurred first. To avoid any ambiguity in the comparison process, each computer received identical sensor inputs at the start of each frame. The simulation was conducted "open-loop", with sensor values selected independently and at random.

- To determine the effectiveness of self-test, each fault was injected and a candidate self-test program was executed. A fault was defined as "detected" if the mechanism of the self-test routine so indicated.

### 1.2. Charles Stark Draper Laboratories Study

The purpose of this study was to assess the fault detection, identification, and reconfiguration capabilities of the CSDL-designed fault-tolerant multiprocessor (FTMP). This system featured the Rockwell/Collins CAPS-6 computer as the principal processing element. The CAPS-6 is similar to the BDX-930 in that it, too, is a bit-sliced processor utilizing the 2901A ALU. FTMP is a triple modular redundant (TMR), bit-synchronized multiprocessor, designed for ultra high reliability ($10^{-10}$ failures/hour) and fault-tolerance. The design is based on independent processor-cache memory modules and common memory modules, which communicate via redundant serial buses. All information processing and transmission is performed in triplicate. Data transmitted over the bus network is monitored by Bus Guardian Units (BGUs), which compare the transmitted data, bit by bit. Faults are detected when they produce errors at the BGUs. Faults are

also detected by self-test programs, which are executed continuously in background.

The fault injection experiments were conducted on real hardware. Stuck-at faults were injected, one at a time, on device pins of one of the three processing elements. (The fault injection hardware was identical to that used in the FIIS experiments.) Faults were systematically selected (unweighted) and injected on pins of eight circuit boards. FTMP executed a flight control software program consisting of inner-loops and autopilot modes, which were patterned after the L-1011 Tristar FCS. The fault runs were terminated after 15 seconds of real time. Unlike the Bendix experiments, the simulation was closed-loop (i.e., the sensor values were obtained from the motion parameters of a simulated aircraft).

### 1.3. Results

Both studies indicated that most faults are were detected within a few computational frames of their occurrence (e.g., 0 to 500 ms). Because fault runs were terminated after eight repetitions, the Bendix study gave no information about detection beyond 800 ms (assuming 100 ms per frame). The CSDL results indicated that relatively few faults (2 to 4 percent) were detected in later repetitions and these were detected by background self-test programs. From these results it was conjectured that most faults are activated by the baseline software program and are independent of the input sequence.

CSDL injected 21,055 faults (each pin fault was injected five times, in different locations of the program) and 3,637 (17.3 percent) were undetected. CSDL estimated that about 3000 of the undetected faults were "don't care" faults (i.e., faults on unused pins or on signals that were always low or always high under normal circumstances). Of the remainder, a few were analyzed and found to be "don't cares". In the Bendix study, 3000 faults were injected during the self-test evaluation phase and each undetected fault was analyzed. It was found that 466 faults (15.5 percent) were "don't cares" and, of these, 245 occurred in the bits of the micromemory and 47 in the control Programmable Read-Only Memory (PROM). When memory bit-faults were excluded, 7.2 percent of gate-level faults were "don't cares". Both Bendix and CSDL concluded that the identification of "don't care" faults was a non-trivial task, but essential to obtaining reasonably accurate estimates of fault coverage.

The Bendix study found that 86.5 percent of gate-level faults (excluding bit-faults in the micromemory and control PROMS) were detected during the eight repetitions of the FCS program, based on 148 injected "care" faults. Pin-level faults were not injected while executing the FCS program. The equivalent coverage in the CSDL study is not clear because of the large number of undetected, unanalyzed faults. Most of these faults (but, possibly, not all) were "don't cares." CSDL did conclude, however, that between 2 percent and 4 percent of all detected faults were detected by the background self-test programs and not by the comparators. Since self-test was not executed during the FCS experiments, Bendix would have concluded that these faults were undetected.

- The Bendix study showed that 88 percent of 2901A gate-level faults were detected during the FCS experiments, based on 52 injected "care" faults. The CSDL study gave no results for this device.

- In the Bendix study 97.4 percent of gate-level faults and 97.6 percent of pin-level faults were detected by self-test, based on 2234 and 376 injected "care" faults, respectively. The gate-level coverage excluded bit-faults in the memory elements. When these faults were included, coverage was 92 percent. The self-test program consisted of 346 Assembly Language instructions and required two to three ms to complete. The CSDL study gave no results for self-test coverage.

- The Bendix study concluded that gate-level faults were more difficult to detect than pin-level faults, especially when faults were injected into single bits of the memories. When executing software, other than the FCS, the ratio of gate-level to pin-level undetected faults was a factor of two. Unfortunately, "don't care" faults were not identified in these runs. As a result, detection coverage estimates tended to be pessimistic.

### 1.4. Conclusions

Prior to the FIIS experiments it was concluded, based upon the Bendix and CSDL studies, that:

- Most detected faults are detected within a few computational frames of their occurrence.

- The proportion of faults not detected by comparison-monitoring while executing the baseline program can range from 2 percent to 13.5 percent.

- Detection, isolation, and recovery could take up to several seconds, during which time the system (i.e., FTMP) is potentially vulnerable to second faults. The occurrence of a second fault, before the first fault is isolated, could confuse the majority vote and result in a system breakdown.

- "Don't care" faults constitute a statistically significant proportion of injected faults (between 6 percent and 17.3 percent). These faults are difficult to identify. Unidentified "don't care" faults could result in uncertain and pessimistic detection coverage estimates.

- Self-test coverage of 95 percent is easily achieved for pin-level faults and for gate-level faults if memory bit-faults are excluded.

- There is a significant difference in detection coverage between pin-level and gate-level faults, particularly if memory bit-faults are included in the latter.

## 2. OVERVIEW OF THE FIIS EXPERIMENTS

### 2.1. Objectives

The Bendix and CSDL studies were intended to provide an initial database for future FCS survivability assessment. The objectives of the FIIS experiments were (Benson, Mulcare, and Larsen, 1987):

- Corroborate and augment the results of the Bendix and CSDL studies.
- Provide a database of detection coverage and fault latency which future experimenters could use as goals or as a basis of comparison.
- Evaluate the FIIS fault injection methodology. (It was hoped that this methodology would be the first step in establishing guidelines for future fault injection experiments and fault detection coverage estimation.)
- Provide a database for the construction of single and multiple-fault models which could be used by reliability programs to assess FCS survivability.
- Identify unresolved issues associated with FCS survivability assessment and recommend studies to resolve them.

### 2.2. Test Ground Rules and Procedures

The target hardware was a dual FCC, which implemented a complete FCS for a commercial aircraft (Lockheed L-1011 Tristar). The FCS was dual/dual, but only a single pair was used in the experiments. Faults were detected by comparison monitoring of computed variables which were periodically exchanged between FCCs. Either FCC could request and effect system disengagement if it observed a discrepancy in any of these monitored variables. System disengagement constituted "detection." In addition, the system featured hardware comparators which were located in the secondary actuators. These comparators, which were not faulted, effectively measured the difference between the surface commands generated by the two FCCs. Again, comparator exceedances constituted "detection" and resulted in system disengagement. The experiments were conducted in open-loop and closed-loop scenarios.

#### 2.2.1. Open-loop Scenario

The simulated airframe was disconnected for all runs. As a consequence, sensor inputs were invariant. In this configuration the operational program consisted of the inner loops, mode logic servicing, executive functions, synchronization, control panel and display servicing, voting and monitoring, and intercomputer communications. The open-loop program executed approximately 11,000 Assembly instructions every 50 ms.

### 2.2.2. Closed-loop Scenario

The simulated airframe was connected, including the sensor feedback signals.

In addition to the open-loop programs, the operational program consisted of the autopilot modes: cruise altitude hold, cruise climb, cruise turn, localizer capture, and glideslope capture/track. The land modes consisted of approximately 1200 additional Assembly Language instructions. In the autopilot modes the aircraft was perturbed from equilibrium flight by initial conditions and control wheel steering commands. The localizer mode was executed every 100 ms, the glideslope modes every 200 ms, and the other modes every 50 ms.

The open-loop experiments were conducted first. All faults were first injected in the open-loop scenario and only undetected faults were injected in the closed-loop scenarios. This saved time since most faults were detected in the open-loop experiments. Each undetected fault was subsequently injected during an autopilot mode, the purpose being to determine the proportion of faults detected by programs other than the baseline program. (It was assumed that the baseline program consisted of the inner-loop programs and cruise altitude hold.) The same fault was successively injected while executing each of the autopilot modes.

### 2.2.3. Self-Test

Each FCC contained a self-test program which was normally executed in the background. During the open and closed-loop experiments the self-test program was disabled. In order to determine self-test coverage and to identify "care faults", all faults which were undetected by both the open and closed-loop programs, and a larger sample of detected faults were injected separately, executing only the self-test program.

### 2.2.4. Test Conditions

Faults consisted of permanent S-a-0, S-a-1 and pin inversions, injected on input and output pins of almost every device on the data path and control cards of the CAPS-6 computer. Devices were only excluded when they would not function with the FIIS multiplexers.

- No faults were injected in the analog interface hardware.
- Faults were injected in a single FCC of the dual pair.
- Only single faults were injected.
- Faults were detected by comparison monitoring, exclusively. Self-test was not executed in the open and closed-loop scenarios.
- No faults were injected in the hardware associated with failure detection (i.e., hardware comparators and disengage mechanisms).
- Detection was recorded at 50 ms intervals. (This was the same as the minor frame interval.)

- An undetected fault run was terminated after 15 seconds of real time (i.e., 300 open-loop iterations).
- Each fault was individually identified.
- Coverage was tabulated for unweighted and weighted faults. In the latter, faults were weighted in proportion to the failure rate of the device; in the former, all faults were weighted equally.
- All undetected faults were analyzed and "don't care" faults were eliminated from the tabulations. "Don't care" faults consisted of unused pins or signals that were always high or low under all operating conditions.

## 2.3. Results

Detailed test results can be found in Benson, Mulcare, and Larsen (1987). In this section only the results for unweighted S-a-0 and S-a-1 faults are given.

Table 2.3-1: Out of 1670 injected "care" faults, 1597 (95.63 percent) were detected and 73 (4.37 percent) were undetected. Self-test detected 1115 faults out of 1115 injected.

TABLE 2.3-1. OPEN-LOOP, UNWEIGHTED FAULTS

| Device | # Faults Injected | # Faults Detected | # Faults Undetected | Don't Cares | Percent Detected | Percent Undetected |
|---|---|---|---|---|---|---|
| Data Path | 1016 | 990 | 26 | 60 | 97.44 | 2.56 |
| Control | 654 | 607 | 47 | 63 | 92.81 | 7.19 |
| S-a-0/1 | 1670 | 1597 | 73 | 123 | 95.63 | 4.37 |
| 2901A | 283 | 283 | 0 | 13 | 100. | 0.0 |
| Micromemory | 146 | 146 | 0 | 9 | 100. | 0.0 |
| Input Pins | 1184 | 1133 | 51 | 69 | 95.69 | 4.31 |
| Output Pins | 486 | 464 | 22 | 54 | 95.47 | 4.53 |
| Self-Test | 1115 | 1115 | 0 | 0 | 100. | 0.0 |

[1] Excludes self-test
[2] Detected by self-test
[3] "Don't cares" = 6.86 percent

Table 2.3-2: Each of the 73 undetected open-loop faults was successively injected while executing each of the six autopilot modes. Thus, the same fault could have been detected while executing several modes, as indeed the results show. Of the 73 injected "care" faults, 33 (45.21 percent) were detected and 40 (54.71 percent) were undetected.

TABLE 2.3-2. CLOSED-LOOP, UNWEIGHTED FAULTS

| Mode | # Faults Injected | # Faults Detected | # Faults Undetected | Don't Cares | Percent Detected | Percent Undetected |
|---|---|---|---|---|---|---|
| Alt.Hold | 73 | 11 | 62 | 0 | 15.07 | 84.93 |
| Climb | 73 | 10 | 63 | 0 | 13.7 | 86.3 |
| Turn | 73 | 10 | 63 | 0 | 13.7 | 86.3 |
| Loc/Capt | 73 | 13 | 60 | 0 | 17.81 | 82.12 |
| GS/Capt | 73 | 9 | 64 | 0 | 12.33 | 87.67 |
| GS/Track | 73 | 12 | 61 | 0 | 16.44 | 83.56 |
| Self-Test | 73 | 73 | 0 | 0 | 100. | 0.0 |
| Totals | 73 | 33(1) | 40(1) | 0 | 45.21 | 54.79 |

1 Detected by self-test

Table 2.3-3: These results indicate that most faults are detected while executing the baseline program. However, there is a significant number of faults not detected by either the baseline or auxiliary programs (e.g., 2.4 percent).

TABLE 2.3-3. SUMMARY OF BASELINE/AUXILIARY FAULTS

| | |
|---|---|
| Total faults injected | 1670 |
| Detected by baseline | 1597 (95.63%) |
| Undetected by baseline/detected by auxiliary | 33 (1.98%) |
| Undetected by baseline/undetected by auxiliary | 40 (2.4%) |

## 2.4. Summary of the FIIS Experiments

- Most detected faults were detected within a few computational frames of their occurrence (e.g., within 10 frames = 500 ms). This result corroborates the results of the Bendix and CSDL studies.

- Faults were activated, i.e., produced errors, primarily by the operating software and not by time-varying sensor inputs, as evidenced by the high coverage in the open-loop scenario.

- Most faults were activated (and detected) by the baseline software program (95.63 percent). This coverage is considerably better than the Bendix study indicated, which is not surprising considering that the Bendix FCS program was 2200 words whereas the open-loop program was 11,000 words.

- A small number of open-loop detected faults were detected between 1 and 6.5 seconds (0.3 percent).

- A small number of faults were not activated by the baseline program but were activated by an auxiliary program (1.98 percent).

- A small number of faults were not activated by either the baseline or auxiliary programs (2.4 percent).

- Detection statistics for weighted and unweighted faults were similar (e.g., 96.5 percent for weighted versus 95.6 percent for unweighted).

- Invert faults are less latent than stuck-ats. For example, 1.71 percent are undetected when invert faults are included versus 4.37 percent when excluded. (One-third of all faults were inverts.)

- Micromemory and 2901A faults were 100 percent detected. This is not surprising since pin-level faults on these devices tend to produce massive errors.

- Self-test coverage was 100 percent. This was impressive, even at the pin-level. The self-test designer was obviously well-acquainted with the hardware.

- The percentage of all faults classified as "don't cares" is 6.11. This is consistent with Bendix's results, which estimated 7 percent at the pin-level. The relatively large number of these faults, together with the difficulty in identifying them, precludes accurate estimates of detection coverage.

## 2.5. Conclusions of the FIIS Experiments

- FIIS test results generally corroborate the results of the Bendix and CSDL studies. This was surprising considering the dissimilarity of input sensor selection, computer architecture, and operational software between the three studies.

- The validity of the FIIS approach hinges on the validity of pin-level fault models. Until this issue is resolved the results of the FIIS and previous experiments must be considered tentative. However, the results can be used as a relative measure of fault detection capability.

- The FIIS methodology was thoroughly tested during the study. FIIS performance was impressive, especially in the following areas:

  - Fault Generation Capability: Although only stuck-at and invert faults were injected, FIIS provides the capability of simulating a wide variety of fault types.

  - Ease of Use: Injecting faults and recording data was tedious but relatively simple after the initial setup.

- Fidelity: Since faults are injected into the target hardware there is no question about the fidelity of the experiments nor is there any need to validate the simulation.

- Speed: All of the FIIS experiments were run in real time. As a consequence, it was possible and practicable to determine detection coverage over long periods of time.

# 3. FUNCTION CRITICALITY

## 3.1. Single-Fault Model

The FIIS, Bendix, and CSDL studies established that, although most faults are detected within a few computational frames of their occurrence, there is a small number, perhaps 5 percent, which are not detected for much longer periods of time. These latent faults occur in devices which are infrequently exercised by the software program. As a consequence, they could accumulate and, if eventually activated by a common source of excitation, could result in a near-simultaneous loss of redundancy.

In planning the FIIS experiments, it was agreed by all of the participants that the principal and fundamental objective was to obtain a database for the reliability assessment of FCSs. As a consequence, it was necessary to anticipate the kind of data that might be required. Recognizing that the key element of a reliability assessment program was a single-fault model, it was decided to generate a strawman model in order to identify parameters which could be obtained from the FIIS experiments.

The results of the Bendix and CSDL studies seemed to indicate that most faults are activated by the baseline software (i.e., the executive and inner-loops). Only a small proportion (not activated by the baseline) are activated by auxiliary programs. These observations suggested the following definitions:

- Baseline Program: A set of continuously executed software modules.
- Auxiliary Programs: Software executed occasionally.
- Active Fault: A fault that can produce an error (for some input) while executing the current program.
- $\alpha$-Fault: A fault activated by the baseline program.
- $\beta$-Fault: A fault not activated by the baseline program.
- Benign Fault: A fault that cannot produce an error while executing the current program, regardless of input, but may produce an error for some other program.
- Latent Fault: A fault which has not yet produced a malfunction. (In the context of the single-fault model, benign and latent faults are equivalent.)

Typically the baseline program is the operating system kernel while an application program, executed on demand, would be considered an auxiliary program.

The structure of the strawman single-fault model is shown in figure 3.1-1.

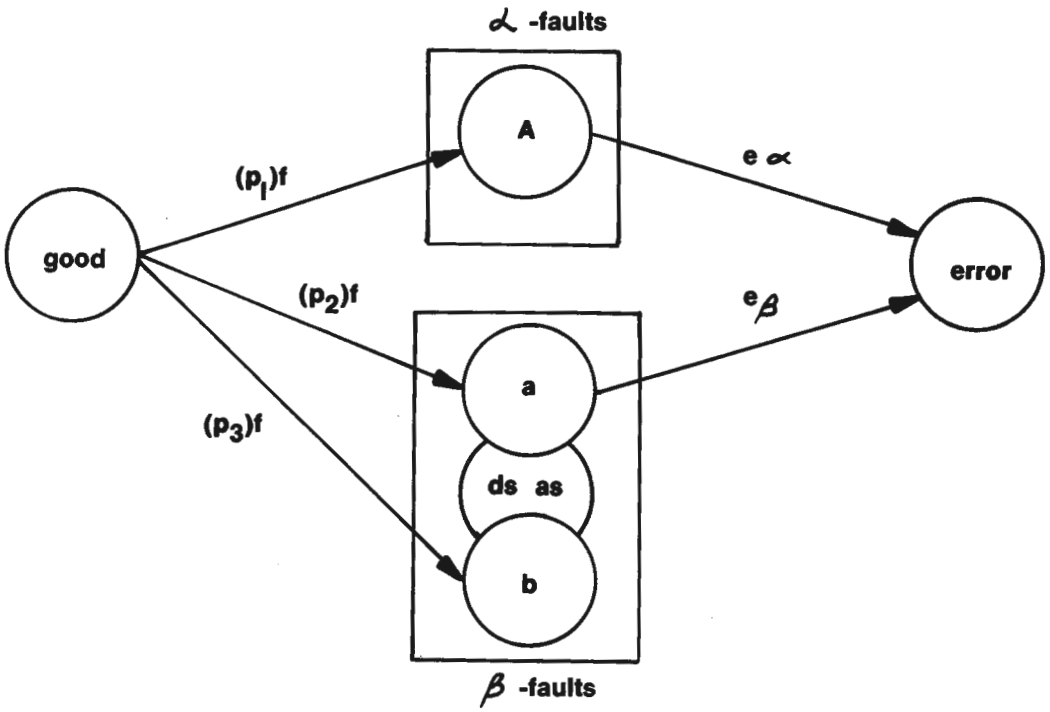



FIGURE 3.1-1. SINGLE-FAULT MODEL

The model is Markovian and appears to have sufficient degrees-of-freedom to model a wide variety of fault and error dynamics, particularly as observed in the Bendix and CSDL studies. The model is intended to model permanent faults only. The following parameters describe the fault and error dynamics:

f = failure rate of a lane (failures/hour)

$p_1$ = proportion of $\alpha$-faults

$p_2$ = proportion of $\beta$-faults which are active at their occurrence

$p_3$ = proportion of $\beta$-faults which are benign at their occurrence

$e_\alpha$ = rate at which $\alpha$-faults produce errors (errors/hour)

$e_\beta$ = rate at which $\beta$-faults produce errors (errors/hour)

as = rate at which auxiliary programs are brought online (programs/hour)

1/ds = average duration of an auxiliary program (hours/program)

q = proportion of auxiliary programs that activate a benign fault

(Henceforth, we will assume that there is only a single auxiliary program. If more than one auxiliary program were involved, the complexity of the model would increase.)

The parameters of the model are not independent. In fact,

$$p_1 + p_2 + p_3 = 1$$

$$p_2 = (1 - p_1)\left[\frac{as}{as + ds}\right]$$

$$p_3 = (1 - p_1)\left[\frac{ds}{as + ds}\right]$$

From the results of the FIIS and previous experiments it was expected to obtain estimates of $p_1$, $p_3$, $e_\alpha$, and $e_\beta$. The parameters $p_2$, as, and ds would be obtained from typical FCS scenarios (e.g., duration of and time between initiation of auxiliary programs).

3.2. Parameter Estimates

- $p_1$: From the Bendix study, $.85 < p_1$; from FIIS, $.96 < p_1 < .98$
- $p_2$: It is estimated that $p_2 = 0$, approximately
  (With $p_2 = 0$, $1 - p_1$ = proportion of latent faults)
- $p_3$: From the Bendix study, $p_3 < .15$; from FIIS, $.02 < p_3 < .04$
- $e_\alpha$: From the FIIS study, $e_\alpha > 7200$/hour
- $e_\beta$: From the FIIS study, $e_\beta > 900$/hour
- as: From FCS scenarios, .01/hour < as < 10/hour
- 1/ds: From FCS scenarios, .01 hours < 1/ds < 1 hour
- f: The current range is .001/hour < f < .0001/hour

3.3. Potential Criticality of Latent Faults

In order to assess the effects of latent faults, we present two examples of conventional FCSs which could be vulnerable to the accumulation of latent faults. The examples noted are Triplex FCS (section 3.4) and Quadruplex FCS (section 3.5). The scenario is one in which latent faults, when activated, lead to a rapid and unexpected loss of components. Such a condition can arise in a Quadruplex FCS, where for example, in time, three out of the four lanes develop latent faults. Eventually, due to a single source of excitation, such as the callup of an outer-loop program, the faults become active. Even assuming, as we do, that the time interval between the successive activation of the faults (referred to, hereafter, as "avalanching latent faults") is sufficient to allow the comparators to detect and isolate each fault in turn, the result is a loss

of the three lanes in quick succession. It is our contention that the probability of such an event may not be insignificant.

3.4. Example 1 - Triplex FCS

The conventional reliability analysis, which assumes that there are no faults at the start of each flight, would conclude that the probability of loss of the FCS function in a flight of one hour is $6f^2$.

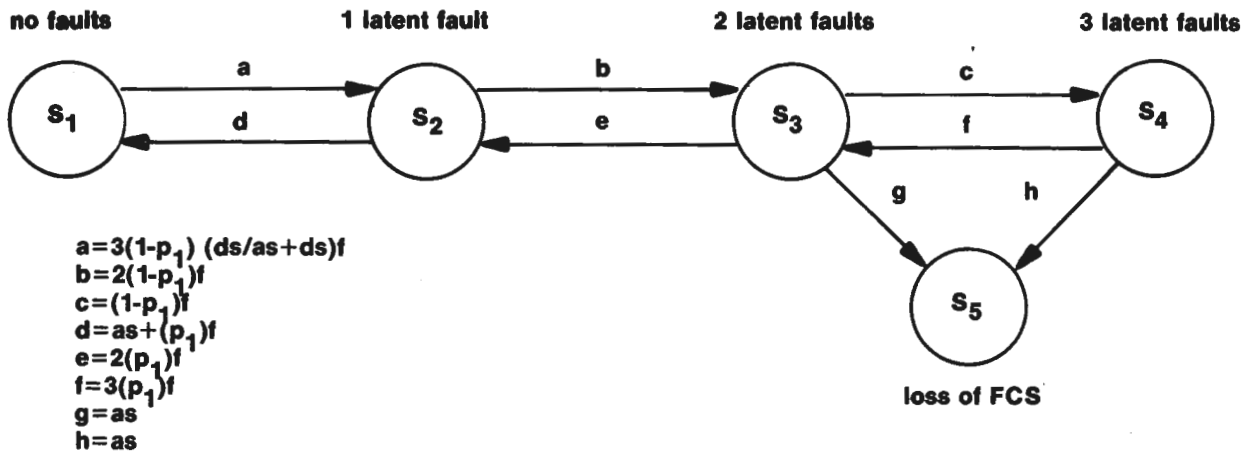



FIGURE 3.4-1. MARKOV MODEL/TRIPLEX FCS WITH LATENT FAULTS

A Markov reliability model of the FCS, representing the effects of avalanching latent faults, is shown in figure 3.4-1. The following assumptions are implicit:

- Loss of control is due, exclusively, to avalanching latent faults.
- Faults are exponentially distributed.
- The time between the successive activation of multiple latent faults is small relative to mission time.
- $\alpha$-faults (i.e., faults active at their occurrence) are detected immediately as they occur, at which time the entire lane is replaced. Thus, existing latent faults are effectively repaired.

- Excitation of latent faults is assumed to be correlated across lanes (i.e., a single excitation activates latent faults in different lanes).

Referring to figure 3.4-1:

- State $S_1$ represents the condition that all lanes are fault-free.
- State $S_2$ has a latent fault in one lane.
- State $S_3$ has a latent fault in two lanes.
- State $S_4$ has a latent fault in three lanes.
- State $S_5$ represents loss of the FCS function.

The transition rates are:

$$a = 3(1-p_1)f[\frac{ds}{as + ds}]$$

$$b = 2(1-p_1)f$$

$$c = (1-p_1)f$$

$$d = as + (p_1)f$$

$$e = 2(p_1)f$$

$$f = 3(p_1)f$$

$$g = as$$

$$h = as$$

It is noted that the backward transitions (d, e, and f) are the result of repairs due to the detection of $\alpha$-faults.

Let $x_1$, $x_2$, $x_3$, $x_4$, and $x_5$ denote the occupancy probabilities of states $S_1$, $S_2$, $S_3$, $S_4$, and $S_5$, respectively. From the figure we obtain the following differential equations:

$$sX_1 - x_1(0) = -aX_1 + dX_2$$

$$sX_2 = aX_1 - (b+d)X_2 + eX_3$$

$$sX_3 = bX_2 - (c+e+g)X_3 + fX_4$$

$$sX_4 = cX_3 - (f+h)X_4$$

$$sX_5 = gX_3 + hX_4$$

with initial conditions, $x_1(0) = 1$, $x_2(0) = x_3(0) = x_4(0) = x_5(0) = 0$. $X_1$, $X_2$, $X_3$, $X_4$, and $X_5$ denote the Laplace transforms of $x_1$, $x_2$, $x_3$, $x_4$, and $x_5$, respectively, and s denotes the Laplace operator.

Figure 3.4-2 shows the mean failure rate (MFR) (see appendix A), for a range of values of $1-p_1$ and f. The parameters $p_2$ and ds were fixed at the values $p_2 = 0$ and ds = 10/hour. Sensitivity studies indicate that the MFR is insensitive to values of ds between 1 and 10 per hour.

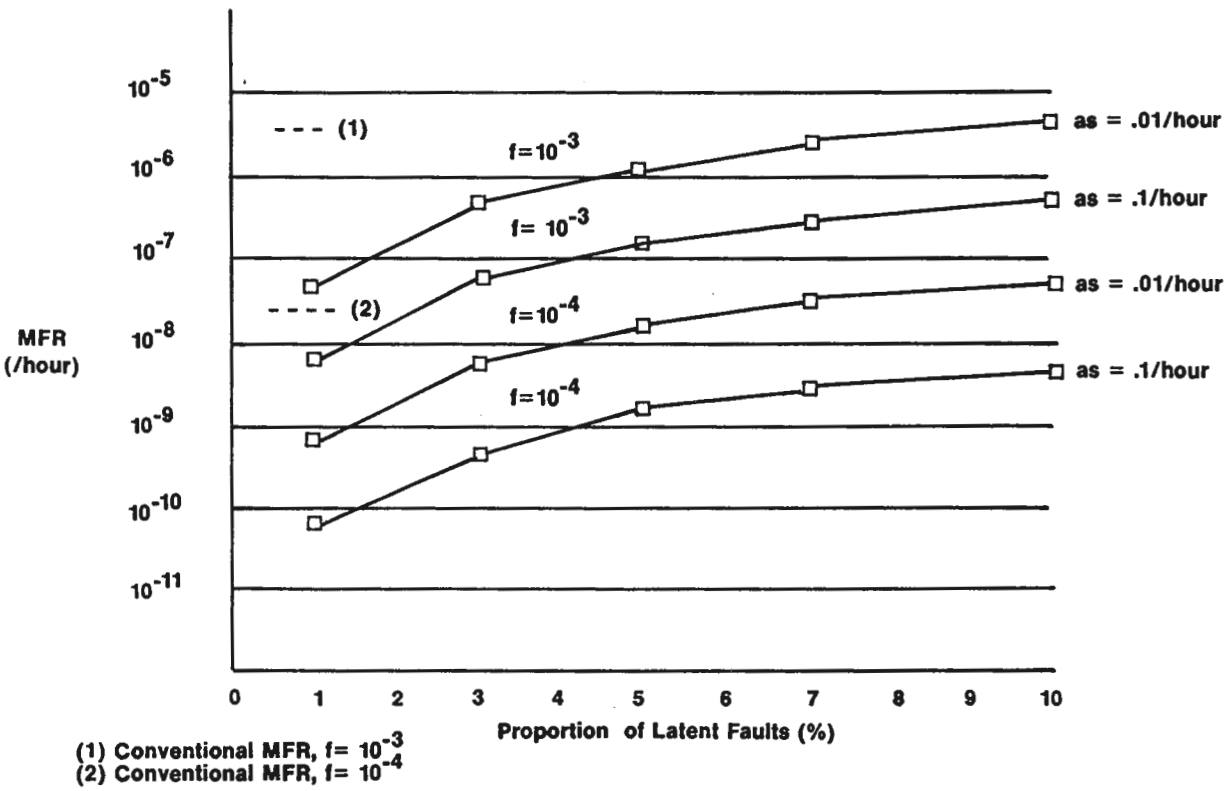



FIGURE 3.4-2. MEAN FAILURE RATE/TRIPLEX FCS

These results are summarized in tables 3.4-1 and 3.4-2 for lane failure rates ($10^{-3}$/hour and $10^{-4}$/hour, respectively). It is noted that the corresponding MFRs, based on conventional reliability analysis, are $6x10^{-6}$/hour and $6x10^{-8}$/hour, respectively.

TABLE 3.4-1. MFR, TRIPLEX FCS, $f = 10^{-3}$/HOUR

| as(/hour) | $1-p_1$(%) | MFR(/hour) |
|---|---|---|
| .1 | 10 | $5.76 \times 10^{-7}$ |
| .1 | 7 | $2.82 \times 10^{-7}$ |
| .1 | 5 | $1.44 \times 10^{-7}$ |
| .1 | 3 | $5.19 \times 10^{-8}$ |
| .1 | 1 | $5.77 \times 10^{-9}$ |
| .01 | 10 | $4.47 \times 10^{-6}$ |
| .01 | 7 | $2.20 \times 10^{-6}$ |
| .01 | 5 | $1.13 \times 10^{-6}$ |
| .01 | 3 | $4.07 \times 10^{-7}$ |
| .01 | 1 | $4.53 \times 10^{-8}$ |

TABLE 3.4-2. MFR, TRIPLEX FCS, $f = 10^{-4}$/HOUR

| as(/hour) | $1-p_1$(%) | MFR(/hour) |
|---|---|---|
| .1 | 10 | $5.92 \times 10^{-9}$ |
| .1 | 7 | $2.90 \times 10^{-9}$ |
| .1 | 5 | $1.48 \times 10^{-9}$ |
| .1 | 3 | $5.33 \times 10^{-10}$ |
| .1 | 1 | $5.92 \times 10^{-11}$ |
| .01 | 10 | $5.81 \times 10^{-8}$ |
| .01 | 7 | $2.85 \times 10^{-8}$ |
| .01 | 5 | $1.45 \times 10^{-8}$ |
| .01 | 3 | $5.23 \times 10^{-9}$ |
| .01 | 1 | $5.82 \times 10^{-10}$ |

3.4.1. Mean Failure Rate Approximation

An approximate value of MFR can be obtained as follows:

On the average, loss of control will occur if two latent faults occur in a time interval of length 1/as hours and no $\alpha$-faults occur in either of the lanes containing the latent faults. If q denotes this probability, then

$$q \approx 6[(1-p_1)f/as]^2[1 - 2(p_1)f/as]$$
$$\approx 6(1-p_1)^2 f^2 (1/as)^2 \qquad (1)$$

Thus, the Mean Time to Failure (MTTF) is:

$$MTTF \approx (1/as)[\ q + 2(1-q)q + 3(1-q)^2q + \ldots\ ] \quad (2)$$
$$= (1/as)/q$$

$$MFR \approx q(as) \quad (3)$$
$$\approx 6(1-p_1)^2f^2/as$$

3.4.2. Summary of Example 1

- The relative effect of latent faults is conveniently described by the ratio

$$R = \frac{\text{probability of loss of control with latent faults}}{\text{probability of loss of control by conventional analysis}}$$

  For values of $R \ll 1$, the effects of latent faults are insignificant. From the tables, the maximum value of R is .97 (for $f = 10^{-4}$, $as = .01$, $p_1 = .9$).

- Relative to the conventional MFR, a Triplex FCS does not appear to be especially vulnerable to latent faults; in the worst case, survivability is reduced by a factor of two.

- MFR is directly proportional to $(1-p_1)^2$.

- MFR is directly proportional to 1/as, the "time on risk."

- Survivability is adversely affected by the proportion of latent faults, $1 - p_1$, although for the range of parameters selected, this effect is not significant.

3.5. Example 2 - Quadruplex FCS

The conventional reliability analysis would indicate that probability of loss of the FCS function in a flight of one hour = $24f^3$.

A Markov reliability model of the FCS is shown in figure 3.5-1. The assumptions are the same as for the triplex system.

Referring to figure 3.5-1:

- State $S_1$ represents the condition that all lanes are fault-free.

- State $S_2$ has a latent fault in one lane.

- State $S_3$ has a latent fault in two lanes.

- State $S_4$ has a latent fault in three lanes.

- State $S_5$ has a latent fault in four lanes.

- State $S_6$ represents loss of the FCS function.

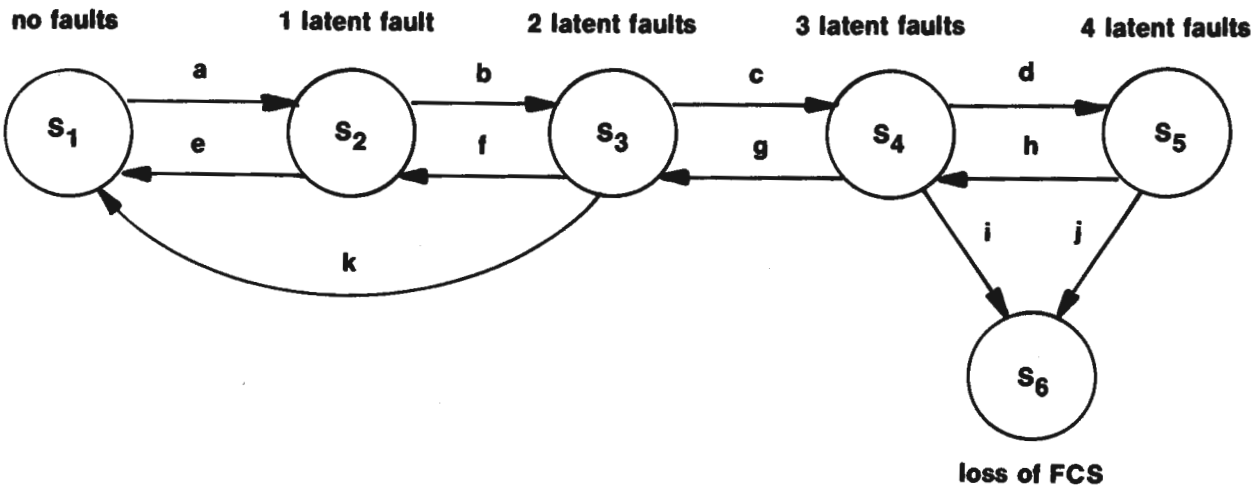



FIGURE 3.5-1. MARKOV MODEL/QUADRUPLEX FCS WITH LATENT FAULTS

The transition rates are:

$$a = 4(1-p_1)f[\frac{ds}{as + ds}]$$

$$b = 3(1-p_1)f$$

$$c = 2(1-p_1)f$$

$$d = (1-p_1)f$$

$$e = as + (p_1)f$$

$$f = 2(p_1)f$$

$$g = 3(p_1)f$$

$$h = 4(p_1)f$$

$$i = as$$

$$j = as$$

$$k = as$$

Again, it is noted that the backward transitions (e, f, g, h, and k) are the result of repairs due to the detection of $\alpha$-faults.

From the figure we obtain the following differential equations:

$$sX_1 - x_1(0) = -aX_1 + eX_2 + kX_3$$

$$sX_2 = aX_1 - (b + e)X_2 + fX_3$$

$$sX_3 = bX_2 - (c + f + k)X_3 + gX_4$$

$$sX_4 = cX_3 - (d + g + i)X_4 + hX_5$$

$$sX_5 = dX_4 - (h + j)X_5$$

$$sX_6 = iX_4 + jX_5$$

with initial conditions, $x_1(0) = 1$, $x_2(0) = x_3(0) = x_4(0) = x_5(0) = x_6(0) = 0$. $X_1$, $X_2$, $X_3$, $X_4$, $X_5$, and $X_6$ denote the Laplace transforms of $x_1$, $x_2$, $x_3$, $x_4$, $x_5$, and $x_6$, respectively.

Figure 3.5-2 shows the MFR for a range of $1-p_1$ and f with $p_2$ and ds again fixed at $p_2 = 0$, ds = 10/hour. These results are summarized in tables 3.5-1 and 3.5-2 for lane failure rates, $10^{-3}$/hour and $10^{-4}$/hour, respectively.

It is noted that the corresponding MFRs, based on conventional reliability analysis, are $2.4 \times 10^{-8}$/hour and $2.4 \times 10^{-11}$/hour, respectively.

TABLE 3.5-1. MFR, QUADRUPLEX FCS, F = $10^{-3}$/HOUR

| as(/hour) | $1-p_1$(%) | MFR(/hour) |
|---|---|---|
| .1 | 10 | $2.23 \times 10^{-9}$ |
| .1 | 5 | $2.80 \times 10^{-10}$ |
| .1 | 3 | $6.04 \times 10^{-11}$ |
| .1 | 1 | $2.24 \times 10^{-12}$ |
| .01 | 10 | $1.37 \times 10^{-7}$ |
| .01 | 5 | $1.73 \times 10^{-8}$ |
| .01 | 3 | $3.75 \times 10^{-9}$ |
| .01 | 1 | $1.39 \times 10^{-10}$ |

TABLE 3.5-2. MFR, QUADRUPLEX FCS, F = $10^{-4}$/HOUR

| as(/hour) | $1-p_1$(%) | MFR(/hour) |
|---|---|---|
| .1 | 10 | $2.36 \times 10^{-12}$ |
| .1 | 5 | $2.95 \times 10^{-13}$ |
| .1 | 3 | $6.38 \times 10^{-14}$ |
| .1 | 1 | $2.37 \times 10^{-15}$ |
| .01 | 10 | $2.25 \times 10^{-10}$ |
| .01 | 5 | $2.82 \times 10^{-11}$ |
| .01 | 3 | $6.1 \times 10^{-12}$ |
| .01 | 1 | $2.26 \times 10^{-13}$ |

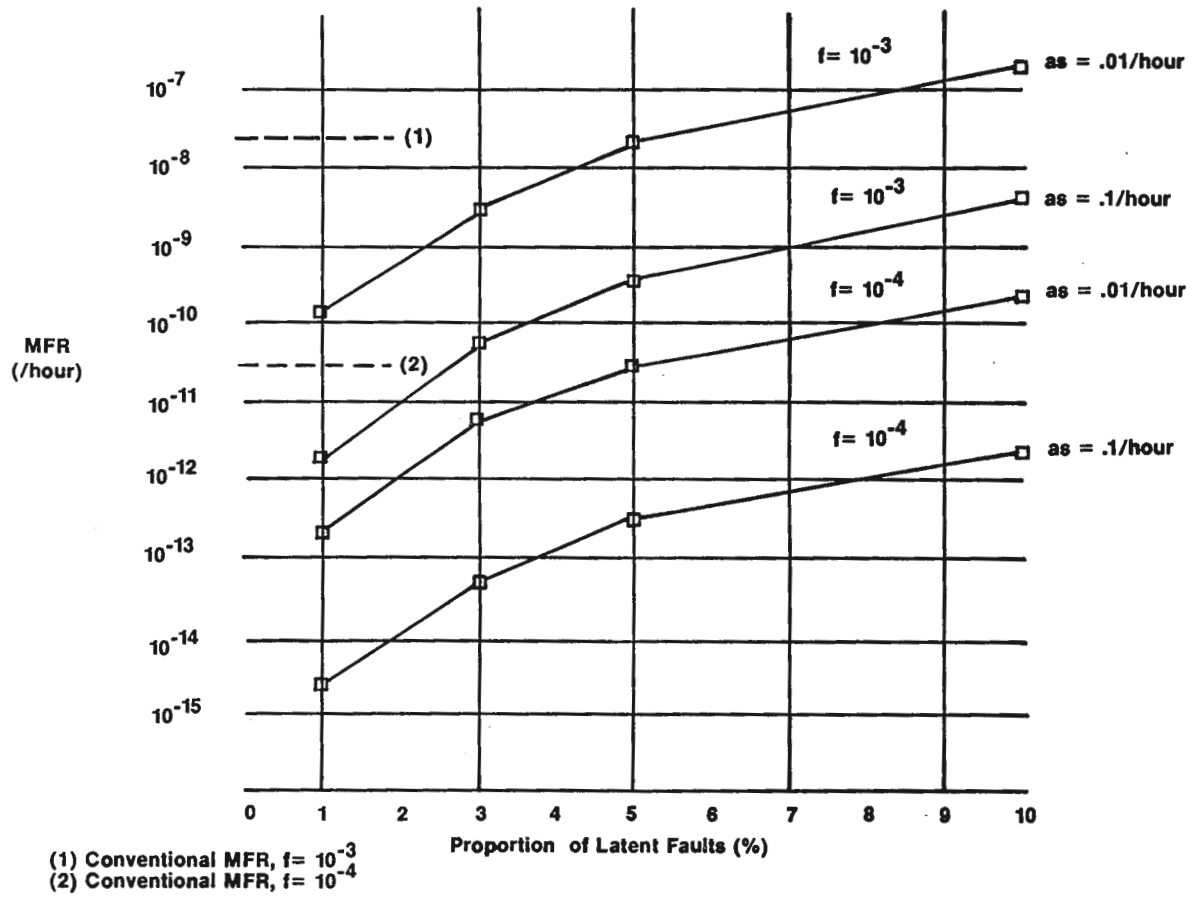



FIGURE 3.5-2. MEAN FAILURE RATE, QUADRUPLEX FCS

### 3.5.1. Mean Failure Rate Approximation

An approximate value of MFR for the Quadruplex FCS can be obtained as follows:

On the average, loss of control will occur if three latent faults occur in a time interval of length 1/as hours and no $\alpha$-faults occur in any of the lanes containing the latent faults. If q denotes this probability, then

$$q \approx 24[(1-p_1)f/as]^3[1 - 3(p_1)f/as] \approx 24(1-p_1)^3f^3/as^3 \quad (4)$$

$$MTTF \approx (1/as)[\, q + 2(1-q)q + 3(1-q)^2q + \ldots] = (1/as)/q \quad (5)$$

$$MFR \approx q(as) \approx 24(1-p_1)^3f^3/as^2 \quad (6)$$

### 3.5.2. Summary of Example 2

- For the range of parameters selected, the maximum value of R is 10 ($f = 10^{-4}$, $as = .01$, $p_1 = .9$), indicating that latent faults could have a significant effect on survivability.
- Relative to the conventional MFR, a Quadruplex FCS is more vulnerable to latent faults than a Triplex FCS.
- MFR is directly proportional to $(1-p_1)^3$.
- MFR is directly proportional to $(1/as)^2$. Thus, for example, increasing the "time on risk", 1/as, by a factor of 10 increases the MFR by a factor of 100.

## 3.6. Self-Test

From the preceding examples, it is apparent how self-test can reduce the degrading effects of latent faults: by reducing the proportion of latent faults, $1-p_1$. If it is assumed that self-test coverage of $\alpha$- and $\beta$-faults is the same, then if 1-u denotes self-test coverage, the proportion of latent faults can be reduced to $u(1-p_1)$. In the Triplex FCS a self-test coverage of only 90 percent reduces the MFR by a factor of 100 and, in the Quadruplex FCS, by a factor of 1000.

## 4. UNRESOLVED ISSUES

Although the Bendix, CSDL, and FIIS studies have contributed significantly to our understanding of fault, error, and detection dynamics, a number of important issues connected with FCS survivability assessment remain unresolved.

- Failure Modes of Digital Devices

  Previous fault injection experiments have employed stuck-at faults either on device pins or on internal gate nodes. It remains to be determined to what extent stuck-at faults represent failure modes of real devices.

- Pin-Level Versus Gate-Level Faults

  Until actual failure mode data becomes available, experimenters will, no doubt, continue to inject pin-level and gate-level faults. The Bendix study indicated a significant difference between pin-level and gate-level detection coverage. At issue is the validity of either fault type and the extent to which one is more or less latent than the other.

- Single-Fault Model Incorporating Intermittent/Transient Faults

  The single-fault model of figure 3.1-1 appears to be adequate for permanent faults. The model should be extended to include intermittent and transient faults. Estimates of occurrence, duration, and reoccurrence rates are required.

- Effects of Periodic Maintenance

  Periodic maintenance can have a significant effect on reducing latent faults to acceptable levels. Unfortunately, incorporating maintenance effects in reliability models leads to computational difficulties. McGough, Reibman, and Trivedi (to be published) offers one approach to this problem.

- Multiple-Fault Models

  In order to perform a reliability assessment of an FCS it is essential to model multiple fault effects. The multiple-fault models of figures 3.4-1 and 3.5-1 are too rudimentary to be of general use. Required is a more comprehensive multiple-fault model which includes the correlation of latent faults across different lanes.

# 5. SUPPLEMENTAL WORKED EXAMPLE

In the two examples of section 3, the assessment of latent faults was based on several assumptions e.g., the structure of the single-fault model and the correlation of excitation across different lanes.

In this section another perspective of the effects and potential criticality of latent faults on aircraft survivability is presented.

In general, the effects and criticality of latent faults are dependent upon the detailed FCS design, in particular, the levels of redundancy and the fault detection, isolation, and recovery mechanisms employed. As a consequence, the strawman FCS selected for this example should not be construed as being typical or even representative of FCSs. It was selected for the following reasons:

- It is a realistic FCS; similar configurations have been used in the past.
- The associated fault and reliability models are relatively simple and easily understood; the state occupancy probabilities can be obtained analytically.

## 5.1. FCS Configuration

The FCS is required to increase the fatigue life of the aircraft. This is achieved by improving the damping of the short period and dutch roll modes and controlling the phase of one or two bending modes. The aircraft, however, is controllable without the FCS. Thus, loss of the FCS function does not, in any way, impair the safety of the vehicle. However, an undetected fault could result in an unsafe condition. Thus, it is a firm requirement that such faults be precluded, at least to a probability of $10^{-7}$/hour, the generally accepted goal for military aircraft.

The FCS configuration is shown in figure 5.1-1. Referring to the figure:

- The FCS consists of two independent lanes, each containing a processor (P) and dedicated disengage logic (Q and associated relay).
- Each processor drives a different actuator.
- Each processor receives information from an independent set of sensors.
- Each processor computes the commands for its own and the other actuator and compares both commands with those of the other processor. This data is exchanged via the intercomputer data links. Any observed difference, exceeding a comparator threshold, will cause that processor to send a discrete to both $Q_1$ and $Q_2$, opening the relays and disengaging both actuators.

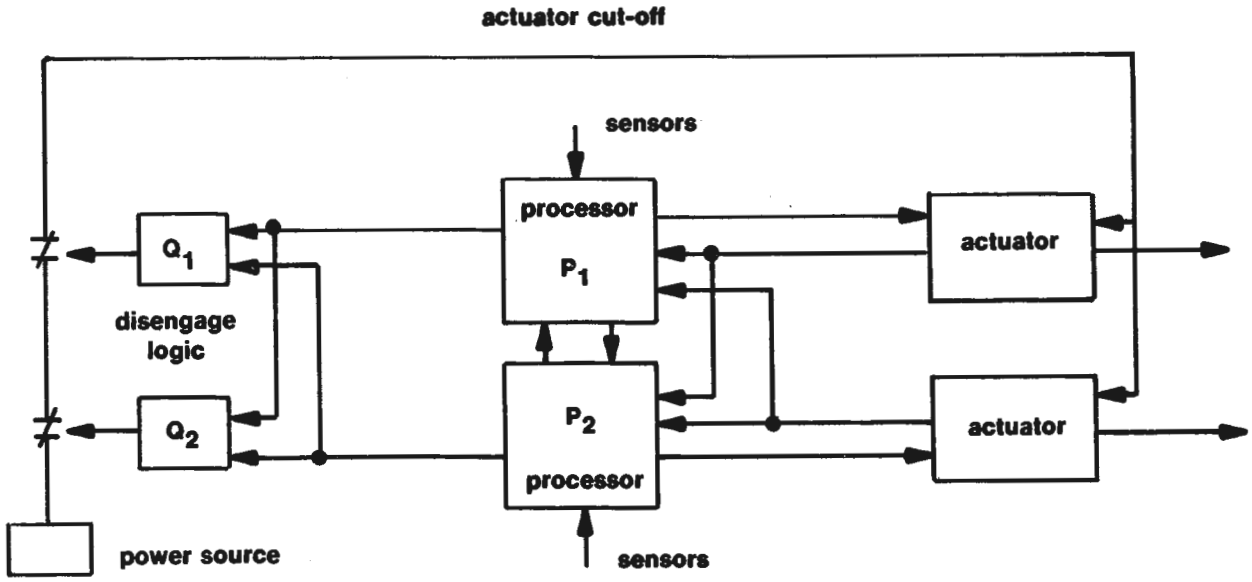



FIGURE 5.1-1. DUAL, FAIL SAFE FLIGHT CONTROL SYSTEM

## 5.2. Unsafe Condition

An unsafe condition can arise only if a processor fails and it and the non-failed processor are unable to disengage the actuators. Such an event could only occur if both disengage elements, $Q_1$ and $Q_2$, have failed previously. In practice, the disengage logic would be tested in preflight test. The issue is, what level of failure detection coverage is required?

## 5.3. Repair of Latent Faults

In generating the reliability model it must be determined when and how latent faults are repaired. In this example it is assumed that a detected fault will result in the replacement of the entire lane, both processor and disengage logic, by non-faulted components.

## 5.4. Markov Reliability Model

The reliability model of the FCS is shown in figure 5.4-1. The following assumptions are implicit:

- The model only includes the effects of latent faults.
- Faults are exponentially distributed.
- Mission duration is small relative to the mean time between failures.

- All faults in the disengage logic, $Q_1$ and $Q_2$, are latent (i.e., they produce no effects until a fault in $P_1$ or $P_2$ occurs).
- Any fault in the disengage logic prevents disengagement via the associated relay.
- All faults in the processors are detected immediately as they occur at which time the entire lane is replaced.
- A fault, undetected by preflight test, will remain undetected no matter how often the test is executed.
- All faults in $P_1$ and $P_2$ are detected by preflight test.

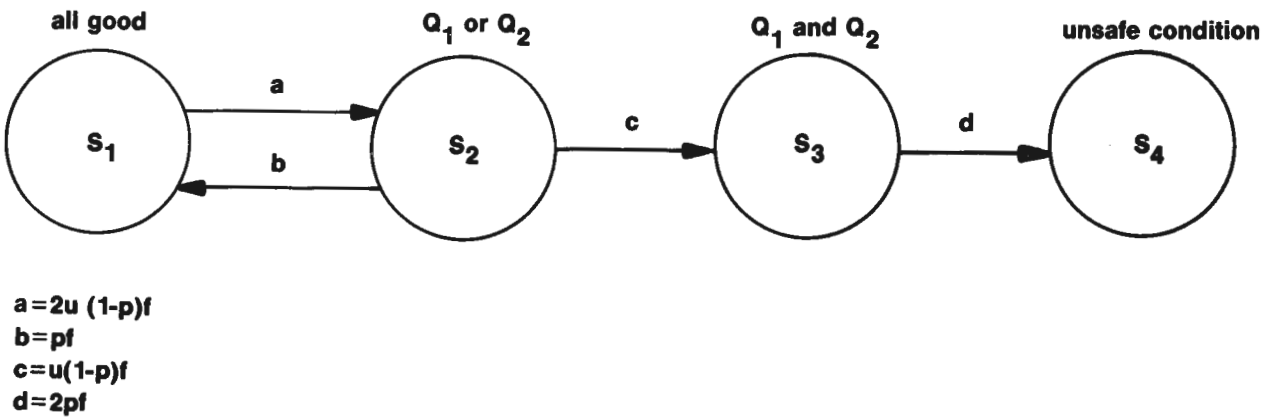



FIGURE 5.4-1. MARKOV MODEL/DUAL, FAIL SAFE, FLIGHT CONTROL SYSTEM

Referring to figure 5.4-1:

- State $S_1$ represents the condition that all lanes are fault-free.
- State $S_2$ represents the condition of a latent fault in $Q_1$ or $Q_2$ but not both.
- State $S_3$ represents the condition of latent faults in both $Q_1$ and $Q_2$.
- State $S_4$ represents the unsafe condition.
- A transition from $S_1$ to $S_2$ occurs when a fault occurs in $Q_1$ or $Q_2$.

- A transition from $S_2$ to $S_1$ occurs if $Q_1$ ($Q_2$) fails and $P_1$ ($P_2$) fails subsequently. Thus, the latent fault is repaired.

- A transition from $S_2$ to $S_3$ occurs when a second logic element fails.

- A transition from $S_3$ to $S_4$ occurs when both $Q_1$ and $Q_2$ have failed and a processor fails subsequently.

5.4.1. Model Parameters

- f = failure rate of a lane (failures/hour)

- pf = failure rate of a processor (failures/hour)

- (1-p)f = failure rate of the disengage logic (failures/hour)

- u = proportion of faults in the disengage logic not detected by pre-flight test

- a = 2u(1-p)f

- b = pf

- c = u(1-p)f

- d = 2pf

- 1-p = proportion of faults in the disengage logic

- p = proportion of faults in the processor, $P_1$ or $P_2$

Typical parameter values are

$$10^{-4}/\text{hour} < f < 10^{-3}/\text{hour}$$

$$.9 < p < 1$$

$$.9 < u < 1$$

5.5. Effects of Latent Faults on Survivability

Let $x_1$, $x_2$, $x_3$, and $x_4$ denote the occupancy probabilities of states $S_1$, $S_2$, $S_3$, and $S_4$, respectively. From the figure we obtain the following differential equations:

$$sX_1 - x_1(0) = -aX_1 + bX_2$$

$$sX_2 = aX_1 - (b+c)X_2$$

$$sX_3 = cX_2 - dX_3$$

$$sX_4 = dX_3$$

with initial conditions, $x_1(0) = 1$, $x_2(0) = x_3(0) = x_4(0) = 0$. $X_1$, $X_2$, $X_3$, and $X_4$ denote the Laplace transforms of $x_1$, $x_2$, $x_3$, and $x_4$, respectively, and s denotes the Laplace operator.

We could solve for $x_4(t)$, the probability of an unsafe condition, analytically, as a function of time. Instead, we compute the MFR. As shown in Appendix A:

$$MTTF = -\lim_{s \to 0} sX_4(s)/ds \tag{7}$$

We can solve directly to obtain

$$X_4(s) = \frac{acd}{s(s + d)(s^2 + (a + b + c)s + ac)} \tag{8}$$

$$MTTF = \frac{ac + d(a + b + c)}{acd} \tag{9}$$

$$MFR = \frac{acd}{ac + d(a + b + c)} = \frac{4u^2p(1-p)^2f}{2u^2(1-p)^2 + 2p[3u(1-p) + p]} \tag{10}$$

Setting $v = u\left(\frac{1-p}{p}\right)$,

$$MFR = \frac{2v^2pf}{v^2 + 3v + 1} \quad (\approx 2v^2pf) \tag{11}$$

5.6. The Conventional Reliability Analysis

The conventional reliability analysis assumes that no faults exist at the start of a flight. In this case, an unsafe condition can occur if and only if $Q_1$ and $Q_2$ both fail and $P_1$ or $P_2$ fails subsequently. Assuming a one-hour flight, this probability is given by

$$q = (1-p)^2f^2(2pf)/6 = p(1-p)^2f^3/3 \tag{12}$$

This event is order-dependent (i.e., $Q_1$ and $Q_2$ must both fail before $P_1$ or $P_2$). It can be shown that the order of occurrence is equiprobable if the flight time is small relative to 1/f. There are 12 possible orderings of $Q_1$, $Q_2$, $P_1$, and $P_2$, taken three at a time, but only four result in an unsafe condition, for example, $(Q_1,Q_2,P_1)$, $(Q_1,Q_2,P_2)$, $(Q_2,Q_1,P_1)$, $(Q_2,Q_1,P_2)$.

Employing Bernoulli trials, the mean time to first failure (i.e., an unsafe condition) is

$$MTTF = 1/q$$

and, thus,

$$q = MFR \approx p(1-p)^2 f^3/3 \tag{13}$$

Comparing (11) and (13) it can be seen that the MFR, due exclusively to latent fault conditions, is proportional to f whereas the conventional MFR is proportional to f cubed.

5.7. Numerical Example

To assess the relative effects of latent fault accumulation on system survivability we give several numerical examples, assuming p = .95 and f = $10^3$/hour.

Case 1. No preflight test (i.e., u = 1).

MFR(latent) = $4.53 \times 10^{-6}$/hour.

Case 2. Preflight test with u = 0.1 (90 percent fault detection coverage).

MFR(latent) = $4 \times 10^{-8}$/hour.

In contrast, the conventional MFR is

MFR(conventional) = $7.91 \times 10^{-13}$/hour.

5.8. Summary of Example

From this simple example, it can be seen that the accumulation of latent faults can have a profound effect on system survivability. In fact, they are often the dominant cause of system loss. As a consequence, it is incumbent on the system designer to assess the vulnerability of the FCS to such faults and, if significant, establish self-test coverage requirements accordingly.

# 6. CONCLUSIONS

In the Introduction, four questions were raised in connection with latent faults. As the examples given in this tutorial illustrate, the answers depend upon the details of the FCS design, the levels of redundancy, the fault detection, isolation, and recovery mechanisms employed, and the survivability goals of the FCS. As a consequence, we venture to give some answers but only with respect to the FCSs presented in this tutorial.

- What proportion of faults are latent?

  In the FCSs used in the Bendix, CSDL, and FIIS experiments the proportion of latent faults ranged between 2 percent and 15 percent. A less pessimistic estimate, based on the FIIS experiments, indicates a range between 2 percent and 4 percent. In the FCS of section 5, the proportion is a function of the relative failure rate of the disengage logic.

- What mechanisms activate latent faults?

  Latent faults are activated when the faulty components are exercised. In the systems used in the Bendix, CSDL, and FIIS experiments latent faults appear to be activated primarily by software programs. In the FCS of section 5, latent faults are activated when faults occur in other components.

- What are the effects of latent faults on aircraft survivability?

  In the two examples of section 3 these effects are given in figures 3.4-2 and 3.5-2. In the FCS of section 5 these effects are given by equation (10). Obviously, the effects of latent faults are highly dependent on the FCS configuration. In any case, their significance can only be evaluated relative to the survivability goal of the FCS. (In the examples given, this goal was assumed to be the survivability as determined by conventional analysis.)

- If latent faults are a contributing factor to reduced survivability, what level of coverage is required to detect and eliminate them?

  In the examples given in sections 3 and 5, a modest coverage could reduce these effects to insignificance (e.g., a 90 percent coverage will improve survivability by several orders of magnitude).

## APPENDIX A - MEAN FAILURE RATE

### Occupancy Probability

A homogeneous Markov model is described by a set of linear differential equations of the form

$$dx/dt = Ax \tag{1}$$

where

$A = (a_{ij}) = nxn$, is a constant matrix of transition rates, and

$x = col(x_1, x_2, \ldots, x_n)$, is a vector of occupancy probabilities of states S1, S2,..., Sn, respectively.

In most FCS reliability models there is exactly one absorbing state, the loss of control state, which we designate as $S_n$. The associated occupancy probability, $x_n(t)$, is then a cumulative distribution function and can be expressed in the form

$$dx_n(t)/dt = a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn-1}x_{n-1} \tag{2}$$

In many cases, $x_n(t)$ is not a convenient measure of survivability. For example:

- For models with large numbers of states or models which contain cyclic paths (Marie, Reibman, and Trivedi, 1987), the computation of $x_n(t)$ requires the aid of a reliability analysis program such as ARIES, CARE III and HARP.

- In the conventional reliability model the FCS is assumed to be completely repaired at the start of each flight. Thus, the probability of loss of control during the first flight is the same for all flights. When latent faults are present, however, the FCS may only be partially repaired at the start of a flight. Consequently, the probability of loss of control is different for each flight, tending to increase with time. In these scenarios the occupancy probability of loss of control is not a practicable measure of survivability because it requires a recursive solution of the Markov model over a large number of flights.

We propose an alternate measure of survivability, called MFR. The MFR is defined as the reciprocal of the MTTF (McGough, Reibman, and Trivedi [to be published]).

The advantages of the MFR criteria are as follows:

- It is relatively easy to compute, even for models with cyclic paths, requiring only solutions of linear, algebraic equations.

- It is independent of time.
- It characterizes survivability over the life of the airplane.
- It provides a relative measure of survivability even if rejected as an absolute measure.

The disadvantage of MFR is that it gives no indication of how survivability degrades with time.

Computation of Mean Failure Rate

Taking the Laplace transform of both sides of equation (2) gives, for $x_n(0) = 0$,

$$sX_n(s) = a_{n1}X_1(s) + a_{n2}X_2(s) + \ldots + a_{nn-1}X_{n-1}(s) \tag{3}$$

where

$X_k(s)$ is the Laplace transform of $x_k(t)$ and s is the Laplace operator.

Using the moment generating property of Laplace transforms:

$$\begin{aligned} MTTF &= -\lim_{s\to 0} sdX_n(s)/ds \\ &= -[anlX_1'(s) + an2X_2'(s) + \ldots, + ann\text{-}1X_{n-1}'(s)] \end{aligned} \tag{4}$$

where the prime denotes the derivative with respect to s.

If we let $X(s) = \text{col}\,[X_1(s), X_2(s), \ldots, X_n(s)]$, equation (1) can be expressed as

$$sX(s) - x(0) = AX(s) \tag{5}$$

Differentiating both sides of equation (5), we obtain

$$X(s) + sX'(s) = AX'(s) \tag{6}$$

Setting $s = 0$ gives

$$X(0) = AX'(0) \tag{7}$$

The quantities $X_1(0)$, $X_2(0)$, ..., $X_{n-1}(0)$ are obtained by setting $s = 0$ in equation (5) and solving the resultant linear system by using the method of Gaussian elimination or an iterative method like Gauss-Seidel. Substituting these values into the first n-1 equations in (7) we can solve for $X_1'(0)$, $X_2'(0)$, ..., $X_{n-1}'(0)$. A unique solution exists because $S_n$ is the only absorbing state. The resulting values $X_1'(0)$, $X_2'(0)$, ..., $X_{n-1}'(0)$ are substituted into equation (4) to obtain the MTTF. Then

$$MFR = 1/MTTF.$$

BIBLIOGRAPHY

Benson, W., Mulcare, D., and Larsen, W., Hardware Fault Insertion and Instrumentation System: Experimentation and Results, DOT/FAA/CT-86/34, FAA Technical Center, March 1987.

Lala, J. and Smith, T. B., Development and Evaluation of a Fault-Tolerant Multiprocessor (FTMP) Computer - Volume III, FTMP Test and Evaluation, NASA CR 166073, NASA Langley Research Center, May 1983.

Marie, R. A., Reibman, A. L., and Trivedi, K. S., Transient Solution of Acyclic Markov Chains, Performance Evaluation, Vol.7, No.3, 1987, pp. 175-194.

McGough, J. G., Reibman, A. L., and Trivedi, K. S., "Markov Reliability Models for Digital Flight Control Systems," AIAA Journal of Guidance and Control (to be published).

McGough, J. G. and Swern, F., Measurement of Fault Latency in a Digital Avionic Processor, Part II, NASA CR 3651, NASA Langley Research Center, January 1983.

## GLOSSARY

AVALANCHING LATENT FAULTS. The successive activation of latent faults.

MEAN FAILURE RATE. A measure of survivability defined as the reciprocal of the mean time to system failure.

BASELINE PROGRAM. A set of continuously executed software modules.

AUXILIARY PROGRAMS. Software executed occasionally.

ACTIVE FAULT. A fault that can produce an error (for some input) while executing the current program.

$\alpha$-FAULT. A fault activated by the baseline program.

$\beta$-FAULT. A fault not activated by the baseline program.

BENIGN FAULT. A fault that cannot produce an error while executing the current program, regardless of input, but may produce an error for some other program.

LATENT FAULT. A fault which has not yet produced a malfunction. (In the context of the single-fault model, benign and latent faults are equivalent.)

## ACRONYMS AND ABBREVIATIONS

| | |
|---|---|
| ALU | Arithmetic Logic Unit |
| BGU | Bus Guardian Unit |
| CPU | Central Processing Unit |
| CSDL | Charles Stark Draper Laboratories |
| FCC | Flight Control Computers |
| FCS | Flight Control System |
| FIIS | Fault Insertion and Instrumentation System |
| FTMP | Fault-Tolerant Multiprocessor |
| GS | Glideslope |
| I/O | Input/Output |
| MFR | Mean Failure Rate |
| ms | Millisecond |
| MTTF | Mean Time to Failure |
| P | Processor |
| PROM | Programmable Read-Only Memory |
| RAM | Random Access Memory |
| ROM | Program Memory |
| S-a-0 | Stuck at Zero |
| S-a-1 | Stuck at One |
| TMR | Triple Modular Redundant |

U.S. Department of Transportation
Federal Aviation Administration

DOT/FAA/CT-88/10

# HANDBOOK-VOLUME II
# DIGITAL SYSTEMS VALIDATION

## CHAPTER 11
## AIRCRAFT ELECTROMAGNETIC COMPATIBILITY
## (GUIDELINES TO ASSESS EMC DESIGNS)

**PREPARED BY:**

**COMPUTER RESOURCE MANAGEMENT, INC.**
**950 HERNDON PARKWAY, SUITE 360**
**HERNDON, VIRGINIA 22070**

**PREPARED FOR:**

**FEDERAL AVIATION ADMINISTRATION**
**TECHNICAL CENTER**
**ATLANTIC CITY INTERNATIONAL AIRPORT, NEW JERSEY 08405**

NOTICE

This document is disseminated under the sponsorship of the U.S. Department of Transportation in the interest of information exchange. The United States Government assumes no liability for the contents or use thereof.

The United States Government does not endorse products or manufacturers. Trade or manufacturers' names appear herein solely because they are considered essential to the objective of this report.

# PREFACE

This document is based on work performed by the Boeing Commercial Airplane Company, P.O. Box 3707, Seattle, Washington 98124, under the National Aeronautics and Space Administration (NASA) contract NASA-12261 for the Federal Aviation Administration (FAA), NASA-Ames Research Center, Moffett Field, California 94035. The contract was firm fixed price, level-of-effort term from September 1985 to September 1986 with an extension to June 1987. Contracting officers were W. C. Botts, Boeing and A. N. Johnson, NASA. The FAA contracting officer's technical representative (COTR) was William E. Larsen. Deliverables were an Interim Report (Draft) and an Interim Report (Final). The program manager was Robert D. Force, and principal investigator, Clifton A. Clarke.

Bob Force, who helped put the program together, played a central part in managing the planning and organization of the total document. The section 2.1, "Existing Systems," is derived from the valuable "Active Controls Technology" report which Bob co-authored. Bill Larsen provided extensive expertise taken from his own experience, and through constructive source material. He also provided sound and highly regarded recommendations for content and organization. Dale R. Reed collaborated on the wiring-induced voltages and worked some of the in-depth computations. He offered very profitable perspectives on the approach to aircraft engineering analysis and design; his forbearance and tenacity are appreciated.

Special contributions were made by John Tinner and John Bishop who consulted on significant aircraft test and troubleshooting procedures, helping to fill in the picture of aircraft electromagnetic interference. Veteran EMC engineers who supplied valued and time-tested data that form a part of this report were Jerry Carter, John Foster, and George Ketterling. Thanks are also due those individuals mentioned or quoted in the text.

Many people contributed important comments and helpful criticisms to the Interim Report (Draft). Chris Kendall supplied valuable consultations and comments along with Roger McConnell of CKC Associates. Henrietta Gilbert, FAA; Richard Hess, Sperry Corp.; Russell Carstensen, Naval Air Systems Command; and Kary Miller of Collins generously took time to review and comment with useful corrections and suggestions. My thanks are also owed to Nancy Clarke for helpful editorial comments. Fellow EMC engineers Glenn Olson, Kieth Kalanquin, Charles King, and Sy O'Young (who worked on the proposal) contributed their thoughts.

The document format and graphics were expertly delineated by Primo Mattieligh and drawn by Irene Ohashi. Their generosity and patience are much appreciated. Gary Breidenstein offered expert aid in the editing and was a source of inspiration in the preparation of the final copy. Nancy Eaton not only supervised the typing but helped proof the manuscript.

This document would not be possible without the unparalleled IEEE Symposium Records and the periodical "ITEM", R & B Enterprises, whose presentations were a source of valuable data applicable to an aircraft.

TABLE OF CONTENTS


| Section | | Page |
|---|---|---|
| 1. | INTRODUCTION | 11-1 |
| 1.1. | Background and Scope | 11-1 |
| 1.1.1. | A Case of Engine Shutdown | 11-1 |
| 1.1.2. | Electromagnetic Compatibility | 11-2 |
| 1.2. | Electromagnetic Compatibility Priorities | 11-8 |
| 1.2.1. | Responsibilities and Policies | 11-8 |
| 1.2.2. | Documentation | 11-11 |
| 1.3. | How to Use This Document | 11-11 |
| 2. | AIRPLANE AVIONICS AND CRITICALITIES | 11-15 |
| 2.1. | Existing Systems | 11-15 |
| 2.2. | Future Aircraft | 11-31 |
| 2.3. | Criticalities | 11-42 |
| 2.3.1. | Measures and Definitions | 11-42 |
| 2.3.2. | Critical Equipment | 11-44 |
| 2.4. | Packaging and Architecture | 11-46 |
| 2.5. | Issues | 11-49 |
| 2.5.1. | An Early Start | 11-49 |
| 2.5.2. | Design | 11-49 |
| 2.5.3. | Environment and Test | 11-50 |
| 2.6. | Variances in Electromagnetic Compatibility | 11-51 |
| 2.6.1. | Diagnostics and Troubleshooting | 11-51 |
| 2.6.2. | Table of Variances | 11-53 |
| 2.6.3. | Corrective Action and Modeling | 11-56 |
| 3. | AVIONICS THRESHOLDS AND PROTECTION | 11-59 |
| 3.1. | Hardware Tolerance | 11-59 |
| 3.2. | EMI and Software | 11-61 |
| 3.3. | Digital and Discrete Circuits | 11-64 |
| 3.3.1. | ARINC 429 Drivers and Receivers | 11-64 |
| 3.3.2. | Circuit and Shield Grounds | 11-65 |
| 3.3.3. | Discretes | 11-67 |

TABLE OF CONTENTS (Continued)


| Section | | Page |
|---|---|---|
| 3.4. | Equipment/Wiring Isolation and Separation | 11-67 |
| 3.4.1. | Quality of Wiring Design | 11-67 |
| 3.4.2. | Power and Energy Levels | 11-71 |
| 3.4.3. | EMC Quality in Maintenance | 11-76 |
| 3.4.4. | Shielding and Shield Ties | 11-76 |
| 3.5. | Aircraft Protection Measures | 11-77 |
| 3.5.1. | Structure Conductivity | 11-77 |
| 3.5.2. | Shielding | 11-81 |
| 3.5.3. | Safety in Grounding and Returns | 11-84 |
| 3.5.4. | Resistance | 11-87 |
| 3.5.5. | Resonance | 11-90 |
| 3.6. | Composites | 11-90 |
| 4. | EMC ANALYSIS AND ENVIRONMENT | 11-95 |
| 4.1. | Radiated Environment | 11-95 |
| 4.1.1. | Radio Frequency Field Distribution | 11-95 |
| 4.1.2. | Magnetic and Electric Field Distribution | 11-101 |
| 4.1.3. | Transients | 11-104 |
| 4.2. | Aircraft Protection | 11-106 |
| 4.2.1. | Shielding and Ground Reference | 11-106 |
| 4.2.2. | Apertures and Electrical Bonding | 11-107 |
| 4.2.3. | External Wiring Interface Circuits | 11-108 |
| 4.2.4. | Circuit Protection | 11-110 |
| 4.3. | Trades | 11-111 |
| 5. | ACTIVITIES AND DOCUMENTS | 11-117 |
| 5.1. | Specifications/Documents | 11-117 |
| 5.2. | Tasks/Activities | 11-118 |
| 6. | EQUIPMENT SPECIFICATION | 11-121 |
| 6.1. | New and Existing | 11-121 |
| 6.2. | Aircraft Equipment Categories | 11-123 |
| 6.3. | Suppliers | 11-125 |

TABLE OF CONTENTS (Continued)


| Section | | Page |
|---|---|---|
| 7. | VERIFICATION AND VALIDATION | 11-127 |
| | 7.1. Key EMC Designs | 11-127 |
| | 7.2. Validation Plans | 11-128 |
| | 7.3. Validation Procedure | 11-128 |
| | 7.4. Program Design Reviews | 11-131 |
| APPENDIX | | |
| | A - TEST AND TEST LIMITS | 11-133 |
| BIBLIOGRAPHY | | 11-141 |
| GLOSSARY OF TERMS | | 11-149 |
| ACRONYMS AND ABBREVIATIONS | | 11-154 |

LIST OF ILLUSTRATIONS


| Figure | | Page |
|---|---|---|
| E-1 | POSTULATED PERCENTAGES | 11-xi |
| 1.1-1 | REPRESENTATIVE EMI SOURCES | 11-3 |
| 1.1-2 | EMI ENVIRONMENT IN AIRCRAFT | 11-4 |
| 1.1-3 | PERCENT TROUBLESHOOTING-A/C | 11-6 |
| 1.1-4 | ELECTROMAGNETIC INTERFERENCE EFFECTS | 11-6 |
| 1.1-5 | RADIO FREQUENCY RANGE | 11-7 |
| 1.2-1 | COST-EFFECTIVE SUBSYSTEM TESTING | 11-9 |
| 1.2-2 | KEY EMC FACILITATORS | 11-10 |
| 1.3-1 | REPRESENTATIVE "E" FIELD COUPLING | 11-12 |
| 2.1-1 | PRESENT-DAY AIRCRAFT SYSTEMS | 11-16 |
| 2.1-2 | AVIONICS BAYS | 11-17 |
| 2.1-3 | WIRING | 11-17 |
| 2.1-4 | MAIN BAY ALUMINUM STRUCTURE | 11-17 |
| 2.1-5 | STRUCTURE RETURN | 11-18 |
| 2.1-6 | INDUCED NOISE | 11-18 |
| 2.1-7 | NOISE VOLTAGES | 11-19 |
| 2.1-8 | NONMETALLIC INTERFACES | 11-19 |
| 2.1-9 | SEPARATION EXAMPLE | 11-21 |
| 2.1-10 | FLIGHT CONTROL AUTOPILOT REDUNDANCY | 11-21 |
| 2.1-11 | ELECTRONIC ENGINE CONTROL | 11-22 |
| 2.1-12 | TRANSMITTERS-RECEIVERS | 11-22 |
| 2.1-13 | FLIGHT INSTRUMENT SYSTEM | 11-24 |
| 2.1-14 | ROLL CONTROL SYSTEM | 11-24 |
| 2.1-15 | NAVIGATION | 11-25 |
| 2.1-16 | COMMUNICATION | 11-26 |
| 2.1-17 | LINE REPLACEABLE UNIT | 11-27 |
| 2.1-18 | FLIGHT MANAGEMENT SYSTEMS | 11-28 |
| 2.1-19 | EQUIPMENT/WIRING LOCATION COMPLEXITIES | 11-30 |
| 2.1-20 | RADIATED EMISSION | 11-31 |
| 2.2-1 | EMC AND AIRCRAFT SYSTEMS DEVELOPMENT | 11-32 |
| 2.2-2 | DATA BUS ARCHITECTURE | 11-32 |
| 2.2-3 | LINE-OF-SIGHT DISPLAY AND VOICE CONTROL | 11-34 |
| 2.2-4 | HYPOTHETICAL ARCHITECTURE | 11-35 |
| 2.2-5 | DATA BUS COMPARISON | 11-36 |
| 2.2-6 | DIGITAL DATA BUSES | 11-36 |
| 2.2-7 | PERFORMANCE AND STATUS MONITORS | 11-37 |
| 2.2-8 | MICROPROCESSOR EVOLUTION | 11-38 |
| 2.2-9 | AVIONICS TECHNOLOGY PROGRESS | 11-38 |
| 2.2-10 | MATERIALS DISTRIBUTION | 11-39 |
| 2.2-11 | 1990'S ATC IMPLEMENTATION | 11-39 |
| 2.3-1 | PROBABILITY VERSUS CONSEQUENCE | 11-43 |
| 2.3-2 | THREE-STAGE SUBSYSTEM TESTING | 11-43 |
| 2.3-3 | PROJECTED CRITICALITIES | 11-45 |
| 2.6-1 | A DISCREPANCY | 11-52 |
| 2.6-2 | POSTULATED HALLMARKS OF VARIANCES | 11-57 |

| Figure | | Page |
|---|---|---|
| 3.1-1 | MICROCIRCUIT TOLERANCE | 11-60 |
| 3.1-2 | NOISE MARGIN | 11-60 |
| 3.2-1 | TRANSIENT EVENTS | 11-63 |
| 3.3-1 | CIRCUIT NOISE REJECTION (A B C D E F) | 11-66 |
| 3.4-1 | CIRCUIT RESPONSE TO EMI | 11-68 |
| 3.4-2 | WIRING CATEGORIES | 11-70 |
| 3.4-3 | CRITICAL CIRCUITS WIRE SEPARATION | 11-72 |
| 3.4-4 | SELECTED EMI LEVELS | 11-73 |
| 3.4-5 | LABORATORY TEST SETUP | 11-74 |
| 3.4-6 | SHIELD TIES | 11-78 |
| 3.5-1 | THE GROUNDING STEPS | 11-78 |
| 3.5-2 | STRUCTURE EMC ROLES | 11-79 |
| 3.5-3 | DETAILS AND INSTALLATION | 11-80 |
| 3.5-4 | THE SEVEN EARTH GROUND CONNECTIONS | 11-81 |
| 3.5-5 | EMC ANATOMY BLOCK DIAGRAM | 11-82 |
| 3.5-6 | MAGNETIC FIELD SE OF ALUMINUM | 11-82 |
| 3.5-7 | LOSS OF SE WITH JOINT FINISH | 11-83 |
| 3.5-8 | SE COMPARISON | 11-83 |
| 3.5-9 | WIRE SE COMPARISON AT HF-VHF | 11-85 |
| 3.5-10 | MATERIAL/CONFIGURATION SE OVERVIEW | 11-85 |
| 3.5-11 | POWER RETURNS | 11-86 |
| 3.5-12 | MATERIAL/RESISTANCE OVERVIEW | 11-87 |
| 3.5-13 | CONFIGURATION/RESISTANCE OVERVIEW | 11-88 |
| 3.5-14 | RESISTANCE PLOT - COPPER PLANE | 11-89 |
| 3.5-15 | RESISTANCE VERSUS PRESSURE | 11-89 |
| 3.5-16 | JOINT FINISH RESISTANCE | 11-90 |
| 3.5-17 | AIRCRAFT AND STRAP RESONANCE | 11-91 |
| 3.6-1 | TYPICAL NONMETALLIC APPLICATIONS | 11-91 |
| 3.6-2 | GRAPHITE-EPOXY RESISTANCE | 11-93 |
| 3.6-3 | GRAPHITE-EPOXY SE SHORTFALL | 11-93 |
| 3.6-4 | GRAPHITE-EPOXY DETAILS AND INSTALLATION | 11-94 |
| 4.1-1 | HF-VHF RANGE | 11-96 |
| 4.1-2 | SELECTED RF FIELDS: HF-VHF RANGE | 11-96 |
| 4.1-3 | EXPECTED WIRE VOLTAGE - EIGHT FEET | 11-97 |
| 4.1-4 | ENVIRONMENTS, SAFETY, AND TEST | 11-97 |
| 4.1-5 | TRANSISTOR THRESHOLD | 11-98 |
| 4.1-6 | REPRESENTATIVE COAX VOLTAGE COUPLING | 11-98 |
| 4.1-7 | REPRESENTATIVE SECONDARY COUPLING | 11-99 |
| 4.1-8 | GRAPHITE-EPOXY OR AL SHIELDING ADDED | 11-99 |
| 4.1-9 | "H" FIELD COUPLING | 11-102 |
| 4.1-10 | "E" FIELD COUPLING | 11-102 |
| 4.1-11 | COUPLING VERSUS SEPARATION | 11-103 |
| 4.1-12 | MAGNETIC FIELDS | 11-104 |
| 4.1-13 | TRANSIENT RINGING | 11-105 |
| 4.1-14 | ELECTRICAL TRANSIENTS | 11-106 |
| 4.2-1 | LAYERED DESIGN | 11-108 |
| 4.3-1 | TRADE-EQUIPMENT LOCATION | 11-112 |

| Figure | | Page |
|---|---|---|
| 4.3-2 | TRADE-WIRE LENGTH/SEPARATION | 11-113 |
| 4.3-3 | TRADE-DIGITAL CIRCUITS | 11-114 |
| A-1 | EMI TEST LIMITS (4 sheets) | 11-137 |


## LIST OF TABLES


| Table | | Page |
|---|---|---|
| 2.1-1 | NAVIGATION EQUIPMENT DOCUMENTATION | 11-25 |
| 2.2-1 | 1990's ATC IMPLEMENTATION | 11-40 |
| 2.3-1 | CATEGORIES OF CRITICALITY | 11-42 |
| 2.6-1 | VARIANCES IN EMC | 11-54 |
| 2.6-2 | AIR FORCE VARIANCES | 11-55 |
| 7.2-1 | SUBSYSTEM VERSUS AIRCRAFT TEST | 11-128 |
| A-1 | DO-160B AND MIL STD 461B. TEST CROSS REFERENCE | 11-135 |

The Aircraft Electromagnetic Compatibility guidelines document deals with electromagnetic compatibility in a commercial transport aircraft including the specifications, the activities, the design, and the tests to verify and validate compatibility.

Objectives are to view architecture, equipment and wiring location, material properties, circuit susceptibilities, and environment as seen from the electromagnetic compatibility design perspective of balanced circuits, filters, electrical bonding, grounding, and shielding.

Even today, digital electronics are much more common in aircraft. Automated flight controls of future aircraft will operate under the control of digital clocks, data buses, switching regulators, pulse width modulated power, and radio frequency transmitters on the one hand, and on the other, sensitive analog and digital instrumentation.

Safe and efficient flight will depend on the performance of electronics. It will be important to understand the electromagnetic interference types and the electromagnetic interference paths (figure E-1).

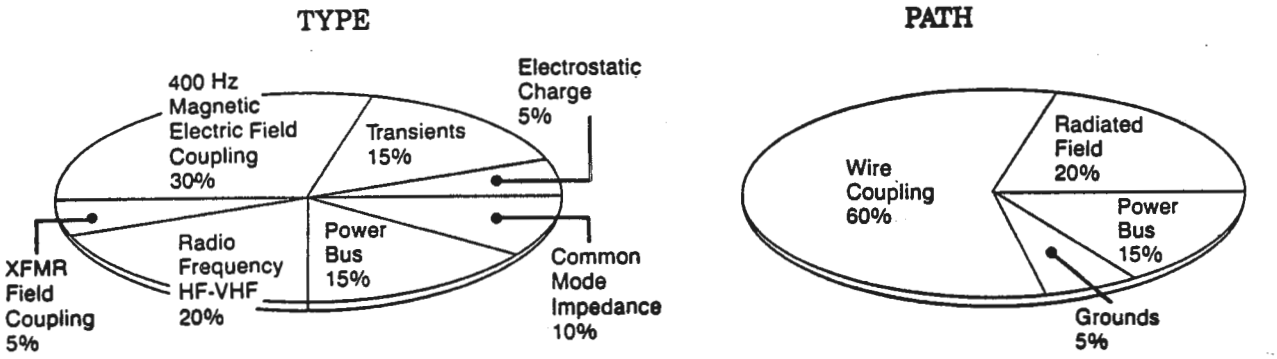



FIGURE E-1. POSTULATED PERCENTAGES

The Electromagnetic Interference (EMI) types are set forth in this document by showing a profile of the magnetic and electric fields from power lines; some military, urban, and rural radio frequency field strengths; and the properties of transients. Significance of the wire circuit return, the balanced circuit, grounding, shielding, and software highlight the protection techniques of a layered design which will block paths of electromagnetic interference and maintain interface signal quality. Design specifications, activities, and reviews are proposed to help set up guidelines for equipment verification and aircraft validation.

# 1. INTRODUCTION

## 1.1. Background and Scope

### 1.1.1. A Case of Engine Shutdown

Captain Hoag thought briefly of the moment when he left home last night. His son, daughter, and wife were all there. They had joked about their planned upcoming vacation, their first together in two years.

A voice broke into his short reverie: "Flight 211, you're low and to the left - please maintain 023 - you have a cell at 2 o'clock."

Hoag said to the first officer, John Pearson: "John, push it forward and bring 'er up."

John said: "Gotcha covered. I flew one of these new ones two weeks ago into Loridan International - they sure do handle smoothly."

"Okay - uh huh." "Gear down."

Flight 211, Atlantic Air, was on approach to Keithrode International Airport (KIA). It was 16:45 on October 16. Two hundred and twenty-seven passengers were on board. The weather had been partly cloudy with thunderstorms predicted and cell activity in the proximity of the air terminal.

Tower: "Flight 211, you are cleared on Runway 3. You are still low."

Captain Hoag: "John, bring 'er up."

A lightning flash occurred off to the left. Then, instantly, a blinding flash, an overwhelming shudder - the aircraft metal structure and body seemed to vibrate under a massive pressure and energy wave.

Hoag: "We've lost number 1! Push it all the way forward!"

Pearson: "Okay"

Hoag: "All the way forward - all the way forward - oh."

The plane hit the earth with a great screech, scrape, a shower of sparks, and a grinding of metal. It then rose again, lumbering and awry, as if struggling to be airborne - struggling - then it smashed again to the ground. The tail buckled. Flames broke out.

Fifty-six people were killed - including the crew. A number of people were injured.

HYPOTHESIS NO. 1: The piercing lightning strike to the left-hand engine caused a large atmospheric pressure wave. This wave traveled through the engine intake into the main engine chamber, snuffing out the flame, and thus the engine power.

HYPOTHESIS NO. 2: The lightning strike to the engine established a large electrical current flow in the engine structure and cowling. Electrical circuits connected to the structure experienced voltage transients causing valve malfunction and leading to an engine shutdown.

This case is dramatic. It is awesome. It commands attention. We recognize that we must protect against this type of event, model it, and develop reiterative computations and tests to uncover the boundaries of transient energies invading important electrical circuits.

This case is given to illustrate the contrast between a very visible and threatening electrical upset or damage phenomenon and the usual run of invisible Electromagnetic Interference (EMI) unfamiliar to most airline passengers. Normal electromagnetic interference environmental problems ordinarily have not carried with them the drama of the above case. Aircraft have been constructed with controls and electrical apparatus having EMI problems but not having any influence on safety.

Knowledge of and protection against induced noise voltages are necessary today. They will be even more necessary in future aircraft where vital and critical control functions are being taken over by avionics interconnected by digital data buses. These data buses that could impair safety if beset by electrical noise. We need access to knowledge of the various types of noise. There is a growing awareness of electromagnetic interference and Electromagnetic Compatibility (EMC).

### 1.1.2. Electromagnetic Compatibility

Electromagnetic interference could cause a flight delay or endanger the operation of an aircraft at 30,000 feet. Generators of electromagnetic interference for aircraft (figure 1.1-1) take on several forms:

- Transmitters of radio frequencies that may be installed on the aircraft itself, such as High-Frequency (HF) or Very High-Frequency (VHF) communication links, or high-energy sources located on the ground such as our everyday frequency modulated (FM) radio or HF-VHF-UHF broadcast stations.

- The aircraft power line 400-Hz electric and magnetic fields.

- The computer and avionics microprocessor timing and control clock signal circuits that generate radio frequencies of one MHz or higher.

- The aircraft power switching regulators which are used to convert from one level of power to another.

Electrical switching transients sparked by the turn on and off of aircraft lights, fans, and engines or by the operation of control surfaces, ailerons, slats, and flaps.

Electrostatic discharges including lightning.

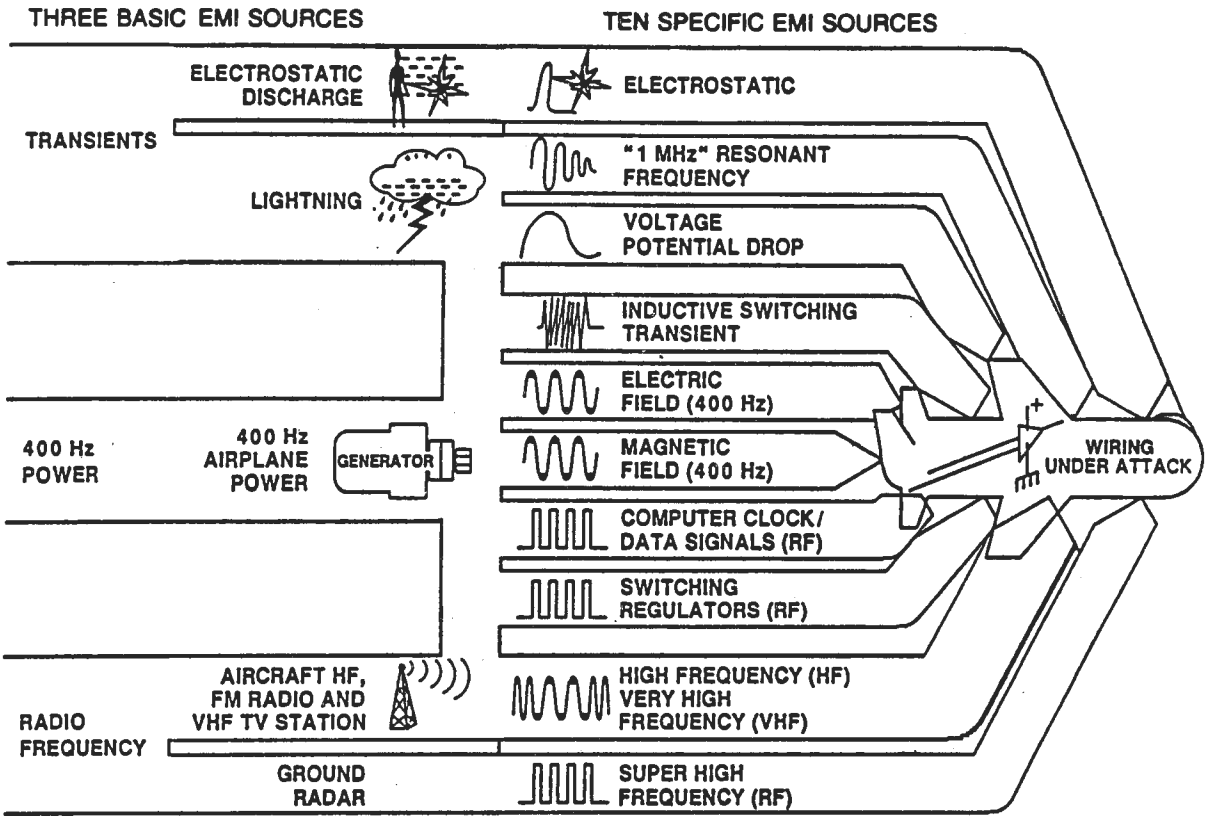



FIGURE 1.1-1. REPRESENTATIVE EMI SOURCES

These transients and electromagnetic waves may transfer into wiring and cause "electromagnetic interference" to microcircuits inside electrical equipment and avionics. This interference could result in a trifling disorder in a flight deck display or, more seriously, an engine shutdown.

The conductive paths of electrical wiring provide an avenue to usher electromagnetic interference directly to airplane avionics and signal inputs. Eliminate wiring, and electromagnetic interference almost vanishes. Wiring is

the most important factor in electromagnetic interference and electromagnetic compatibility. Of much less importance is the electromagnetic interference path through the avionic equipment metal housing or case. Wiring is the electrical interface and connection between avionic equipment. Its designated job is to transfer avionics signals, data, and information. In that function, it can often act to transfer electromagnetic interference energy to other wires in a wire bundle. It also sprays or radiates like a transmitting antenna and very efficiently receives radio frequency energy like a receiving antenna.

Recently, an experienced electromagnetic compatibility engineer, "an old hand," was asked, "What's the number one requirement for an electromagnetic compatibility design program that will rule out electromagnetic interference problems?"

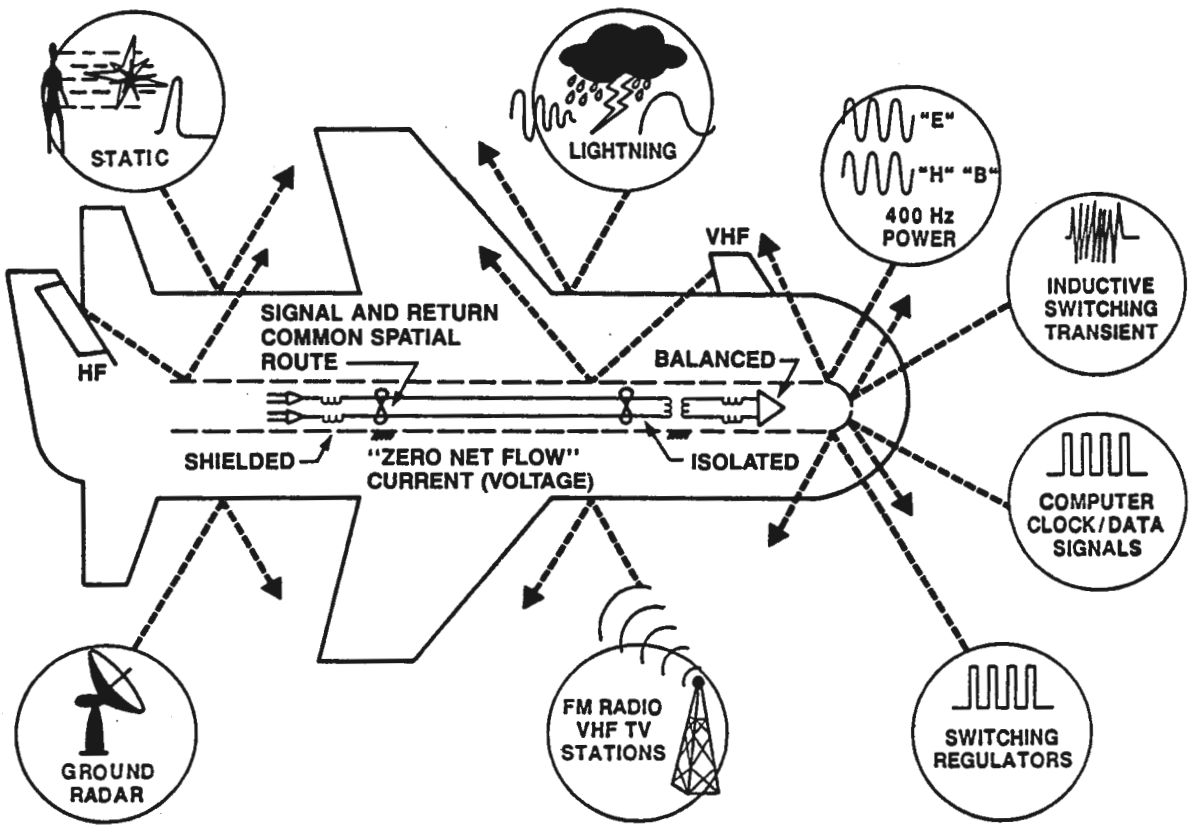



FIGURE 1.1-2. EMI ENVIRONMENT IN AIRCRAFT

His answer, "Zero net current flow in a shielded, balanced, isolated circuit." In other words, always route the signal wire and return or the power wire and its return together using twisted pair, coax, or shielded pair, which means that aircraft basic structure is not used as the return path for the circuit (figure 1.1-2). This will almost eliminate the three electromagnetic interference avenues of entry: common mode impedance paths, magnetic field coupling, and electric field coupling. (Common mode impedance conditions exist when two circuits share a portion of the same electrical path.)

So, that's it: use a return wire and use a shield for each circuit.

Without good shielding, the results are predictable. Removing the shield from the circuit and separating the return from the signal wire not only destroys the efficiency of the circuit, it also opens up the circuit to intervention from stray electric and magnetic fields. These fields provide an avenue directly to the micro-processor memories, computers, and controllers that are needed for aircraft operation and for processing and control of flight deck displays and instruments. A recent estimate of airplane problems (figure 1.1-3) indicates that power line electric and magnetic field coupling (30%), radio frequency fields (20%), transients (15%), and common mode impedance paths (10%) make up a total of 75% of deficiencies. These can be largely corrected with proper wiring design. Electromagnetic interference may occur in many of the aircraft subsystems (figure 1.1-4). If uncontrolled, it appears as radio tones, static, or 400-Hz hum on the passenger entertainment systems. It can show up as flight deck display distortion or illegibility, impaired data transmittals, computer memory loss, and may even result in suspension of equipment operation. Radio frequencies are becoming more of a concern. Today, the predominant radio frequency fields that impair avionic equipment operation fall into the HF-VHF radio frequency spectrum (figure 1.1-5). In future aircraft, the range may vary. (See section 10, Bibliography, ECAC Study.) Electromagnetic compatibility requirements encompass almost every subsystem on an aircraft.

The field of electromagnetic compatibility is not only a discipline in itself, it is also the cornerstone in the proper application of other engineering technologies. Good engineering design techniques employed in the areas of avionics, wiring and cabling, electrical bonding and grounding, lightning protection, and others, go a long way in achieving electromagnetic compatibility. Electromagnetic fields are produced by the generation, transmission, and utilization of electrical energy. Stray electromagnetic energy is generally not desired, and quite often interferes with the operation of electrical/electronic equipment, hence, the name electromagnetic interference. Control specifications for electromagnetic interference generation and electromagnetic interference susceptibility are required to achieve electromagnetic compatibility. With proper wiring, shielding, and application of voltage limiters in a good electromagnetic compatibility design, there is increased confidence in equipment operation which converts directly to the bottom line of on-time dispatch for the airlines and their passengers.

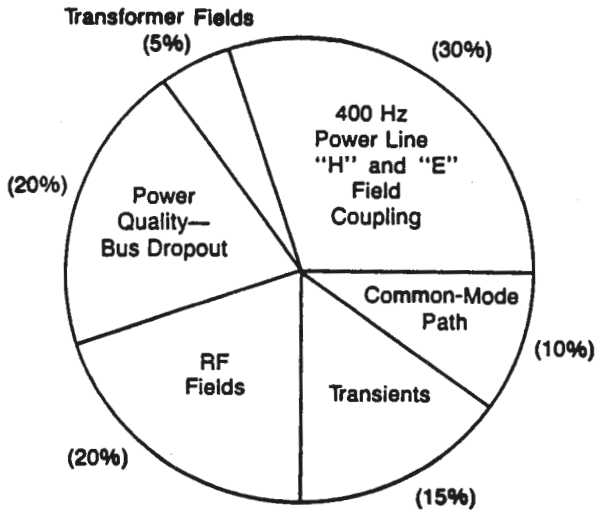



FIGURE 1.1-3. PERCENT TROUBLESHOOTING-A/C

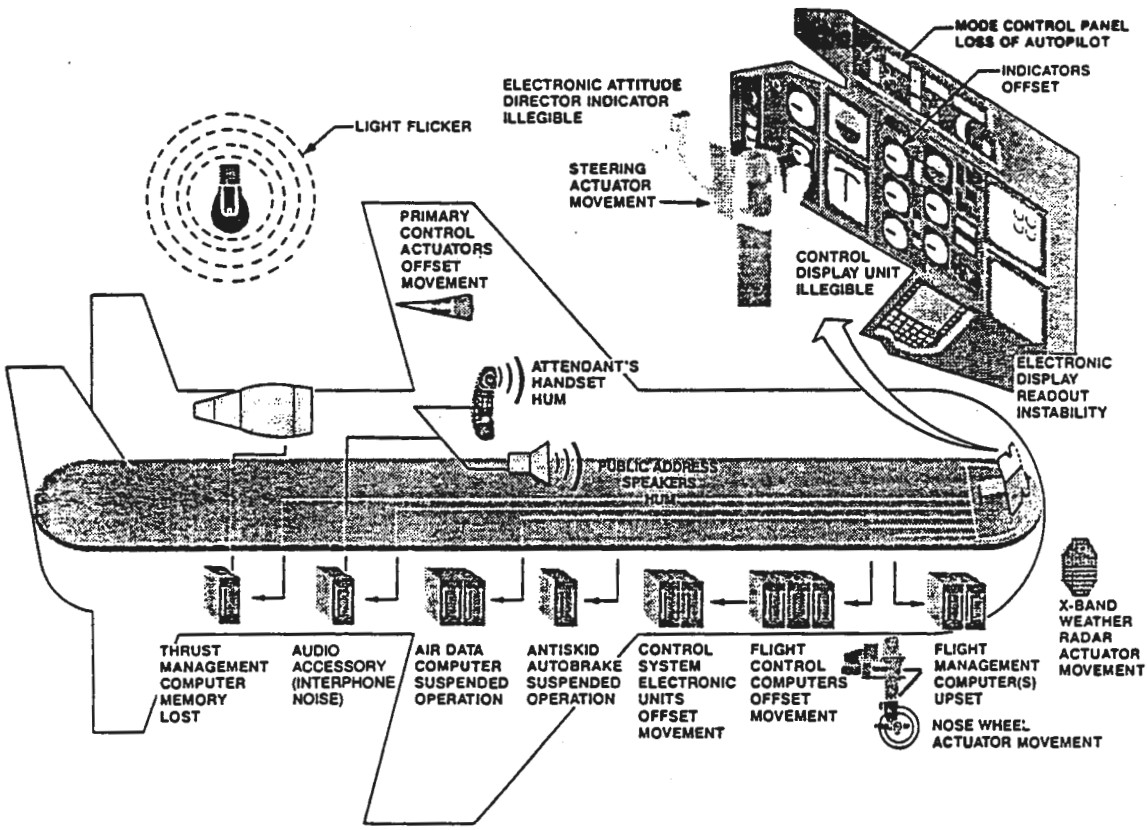



FIGURE 1.1-4. ELECTROMAGNETIC INTERFERENCE EFFECTS

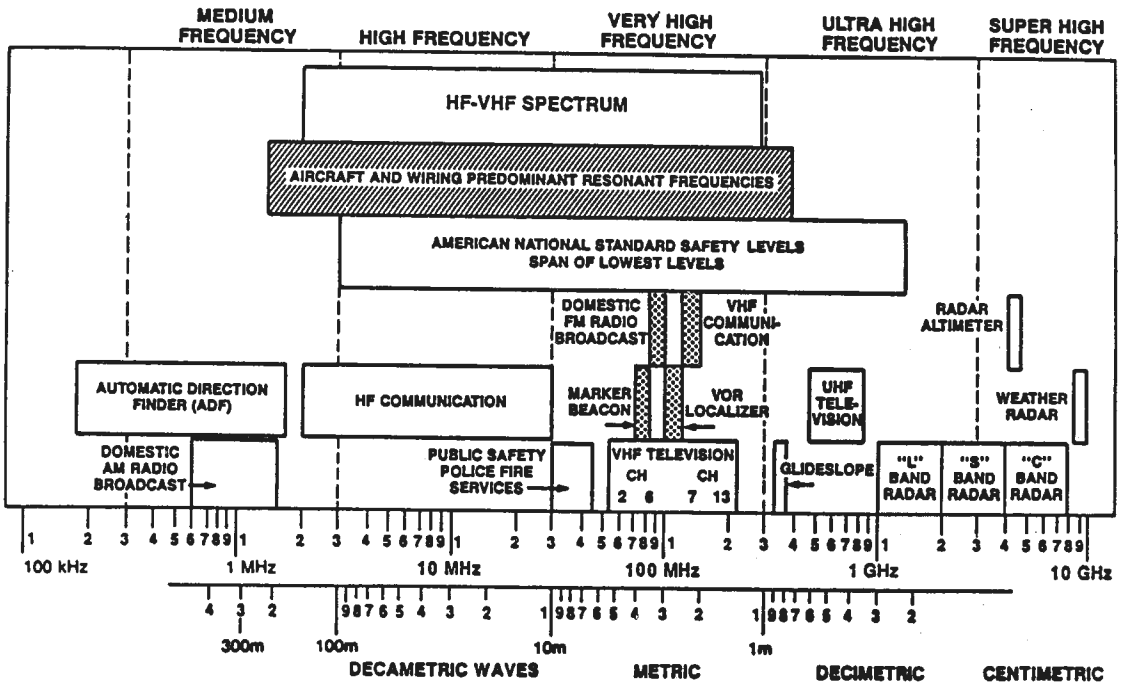



FIGURE 1.1-5. RADIO FREQUENCY RANGE

The steps to reach electromagnetic compatibility in an airline are basically threefold: (1) procure equipment and wiring according to EMC specifications, (2) package the equipment and wiring in the aircraft to obtain protection from structure, and (3) measure and test the equipment and wiring for all aspects of EMI during the program to guarantee verification and validation of EMC (see section 7, Verification and Validation).

EMC specifications for the commercial airplane are covered by Federal Aviation Regulations (FAR) and are specified in terms of system operation. FAR, Part 25.1353, Paragraph A, covers electrical equipment: "Electrical equipment controls and wiring must be installed so that operation of any one unit or system of units will not adversely affect the simultaneous operation of any other electrical unit or system essential to safe operation."

FAR, Part 25.1431, Paragraph C, covers electronic equipment: "Radio and electronic equipment, controls and wiring must be installed so that operation of any one unit or system of units will not adversely affect the simultaneous operation of any other radio or electronic unit or system of units, required by this chapter."

Environmental and emission control specifications for electronic equipment intended for installation in aircraft are documented in the current revision of Radio Technical Commission for Aeronautics (RTCA) Environmental Specification DO-160. (See sections 5 and 9.)

## 1.2. Electromagnetic Compatibility Priorities

### 1.2.1. Responsibilities and Policies

A new airplane presents a challenge of equipment location, packaging, assessment of environment, and resolution of new technology problems. The airframe manufacturer takes on the responsibility in the electromagnetic compatibility design to outline the operational-temporal-spatial anatomy of the airplane early in the conceptual stage. It is a difficult and extensive task. (See sections 4.2 and 4.3.)

Equipment location, wiring, transmitters, receivers, 400-Hz power, lightning diversion, and static dissipation are all familiar items to be pursued, tracked down, identified, and recorded. They absorb many labor hours.

Even with the rapid advance of technology, many components, characteristics, and parameters on a new airplane do not change. The electromagnetic compatibility engineer has a responsibility to know the off-the-shelf equipment. A productive policy is to seek out and rely on existing specifications and designs to the greatest extent possible, thereby sidestepping duplication of effort in analysis, scheduling, and testing. EMI test levels documented in the current issue of the RTCA Document DO-160, in many instances, represent today's environment. (They do not take into account the effects of new composite structures.) The test levels have been developed over the years. Airframe manufacturers and subcontractors can take good advantage of existing specifications and test data.

So, the first priority of the airframe manufacturer is to take on the task of defining the electromagnetic compatibility anatomy - existing equipment, new equipment, new environment, and locations.

Location often sets the electromagnetic compatibility requirements for avionics. Electronics or lack of it influences design requirements for equipment. (See section 6.)

The first priority of the avionics supplier or subcontractor (including in-house suppliers) is to know the electromagnetic interference environment and then identify, design, and protect each power input and signal input to guarantee that the equipment will operate within performance standards in that environment. The designer must shoulder the responsibility of knowing the interface wiring. Often the designer is concerned with operational requirements and must make a special effort to recognize the noise requirements. DO-160 is the basic industry standard. (See section 3.4.2 and section 9.)

The airframe manufacturer deals with new materials and properties, wiring, electrical bonding, shielding effectiveness, and definition of environment.

The avionics supplier deals with input/output circuit protection, internal grounding, circuit card layout, and interface wiring. The supplier must observe the board-level noise thresholds in a real-time, interactive setting to ensure no "state changes" and to ensure adequate containment of electromagnetic

interference at the box level. The equipment engineer designs devices to be tolerant to magnetic fields from 400-Hz power (400-mV induced), 400-Hz electric field (up to 1,000V induced in test), radio frequencies (1V induced), transients from coils and lightning (600V), and electrostatic discharge (10,000V or higher). Future aircraft having critical fly-by-wire systems may require higher levels, especially for radio frequency fields. The designer must also install filters to control and contain emissions from the oscillator and switching regulator clock and harmonic radio frequencies. Interface wiring must be designed, agreed upon, and documented in an interface control drawing, otherwise hardware would require multiple electromagnetic interference designs.

Along with the size of a new program and the extent of its new technology, three pivotal factors influence the level of effort and are sine qua non to success: (1) management support, (2) the electromagnetic compatibility engineer's product experience, and (3) the electromagnetic compatibility experience of other participating engineers on the program. Management support means setting the priority for early resolution of requirements before formal document release of vendor technical specifications and statements of work. Product experience means minimum duplication and maximum productive effort. Experienced participating engineers means satisfactory coverage of subcontractor requirements, and good electrical bonding and wiring practices.

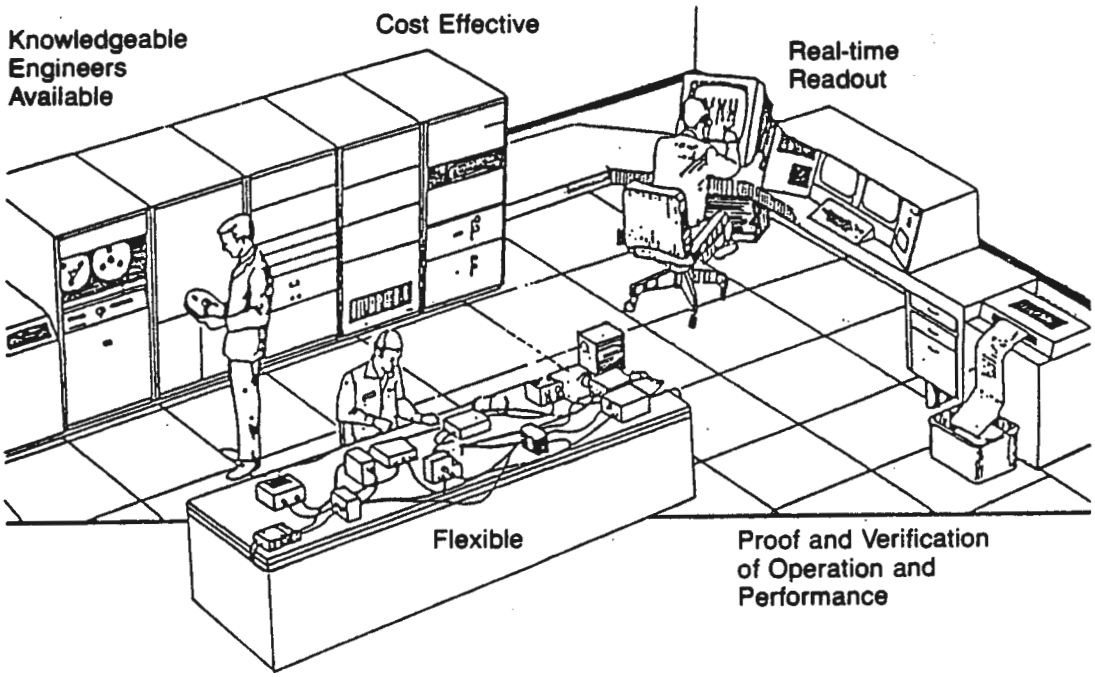



FIGURE 1.2-1. COST-EFFECTIVE SUBSYSTEM TESTING

On a recent airplane development program, extensive subsystem testing was performed on engineering models and production models with these three recognized benefits: (1) proof and verification of equipment performance, (2)

good diagnostic testing, and (3) knowledge of equipment operation. Diagnostic/ trouble-shooting tests can be run in cooperation with the subcontractor. Subsystem testing is becoming a key element in a successful program (figure 1.2-1). (See section 7.)

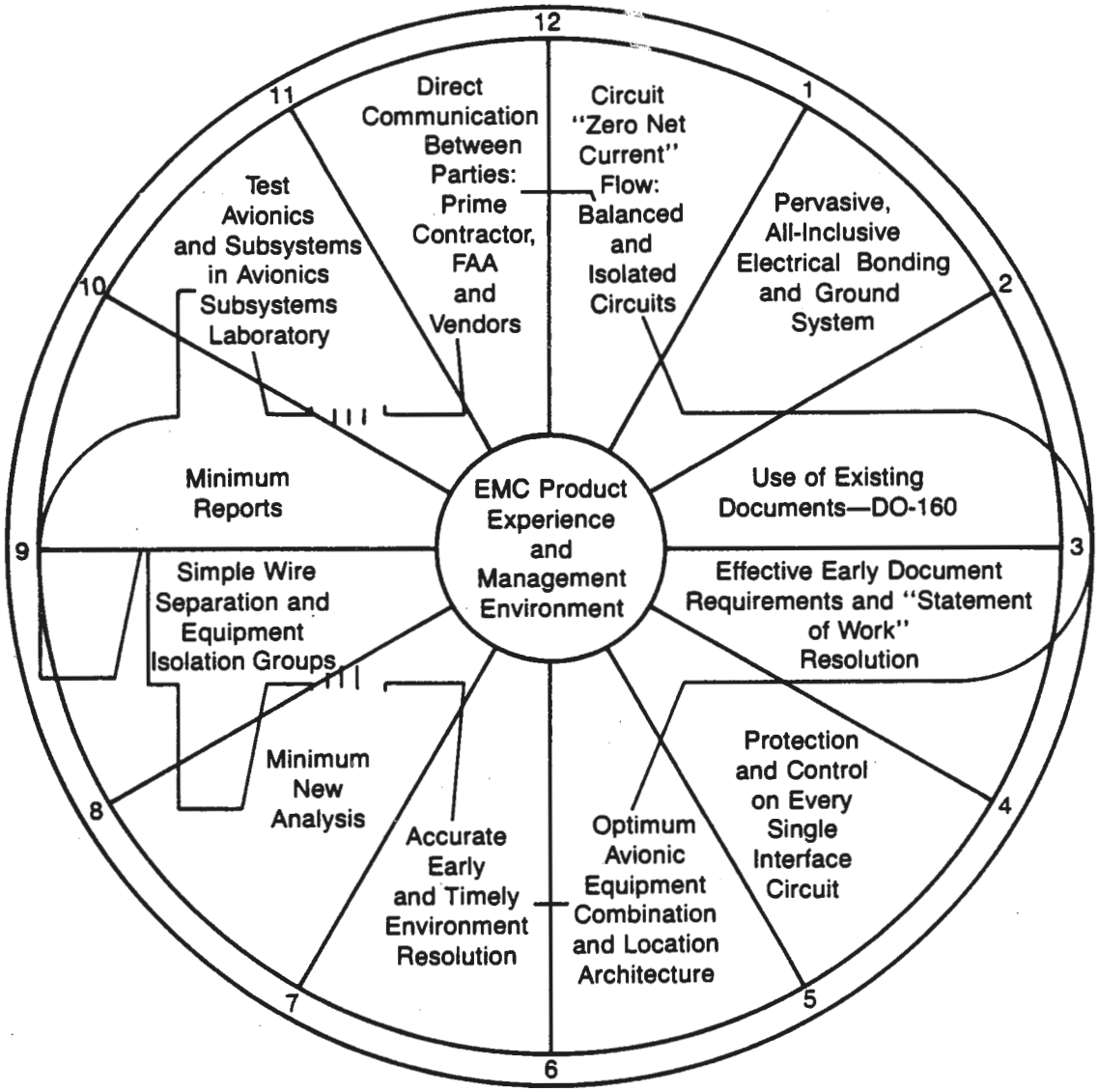



FIGURE 1.2-2. KEY EMC FACILITATORS

Entering into a new program incurs a commitment effort (figure 1.2-2), a commitment sized by the new technology and environment, and a commitment that dissipates with the settling of each subcontractor requirement, each design, each test, and each verification. Early adoption and documentation of known

steps for successful electromagnetic compatibility are established on these baseline technical priorities: (1) zero net current flow in a balanced, isolated, shielded circuit, (2) total, all-inclusive electrical bonding of every structure and every detail (including conductive paint on external dielectric surfaces), and (3) optimum avionic equipment in-line design and location making maximum use of structural shielding.

### 1.2.2. Documentation

To provide a foundation, baseline, reference, and continuity, there are these key documents: system level specification (possibly a paragraph or two), electromagnetic compatibility plan (may be brief), electromagnetic compatibility requirements (extensive and detailed), procurement specifications (with statements of work), interface identification, equipment test (procedure and report), and airplane test (procedure and report). (See sections 5 and 6.)

Procurement specifications must be finalized before contract formal approval.

The fact that minimum documentation leads to a more productive program is as fundamental as the fact that there are certain program top documents that must be instituted to define scope and intent and to reference industry specifications. The documentation of many of the tasks, analyses, and tests on a program can be covered by brief individual memorandums or reports. Some programs issue design notes, a practice that is an efficient method of recording and disseminating design requirements, rationale, and information.

## 1.3. How to Use This Document

The purpose of this Aircraft Electromagnetic Compatibility document is to digest, unify, highlight, and give perspective to the substantive aspects of electromagnetic compatibility applied to a commercial transport aircraft. The material in this document is not new. A very important resource to the electromagnetic compatibility engineer is the electromagnetic compatibility knowledge of other individuals associated with a program. Beliefs exist that graphite-epoxy is an insulator; that box-to-box radiation is important (it is the interface wiring that is the key to EMC); that shield tie (pigtail) length is not critical; that every single interface line does not need to be analyzed for protection and emission control; that ground planes are not required; that single-point grounding is always good (it is good for power, but not good for digital circuitry); and that extensive verification procedures can be ignored and are not essential.

The 1970s and 1980s have seen a striking rise in the quality and extent of EMC/EMI design engineering knowledge that has important consequences for EMC. This document attempts to collect and apply that information to the airplane. Derivations and fundamentals of EMC are not elucidated. This document is limited in that sense. Reference can be made to the bibliography for some excellent articles.

EMC information not found herein:

- Fundamentals or basics of EMC.

- Formulas, models, derivations.
- Antenna-to-antenna coupling.
- Power system quality.
- Lightning.

EMC information included:

- Aircraft-applied EMC.
- Architecture, equipment layout.
- Dominant EMI environments.
- Circuit susceptibilities.
- Bonding, grounding, shielding.
- Wiring design.
- Verification, validation.

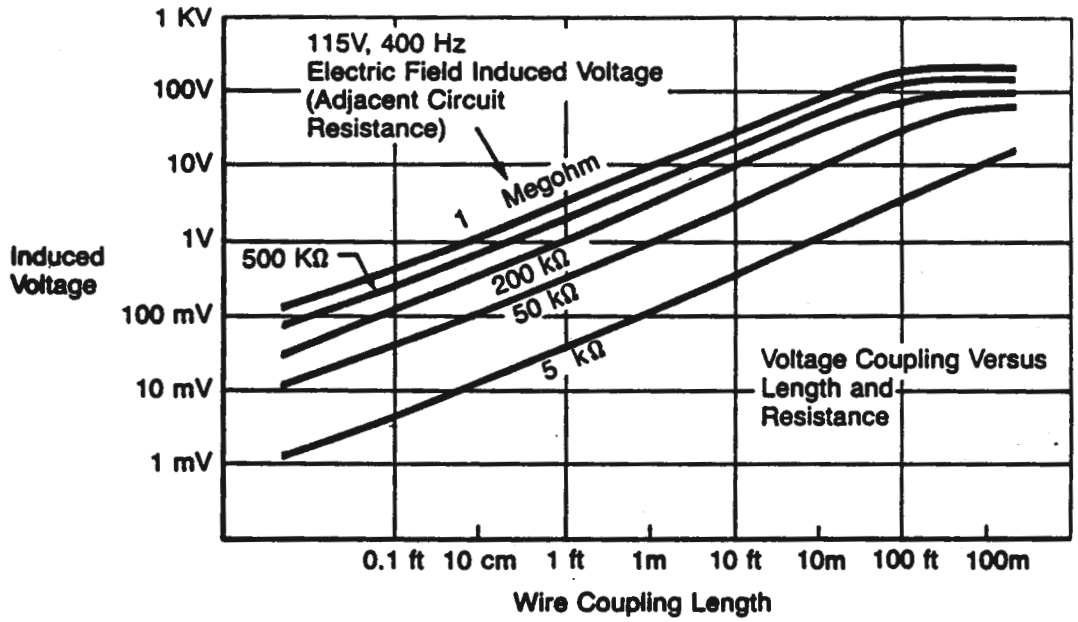



FIGURE 1.3-1. REPRESENTATIVE "E" FIELD COUPLING

It is recommended that this document be read thoroughly before concentrating on specific sections. This document is not a design document or a "design cookbook." In no way can it replace specific analysis and design effort. It is a set of guidelines to outline and scope deficiencies and qualities in present day aircraft that may aid in the approach to future designs. The information and data is illustrative and advisory to provide definition and make comparisons of parametric properties and behavior, and it is not for use or adaptation to specific designs. For example, figure 1.3-1 maps the expected voltages induced in a single aircraft wire circuit (having resistive loads) from an adjacent 115V, 400-Hz power wire where both have their returns in aircraft structure. This figure illustrates the significance of length of coupling and resistive loads and gives a rough estimate of amplitudes. But there are other

interacting parameters that might be considered; ergo, each circuit type in the aircraft must be evaluated separately.

Much emphasis on wiring design exists in this document. The soul of EMC is a balanced, isolated, shielded interface circuit. It closes the door on EMI. It rejects transients, radio frequencies, and 400-Hz fields. It is practically impervious to conductive, inductive, and capacitive transfer of energy.

Linearity rises as one of the elegant attributes of electromagnetics, giving simplicity to variations in the electromagnetic dependent and independent parameters: length, height, resistance, voltage, time, over much of their range. These parameters often vary on a 1:1 ratio (20 dB per decade) or an exponential ratio, possibly 40 dB per decade. The linear relationship breaks down or changes at corner frequencies, 3 dB points, and resonant nodes where the dominance of electrical parameters make a transition from one to the other.

Ratios and the decibel relationship will be used throughout. The decibel, abbreviated dB, is a unit expressing the ratio between two amounts of power, $P_1$ and $P_2$, existing at two points. By definition, the number of dB equals 10 log to the base 10 ($P_1$ is divided by $P_2$). For special cases where $P_2$ equals 1 mW or 1W, the dB ratio is defined as "dBm" or "dBW." For power, a factor of 10 equals 10 dB. Since power P equals $V^2$ divided by R, or $I^2$ times R, decibels can be used to express voltage and current ratios where the voltages and currents are measured at places having identical impedances. By definition, dB equals 20 log of ($V_1$ divided by $V_2$), and dB equals 20 log ($I_1$ divided by $I_2$). For convenience, $V_2$ or $I_2$ are often chosen as 1 $\mu$V and 1 $\mu$A, and the dB ratio defined as dB above a microvolt or dB above a microamp. Also for convenience, these ratios are more often used whether or not they are referenced to identical impedances. A factor of 10 equals 20 dB. Memorize these voltage-current ratios: 6 dB = 2X 10 dB = 3X, 12 dB = 4, 20 dB = 10, 40 dB = 100, and 60 dB = 1000.

# 2. AIRPLANE AVIONICS AND CRITICALITIES

## 2.1. Existing Systems

The well-known hallmarks of a commercial transport aircraft (figure 2.1-1) are: one-hundred and fifty paying passengers or more, flight attendants, airline competition, scheduled dispatch, fixed cost, aisles, lavatories, galleys, and video entertainment.

The necessary control of capital cost and running expenses, the need for quick and easy equipment maintenance, and the desire for on-time dispatch: these goals urge the airframe manufacturer to focus on a well-designed, well-planned electrical architecture, including electromagnetic compatibility. Standardized avionic Line Replaceable Units (LRU) are motivation for low cost and competition among subcontractors (figure 2.1-2). Cost, airworthiness, and safety are the critical drivers for avionics.

Most passengers are unaware of the safety built into the interface wiring and electrical/electronic systems (figure 2.1-3). Passenger and crew safety, with regard to protection from hazards of high voltages, has been guaranteed historically by the ubiquitous aluminum housings, spars, supports, and structure. The high-quality structural aluminum grounding paths are inherently the electrical return or an electrical reference for digital signals, shield ties, motor power current, and fault currents (figure 2.1-4). This structure bypasses the need for separately installed wires or buses to fulfill those functions (figure 2.1-5). Also enhancing the electrical sinking and conducting properties of structure are the extensive air-conditioning, water, and hydraulic systems that form a skeleton of metallic and composite materials throughout the flight deck, cabin, cargo bay, wheel wells, and wing leading and trailing edges. Many of these shielding and sinking properties today help to contain or divert electromagnetic interference from electronic game signals, electrostatic discharge, lightning transients, and high-energy broadcast radio frequencies (figures 2.1-6 and 2.1-7).

Aluminum alloys form the wing, fuselage, and empennage structure, but these metal alloys must mate with materials like fiberglass, Kevlar, and moderately conductive materials like graphite-epoxy. The interfaces at fairings, doors, ducting, and fasteners open up possibilities of apertures and gaps. These mating surfaces must be electrically bonded so that power currents, fault currents, electrostatic charge, and lightning currents flow (figure 2.1-8).

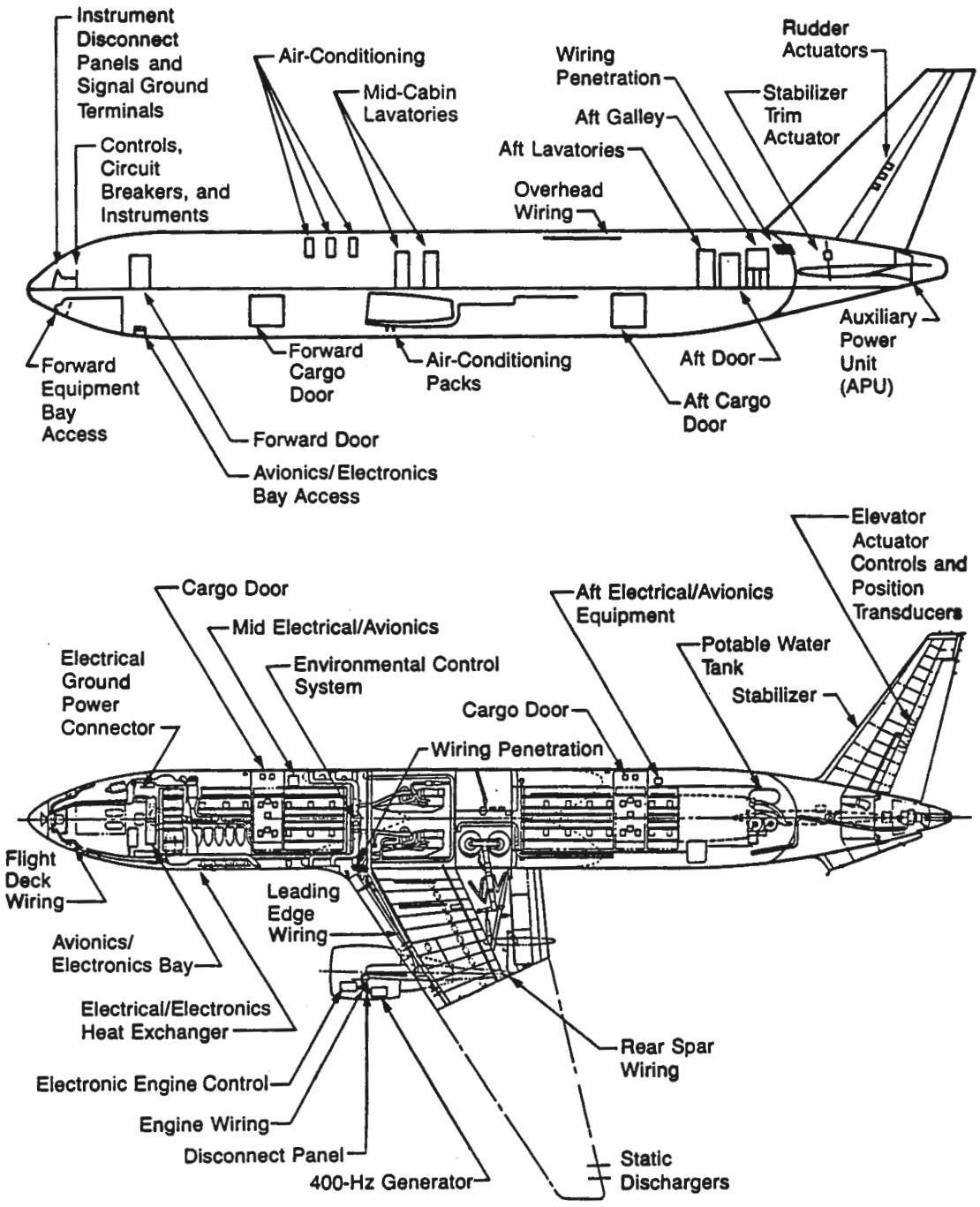



FIGURE 2.1-1. PRESENT-DAY AIRCRAFT SYSTEMS

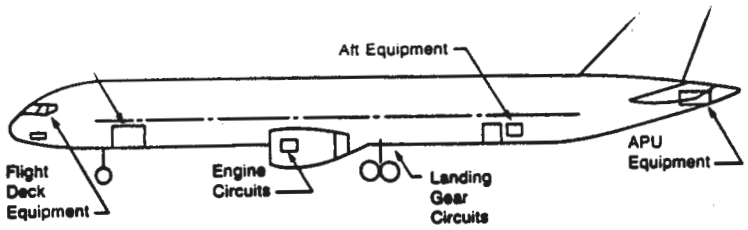



FIGURE 2.1-2. AVIONICS BAYS

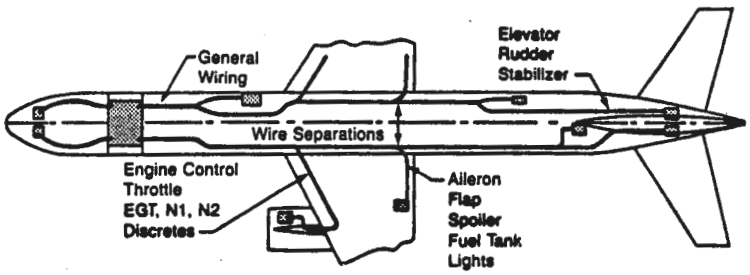



FIGURE 2.1-3. WIRING

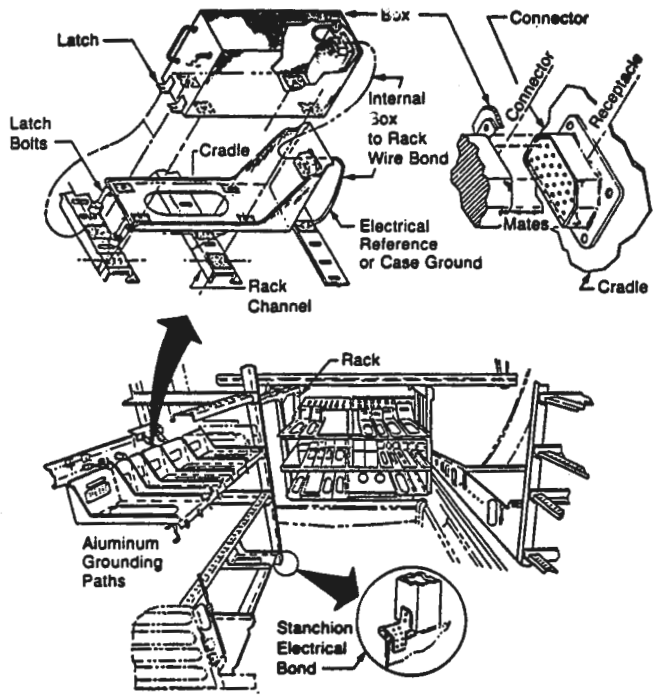



FIGURE 2.1-4. MAIN BAY ALUMINUM STRUCTURE

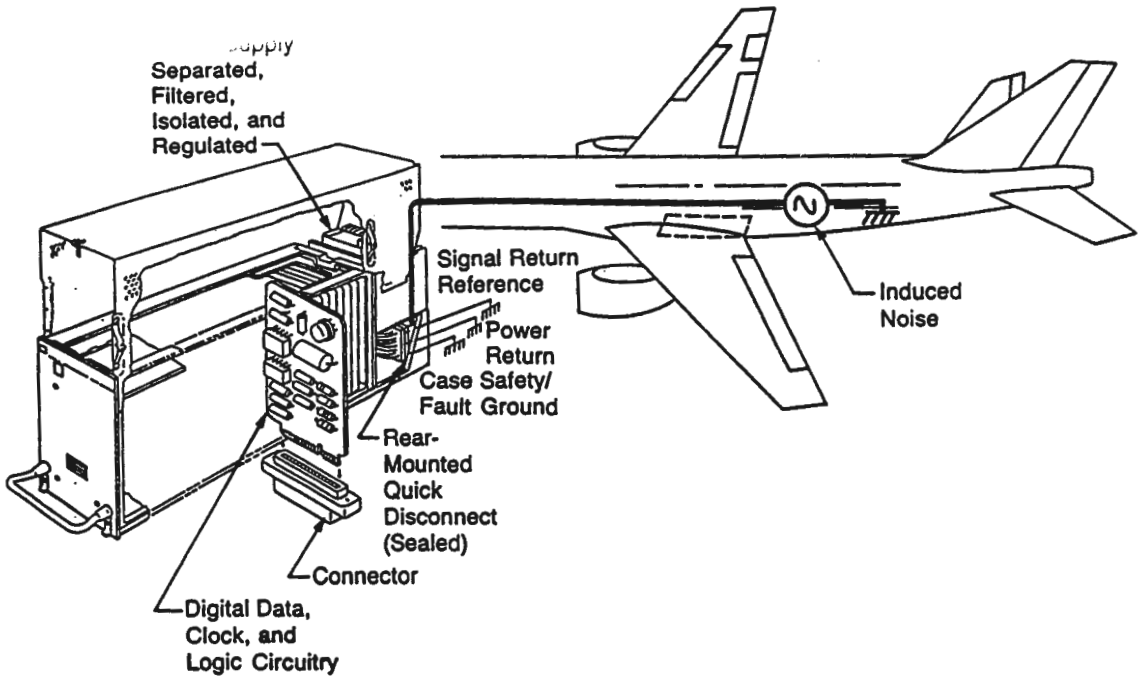



FIGURE 2.1-5. STRUCTURE RETURN

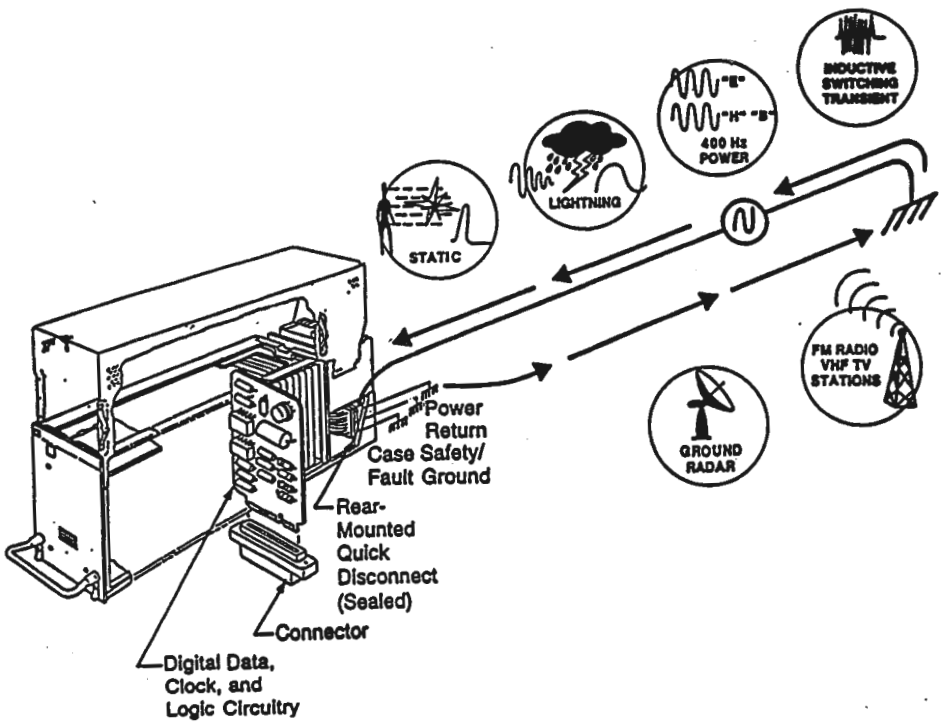



FIGURE 2.1-6. INDUCED NOISE

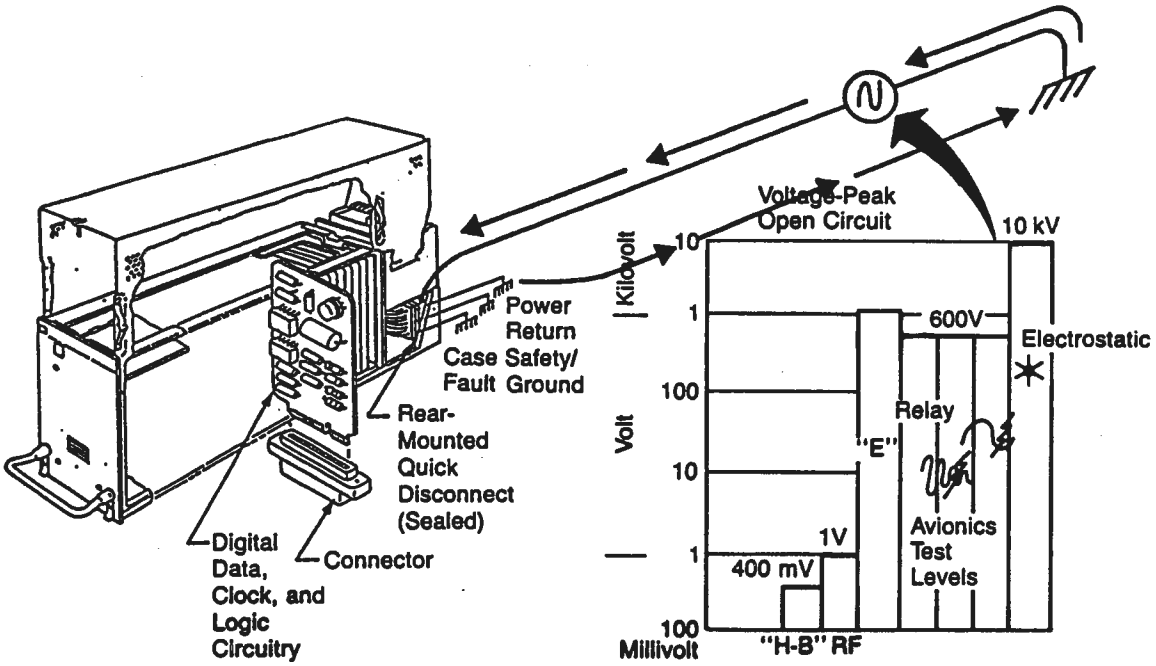



FIGURE 2.1-7. NOISE VOLTAGES

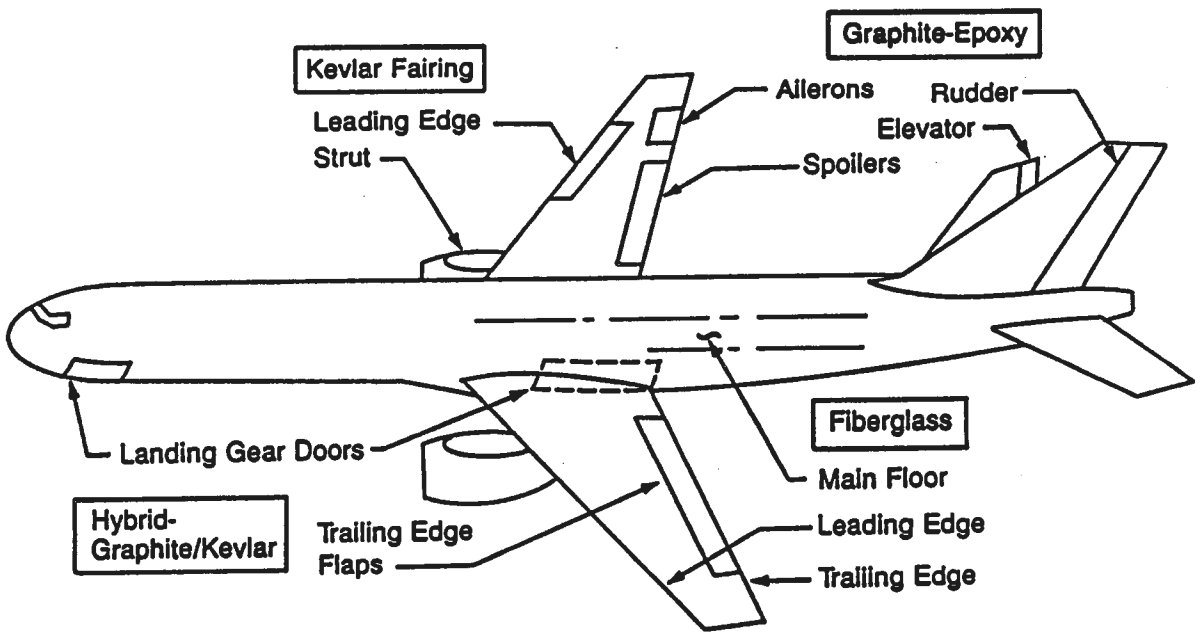



FIGURE 2.1-8. NONMETALLIC INTERFACES

With each new program, we build on the safety and electromagnetic compatibility practices of past experience, and as the program progresses, we purchase and install units of avionic equipment which were designed, built, and tested years ago - existing equipment, "off-the-shelf" equipment. So, years of in-service experience with existing equipment are brought together with the new technology designs on the new program. We cannot overlook the fact that electromagnetic compatibility and safety standards have been well established. Hence, this document does not hold any revealing secrets or creative innovations. Many of the requirements for electromagnetic compatibility are very well known having been spawned, tried, and steadily refined over the years.

With an experienced eye on airworthiness and reliability, traditional electrical and electronic functions have been carefully shaped by the airframe manufacturer and subcontractors to provide and enhance engine instrumentation (safety), communication and navigational aids (safety and convenience), autopilot equipment (pilot workload), and aircraft utilities (safety and passenger comfort). Redundancy of avionics functions and components has become a de facto standard. Avionics equipment in today's aircraft (1986) strongly contribute to the desired levels of performance and safe flight through separation and duality (figure 2.1-9). Any one single part or unit in a fully operational and functioning aircraft is not vital and crucial to continued flight. Triple redundant flight control computers and data buses, for instance, practically guarantee continuous operation and provide a confident reliance on the automatic pilot system (figure 2.1-10). Electromagnetic compatibility is important and must be "designed in" to be cost effective or the avionic equipment will not work, but electromagnetic compatibility has not been critical to aircraft safety.

Engine monitors or sensors are becoming critical, especially on the new electronic engine controls (figure 2.1-11). Left engine circuits are separated from the right for safety. Selected circuits on each engine are separated from each other. Oil pressure transducers, engine temperature thermocouples, exhaust gas temperatures (EGT), and speed sensors (N1, N2) provide information on performance and operational boundaries and status. Knowledge of the status of hydraulics, oil, and fuel systems helps avert critical situations. Throttle lever angle position must be known, and of course, fire detection is mandatory. Electromagnetic interference must be designed out of these circuits early in any program.

Communication receivers and transmitters are the key in today's flight schedules and goals of smooth flight and fuel economy (figure 2.1-12). Noise must be at a minimum to keep from degrading receiver thresholds. Broadband noise will reduce communication range. Narrow band noise will induce unwanted tones. The flight crew does not want to hear static, 400-Hz hum, or popping, even though this electromagnetic compatibility problem may only be a nuisance.

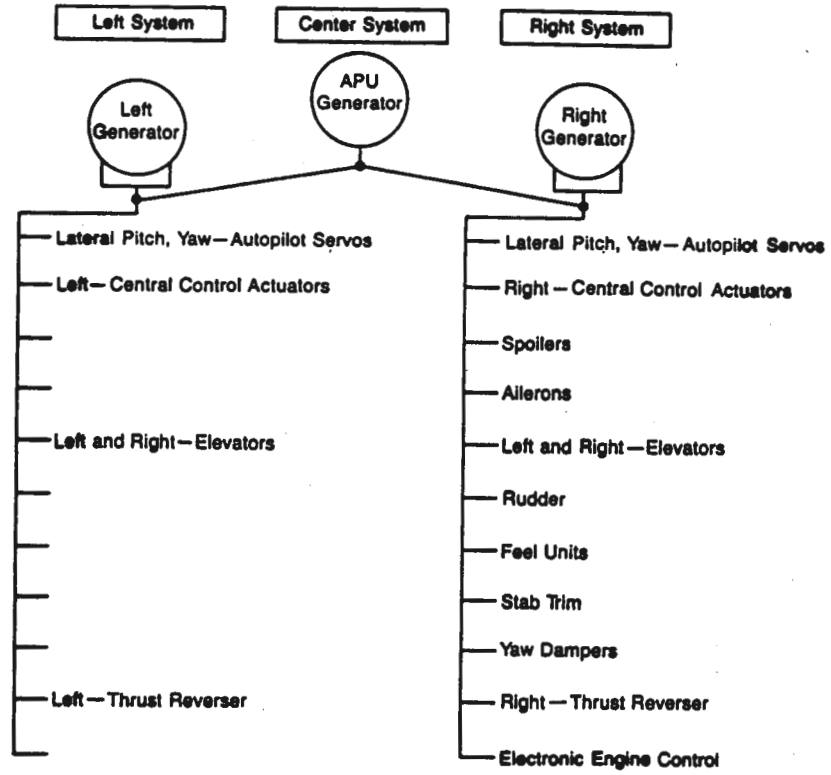



FIGURE 2.1-9. SEPARATION EXAMPLE

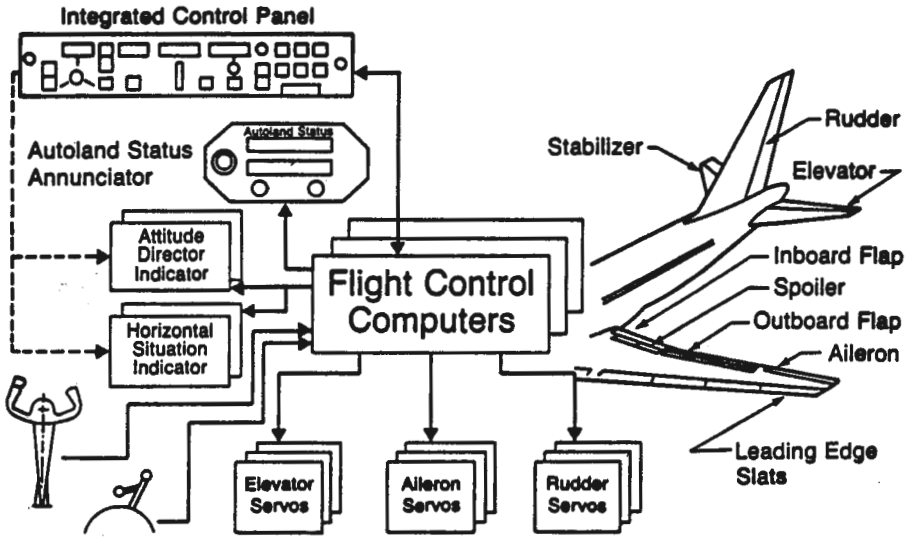



FIGURE 2.1-10. FLIGHT CONTROL AUTOPILOT REDUNDANCY

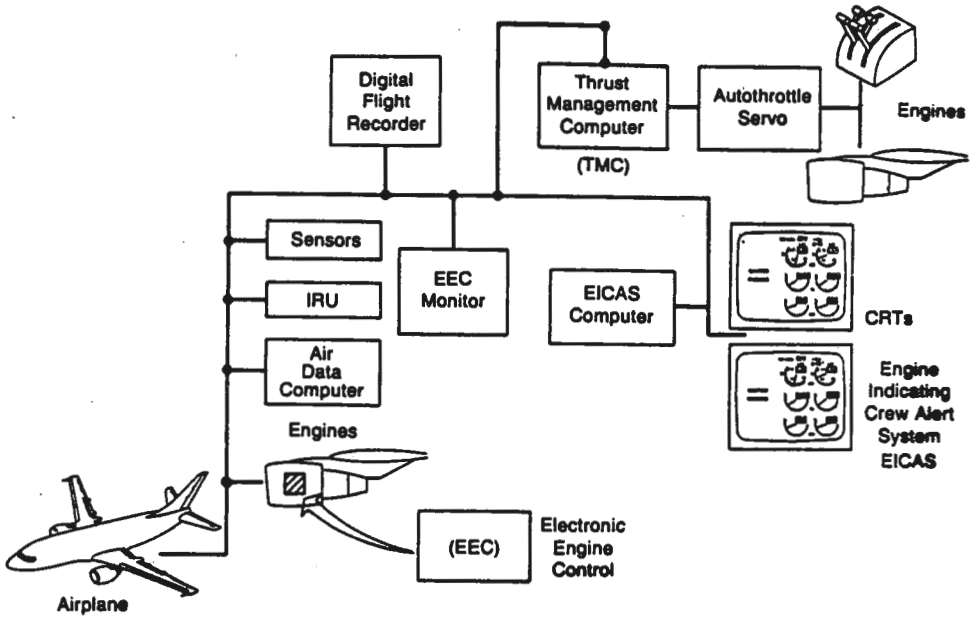



FIGURE 2.1-11. ELECTRONIC ENGINE CONTROL

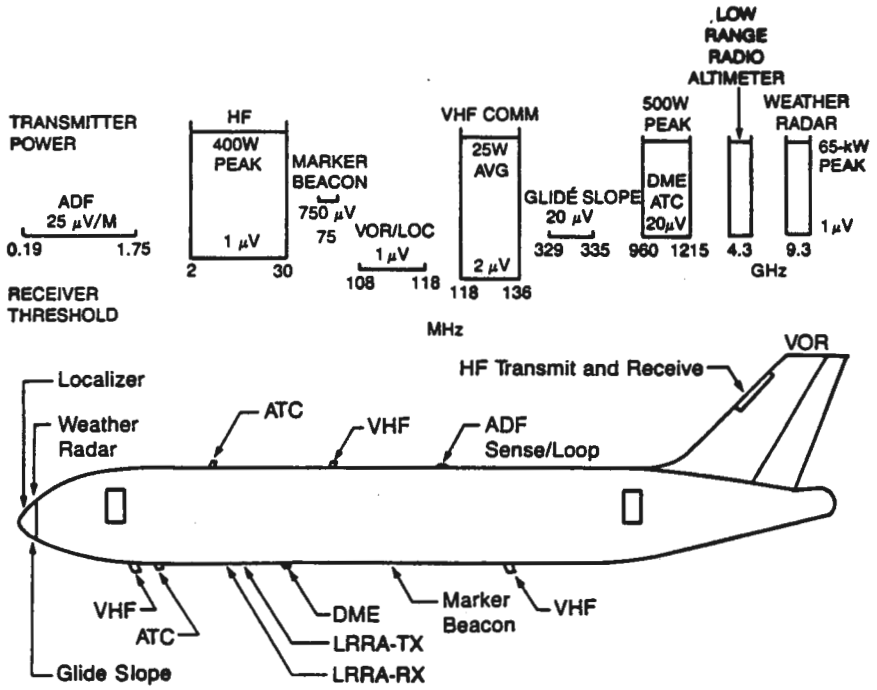



FIGURE 2.1-12. TRANSMITTERS-RECEIVERS

Automatic controls reduce pilot workload and assist in long flights and optimum flight profiles (figure 2.1-10). We still see the old-fashioned compass installed in the instrument panel on the flight deck, but the "horizontal situation indicator" is now a Cathode Ray Tube (CRT) (which, by the way, is very susceptible to 400-Hz magnetic fields) and is designated as the electronic horizontal situation indicator (EHSI). There are two EHSIs to rely on besides the old-fashioned backup compass (figure 2.1-13). Future aircraft will see flat panel displays that must be protected from radio frequencies and transients that induce "snow," "banding," "lines," or "ripple" on the screen.

Today's professional pilot "flies" the plane with the steering column having steel cable strung from the column to hydraulic actuators which then amplify the force and operate control surfaces (figure 2.1-14). The engine throttle is operated by a steel cable. A steel cable does not recognize electromagnetic interference.

Navigation equipment with ground systems that locate and track have grown more and more sophisticated, accurate, and reliable (figure 2.1-15). Specifications and requirements have been developed over the years (table 2.1-1). VHF and HF communications are used over land and sea, all along the airways, and in high-volume areas (figure 2.1-16). Communication avionic units are stand-alone components that can be individually replaced in the electronic equipment bay or on the flight deck (figure 2.1-16). Because of the excellent flexibility of unit replacement, ease of maintenance and substantially lower unit costs, there has been little or no historical incentive for large-scale integration of functions on present day airplanes (figure 2.1-17). Most equipment today is basically digital, but some is still analog (figure 2.1-18). Future aircraft will probably see almost total use of digital circuitry.

Aircraft utilities, of course, are basic to commercial transport operation. Fluorescent lights in the cabin are noisy. Pressurization controls, cooling fans, and window heaters contribute to aircraft magnetic fields. Cargo handling and windshield wiper motors, fans, and galley heaters inflict broadband noise on digital and analog circuits. Future systems may have pulse width modulated voltages that will contribute to noise.

Today's interface circuits span the systems from engine instrumentation to navigation to flight control to utilities, but electromagnetic compatibility engineers do not deal with flight systems. They deal with circuit types, and basically those types can be counted on two hands. The types can be boiled down to digital data, radio frequency signals, analog signals, discrete state changes, and 400-Hz 115V or 28V-dc power and in the future pulsed dc power.

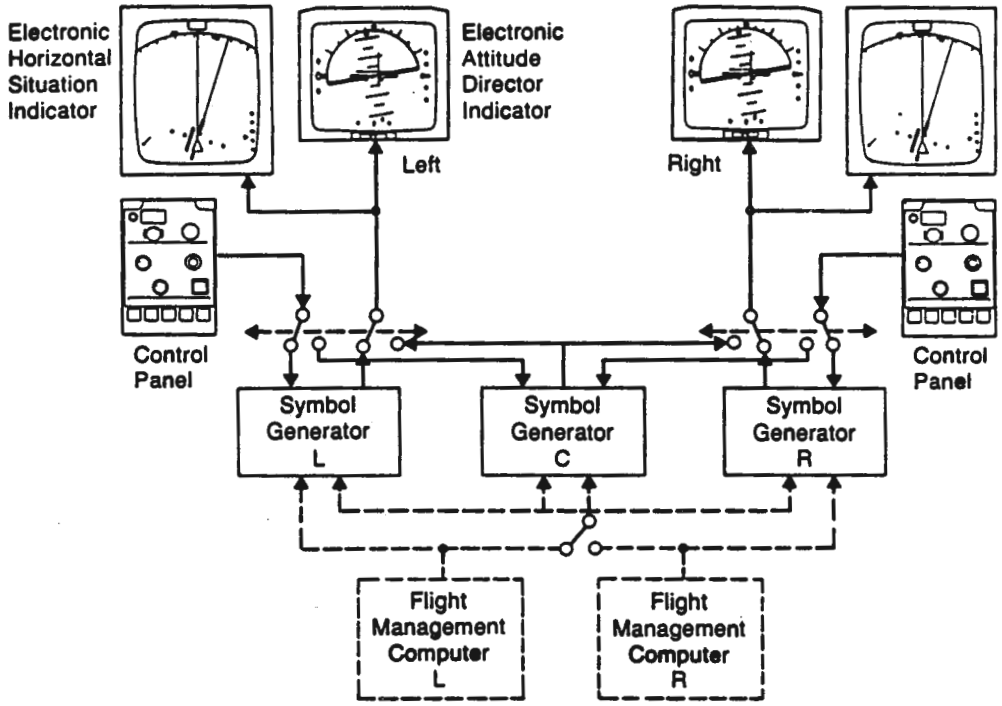



FIGURE 2.1-13. FLIGHT INSTRUMENT SYSTEM

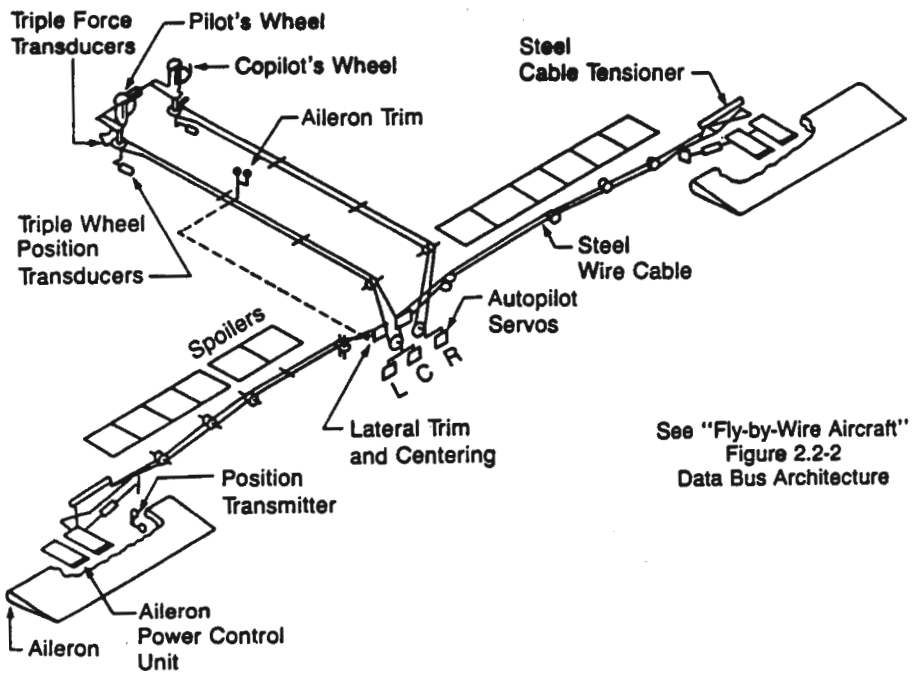



FIGURE 2.1-14. ROLL CONTROL SYSTEM

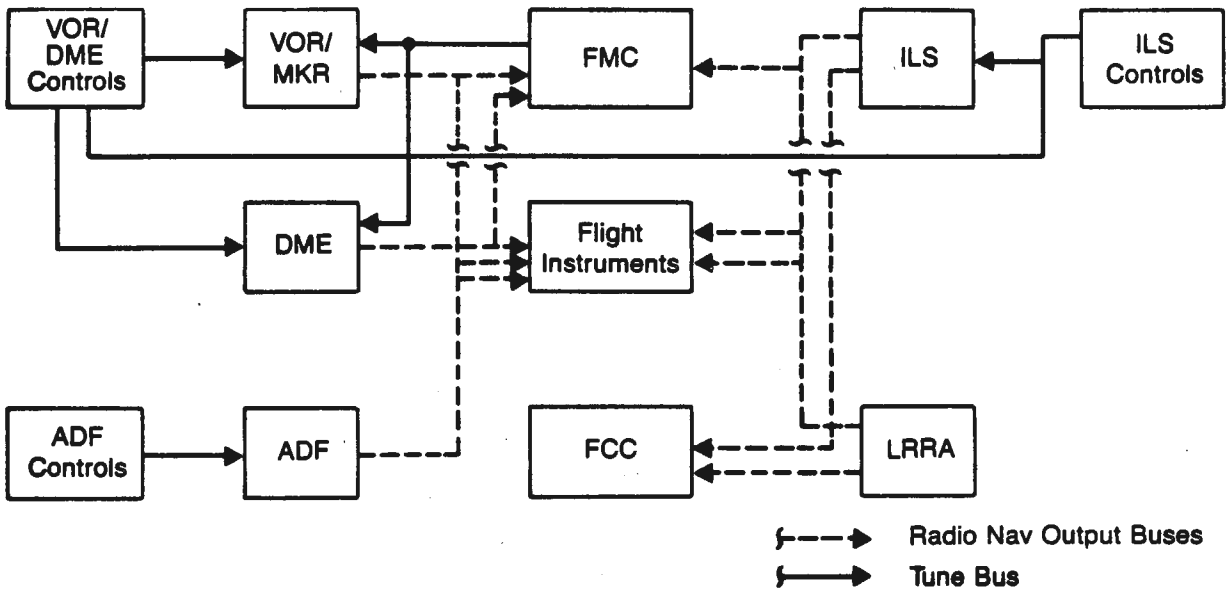



FIGURE 2.1-15. NAVIGATION

TABLE 2.1-1. NAVIGATION EQUIPMENT DOCUMENTATION

| Navigation Equipment | Federal Aviation Regulation (FAR) | Technical Standard Order (TSO) | Advisory Circular | Number Required |
|---|---|---|---|---|
| VOR | 121.349a,e | C40a | 90-45A | 2 |
| DME | 121.349c | C66a | 90-45A | 1 |
| LOC/GS | 121.349a | C34b | 120-28A | 1 |
| | | C36b | 120-29 | — |
| MB | 121.349a | C35c | — | 1 |
| ADF | 121.349b | C41b | 20-63 | 1 |
| INS/ISS | 121.355 | — | 25-4 | 2 |
| | 121, App. G | — | 121-13 | — |
| RNAV | — | — | 90-45A | — |
| Omega | — | — | 120-31 | — |

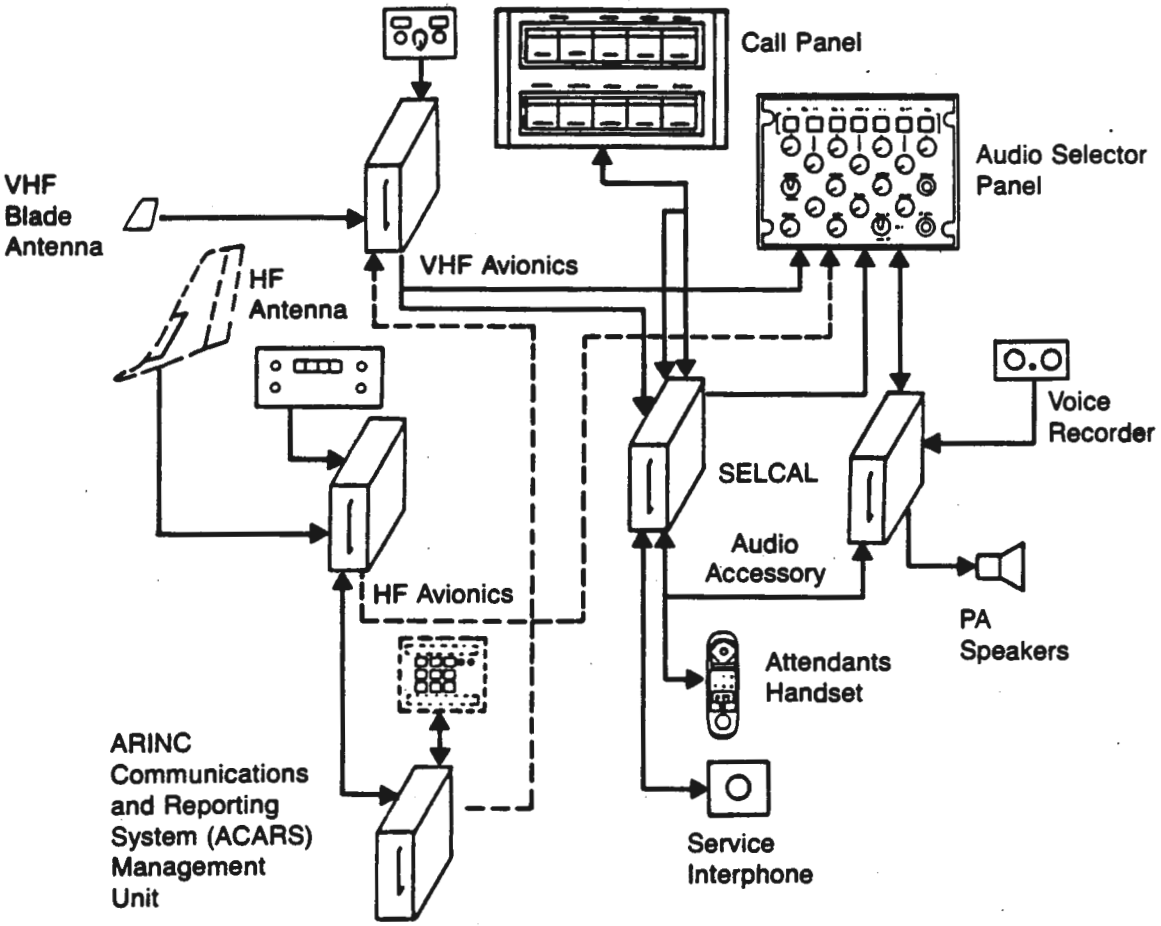



FIGURE 2.1-16. COMMUNICATION

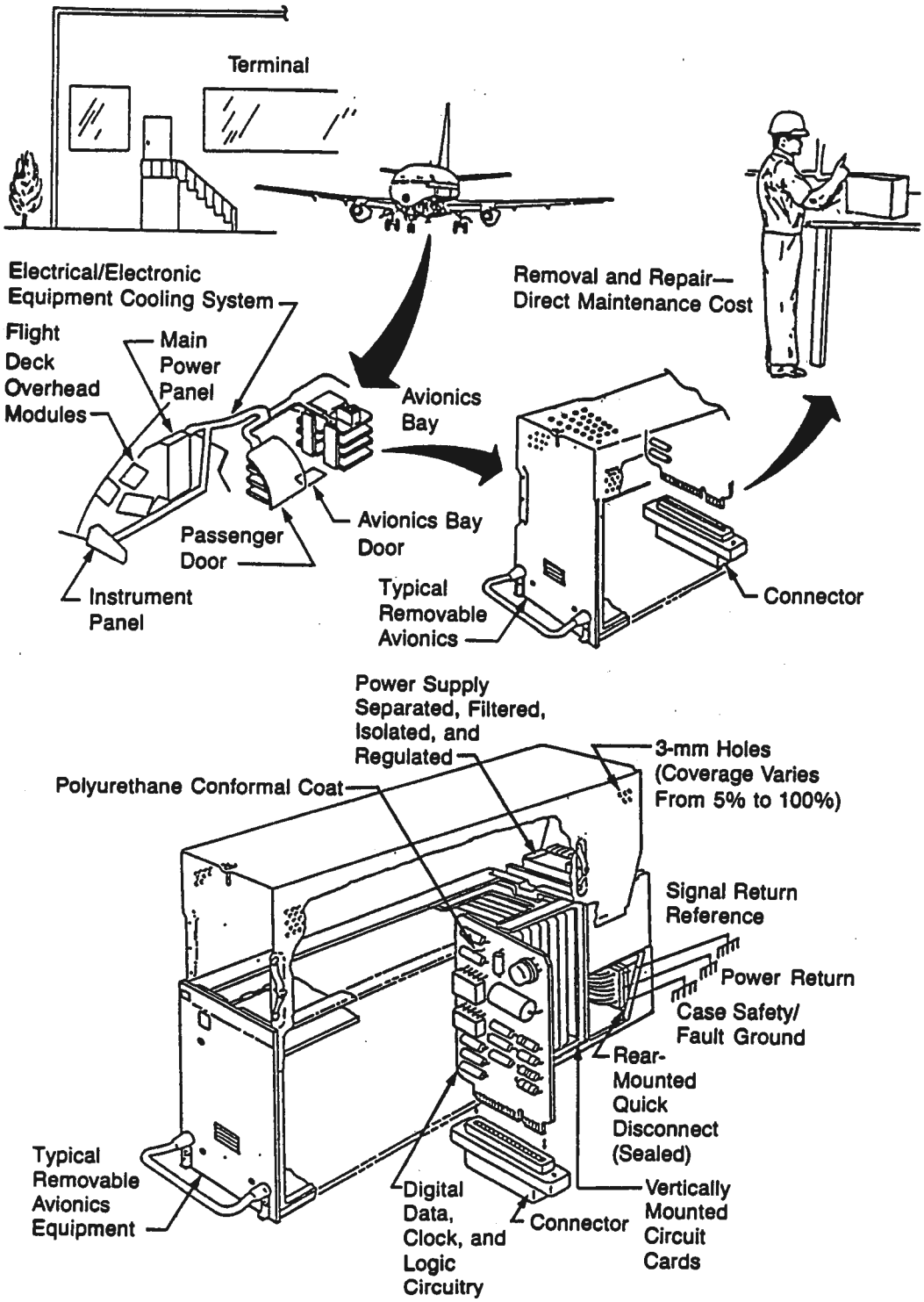



FIGURE 2.1-17. LINE REPLACEABLE UNIT

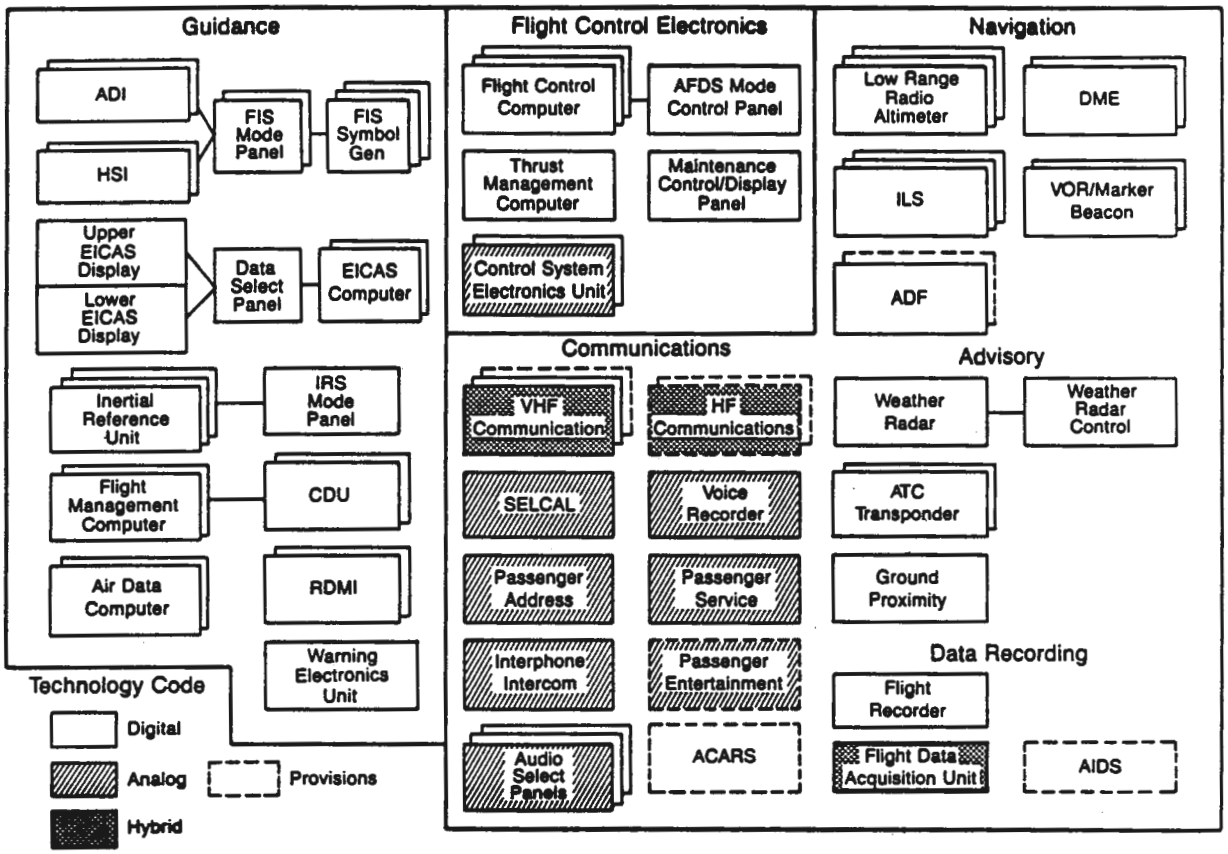



FIGURE 2.1-18. FLIGHT MANAGEMENT SYSTEMS

If one were to look at the specific types of interconnecting circuits, here are some examples of the electrical/electronic characteristics and parameters:

| | | EXAMPLES | | |
|---|---|---|---|---|
| CIRCUIT | | LEVEL | RATE/FREQUENCY | THRESHOLD |
| (1) | Digital data: | +5V | 12 kHz | 2.5V |
| | (ARINC 429 transmitter/receivers: information transfer, altitude, airspeed, direction, positions) | | | |
| (2) | Pulse circuits: | +50V pulse | 1 pps | mV |
| | (Fuel flow; engine speed) | | | |
| (3) | "Discrete" | +30V | Event ON/OFF | 10V |
| | (Status: Pumps, valves, relays, heaters) | | | |
| (4) | Analog | 0V to 10V | Continuous | mV |
| | (Position indicators) | | | |
| (5) | Power | 28V dc<br>115V ac | (sw. reg. 10 to 100 kHz)<br>400 Hz and harmonics | |
| | (Avionics, pumps, lights, motors, generator feeders) | | | |
| (6) | RF | Volts | Variable | $\mu$V |

The electromagnetic compatibility engineer helps to simplify and forge the layout of the equipment and routing of the wires (figure 2.1-19). He pays particular attention to radiated emission from wiring and radio frequency fields imposed on wiring (figure 2.1-20). Equipment location has been settled usually in the past by greater attention to and emphasis on temperature characteristics, size, separation, accessibility, and center of gravity factors rather than electromagnetic compatibility. And, rightly so. Wiring, wire clamps, shield ties to structure, and routing of wire-bundle installations have been shaped by the space available and structure geometry. They have been installed as an "add-on" while all the time being given the appellation of weight penalty. Conduit or metallic ground planes have not been needed. Shielding could be added any time.

We and our predecessors seldom created and documented substantive and comprehensive input/output interface circuit drawings to provide an analysis on a systematic basis of the noise sources and the major avenues of coupling. Electromagnetic interference sometimes has been most elusive and difficult to analyze, model, and predict.

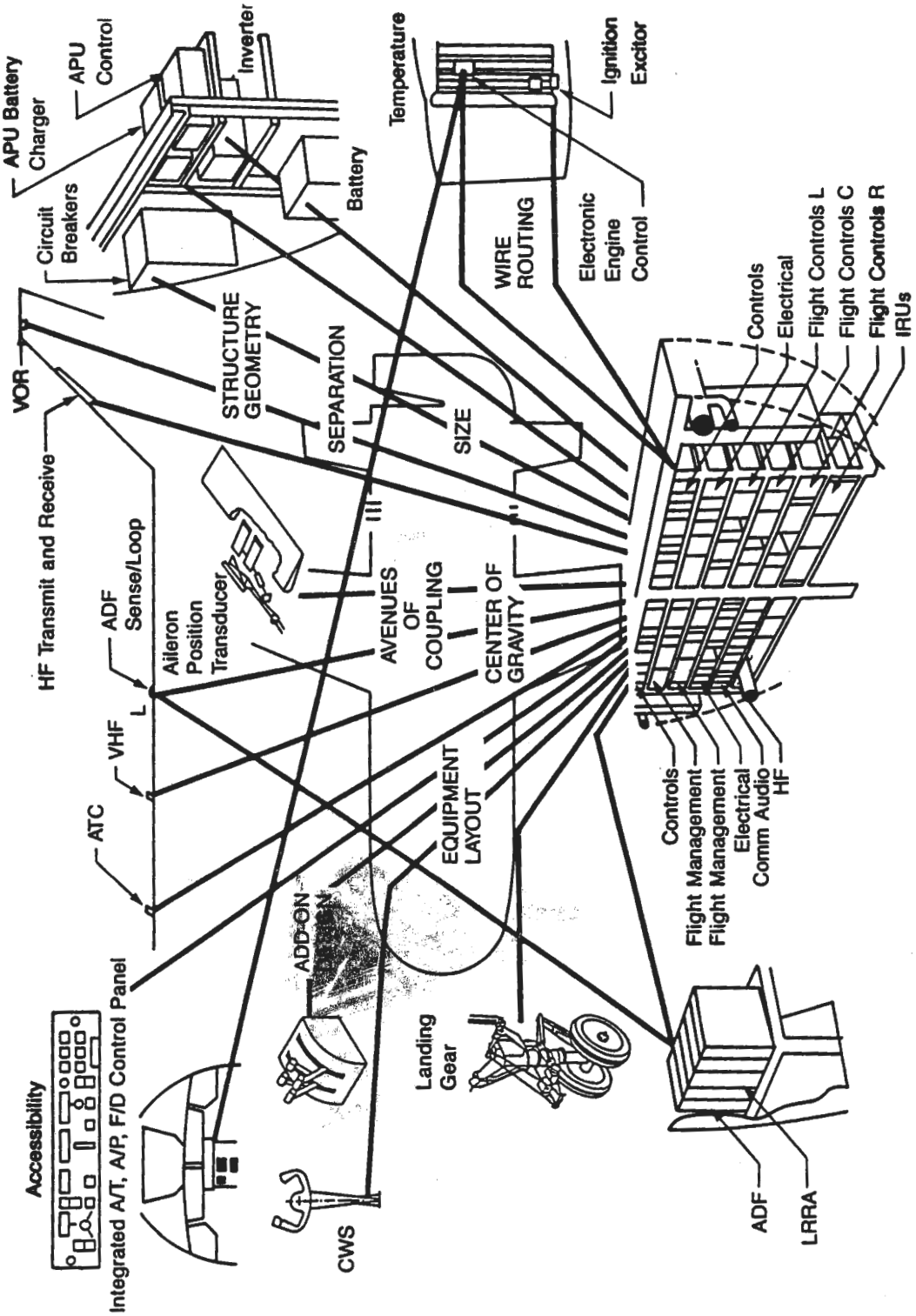



FIGURE 2.1-19. EQUIPMENT/WIRING LOCATION COMPLEXITIES

Avionic equipment units have been designed and tested to industry electromagnetic compatibility specifications and then, along with their interconnecting wiring, installed directly in the aircraft. They almost always perform properly. This method has been cost-effective and not at odds with safety. Fixes to equipment or to the airplane can sometimes be made fairly easily and inexpensively, although they are often done in a hurry and with the ever-present specter of the very expensive "retrofit."

As we look back, it is recognized that the electromagnetic compatibility effort has been reactionary, not anticipatory.

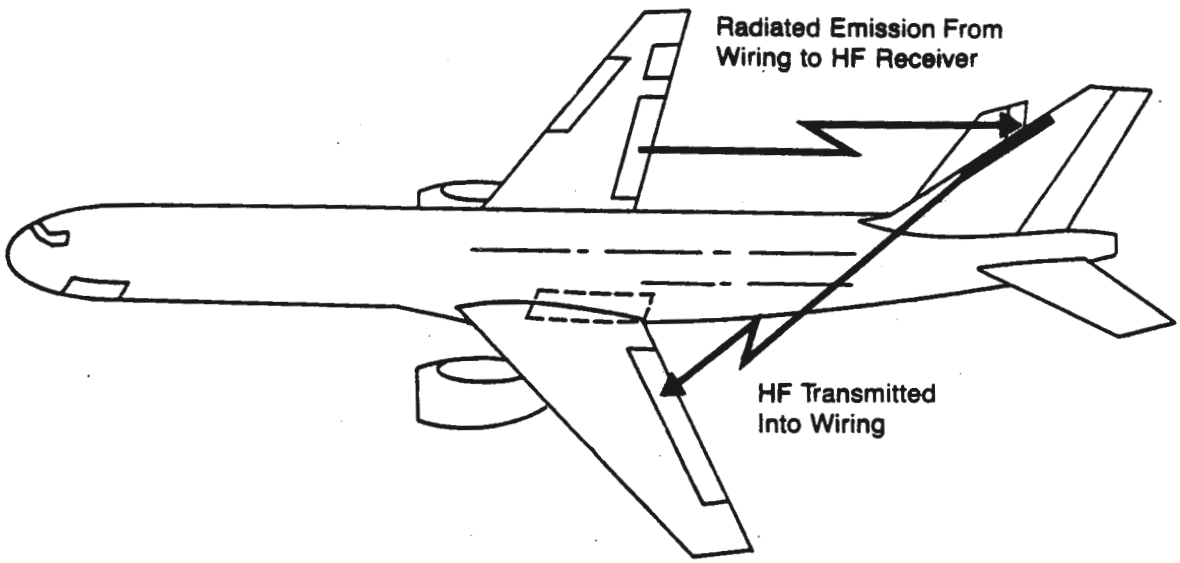



FIGURE 2.1-20. RADIATED EMISSION

## 2.2. Future Aircraft

Turn now and look ahead to a hypothetical future aircraft.

A view of a system avionic architecture might focus on the technologies of digital buses, electric actuators, and composite structural materials as having the greatest interest for the electromagnetic compatibility engineer or airworthiness specialist (figure 2.2-1). A system of communication and control designed around fly-by-wire digital data buses and electric actuators will have an impact on analysis tasks; that is, greater effort will be needed to assure absence of electromagnetic interference in the data bus and control of electromagnetic interference from the actuators and pulsed dc.

Connection of autonomous electronics through a multiple access two-way data bus will remove dependance on a centralized controlling processor (figure 2.2-2).

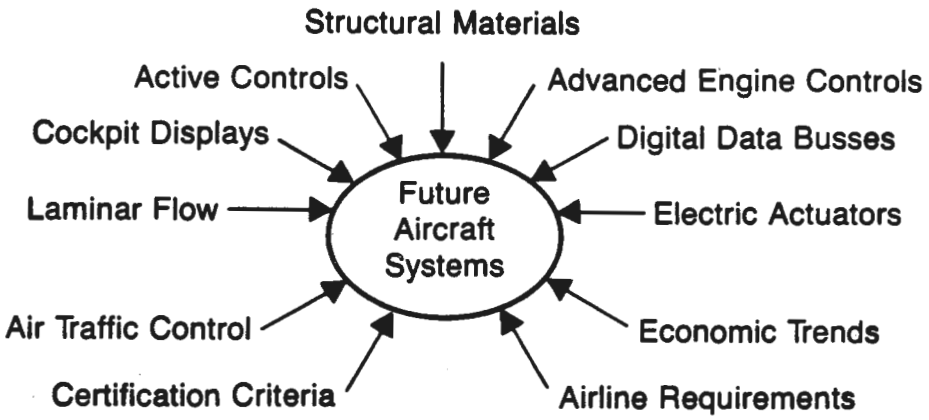



FIGURE 2.2-1. EMC AND AIRCRAFT SYSTEMS DEVELOPMENT

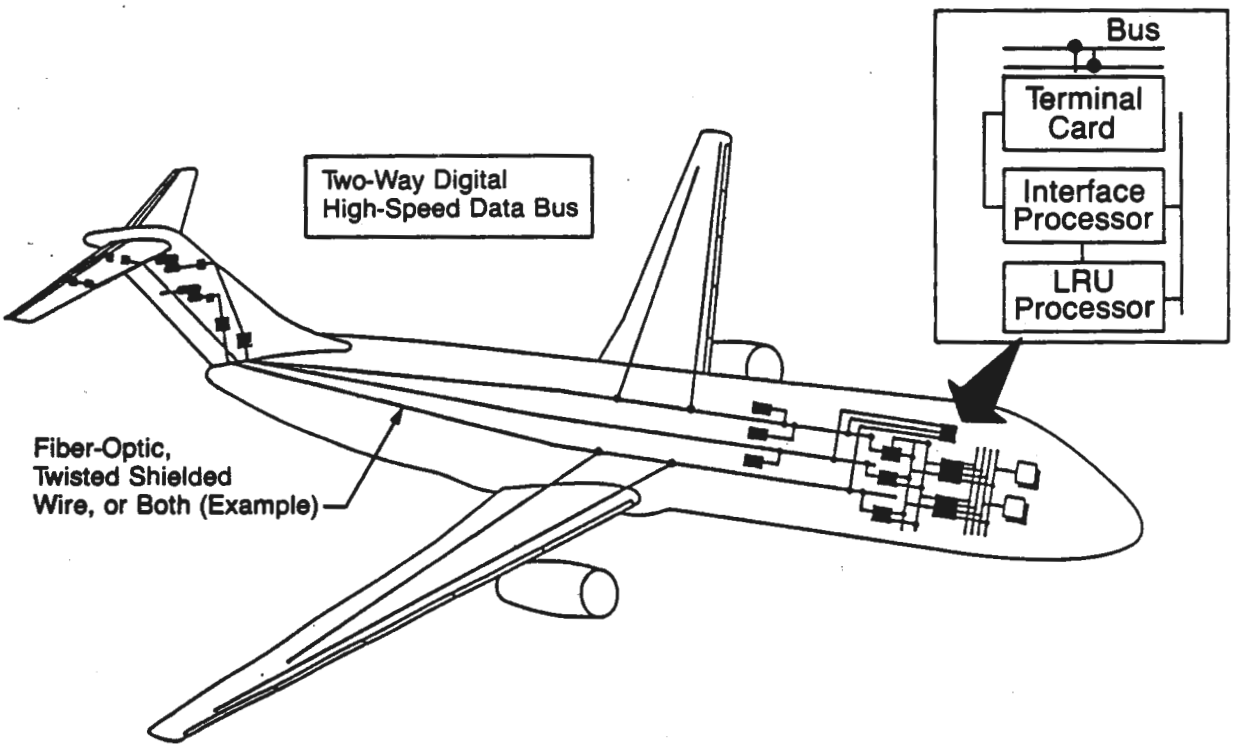



FIGURE 2.2-2. DATA BUS ARCHITECTURE

There will be a number of new aircraft technologies that will moderate problems but will mean new areas of expertise and study:

- All-electric airplane (electric actuators), pulsed dc.
- Autonomous, multiple-access data bus with decentralized computers (dual redundant).
- Self-diagnostics, self-test, error control, and record keeping.
- Electronic keyboard actuation of power and avionics.
- Side-stick controllers.
- Single point-of-entry program loading.
- Flat-panel and head-up displays.
- Voice control.

What will be the impact of these systems on electromagnetic compatibility? More inductive switching transient control? Electrostatic transient control? Better high-energy radio frequency fields rejection and control?

Flat panel displays collocated with their own microprocessor avionics on the flight deck, depending on design, will mean less wiring and fewer coupling paths, less susceptibility, less electromagnetic radiation and electromagnetic interference. An improved flight management system will integrate navigation, communication, guidance, and energy management. Moreover, there will be expanded management capability of all of the systems (figure 2.2-3).

Multi-function keyboards will permit speed, flexibility, and maximum accessibility of options and data selection initiated by voice (figure 2.2-4). Side-stick controllers, electronic throttle, and flap control will not only facilitate operation and open up the instrument panel viewing area, but will also demand guarantees of safe operation and freedom from noise. Head-up displays and electronically-controlled relays and actuators will help clear and simplify instrumentation and add more power and versatility to aircraft control on the flight deck.

One of the key elements in a future aircraft is the data bus, where wire and weight savings will be dramatic. The number of interface circuits will decrease (figure 2.2-5). Research on digital buses will be required to better define susceptibilities to high-energy radio frequencies and transients. It is expected that some form of multiple access bus will be available for future aircraft (figure 2.2-6). Commercial transport planes are now using the ARINC 429 bus, a 12-kHz or 100-kHz, unidirectional, twisted pair, shielded bus that operates from each transmitter to multiple receivers with a binary-coded decimal

format. The military bus is MIL-STD-1553: a 1-MHz, bidirectional, twisted pair, shielded, transformer-isolated bus totally run from a centralized 1553 bus controller with a serial, Manchester II, bi-phase format. (See section 10, Bibliography, ARINC 429 and MIL-STD-1553 bus system.) New systems are being developed to provide multiple access and decentralized processing. One such is called DATAC, a 1-MHz, bidirectional, twisted pair or fiber-optic bus with controlled specifications of protocol to facilitate transmission and reception.

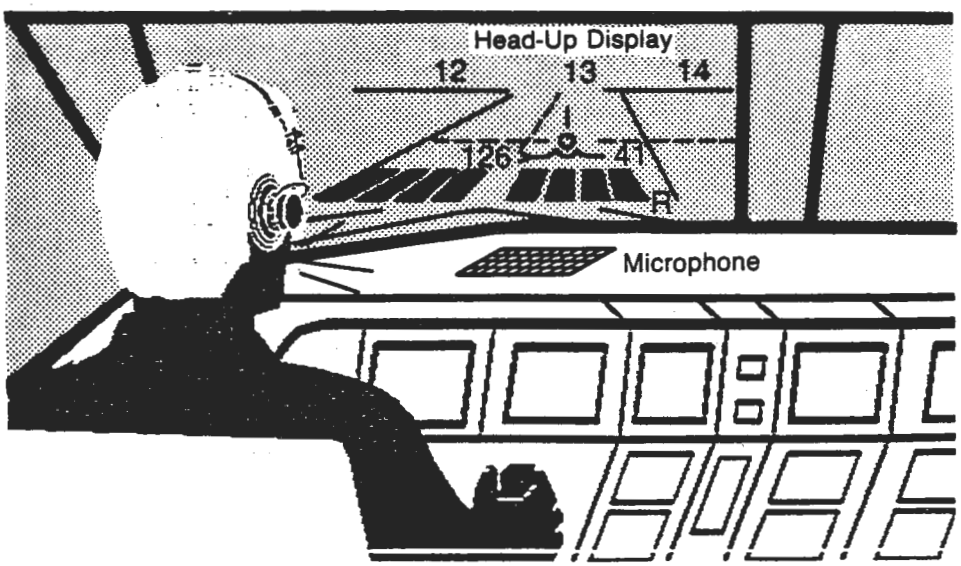



FIGURE 2.2-3. LINE-OF-SIGHT DISPLAY AND VOICE CONTROL

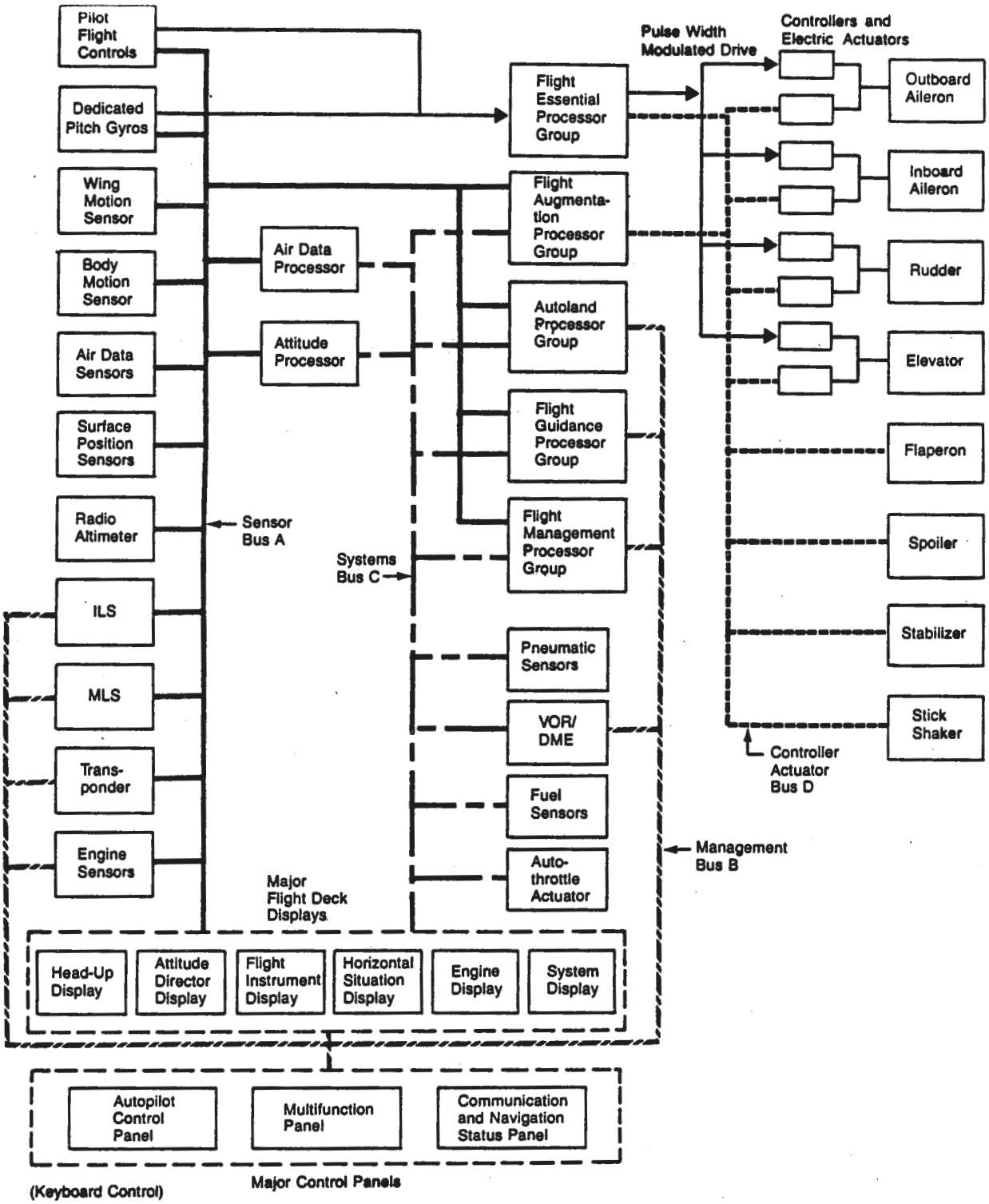



FIGURE 2.2-4. HYPOTHETICAL ARCHITECTURE

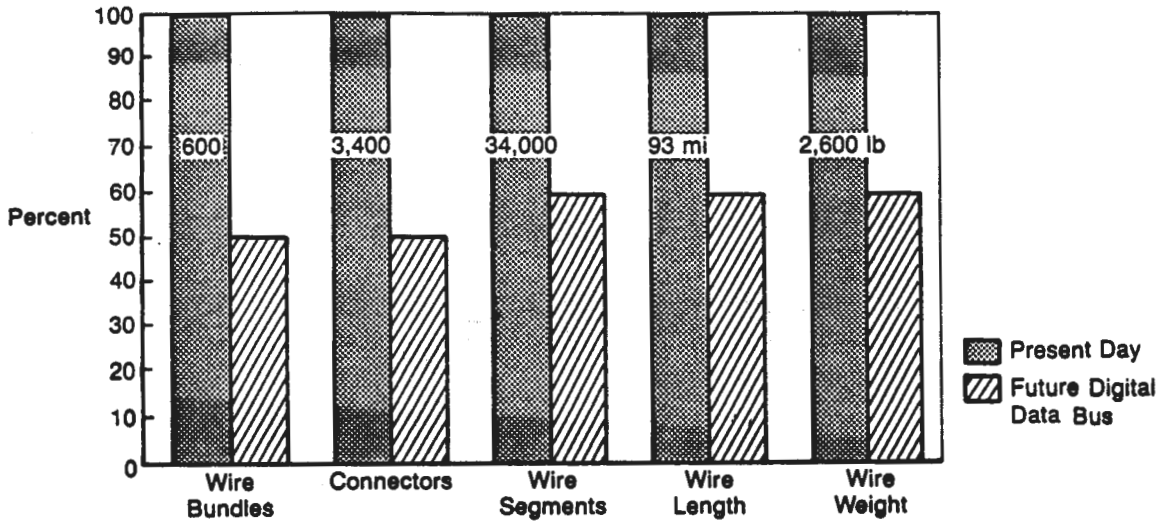



FIGURE 2.2-5. DATA BUS COMPARISON

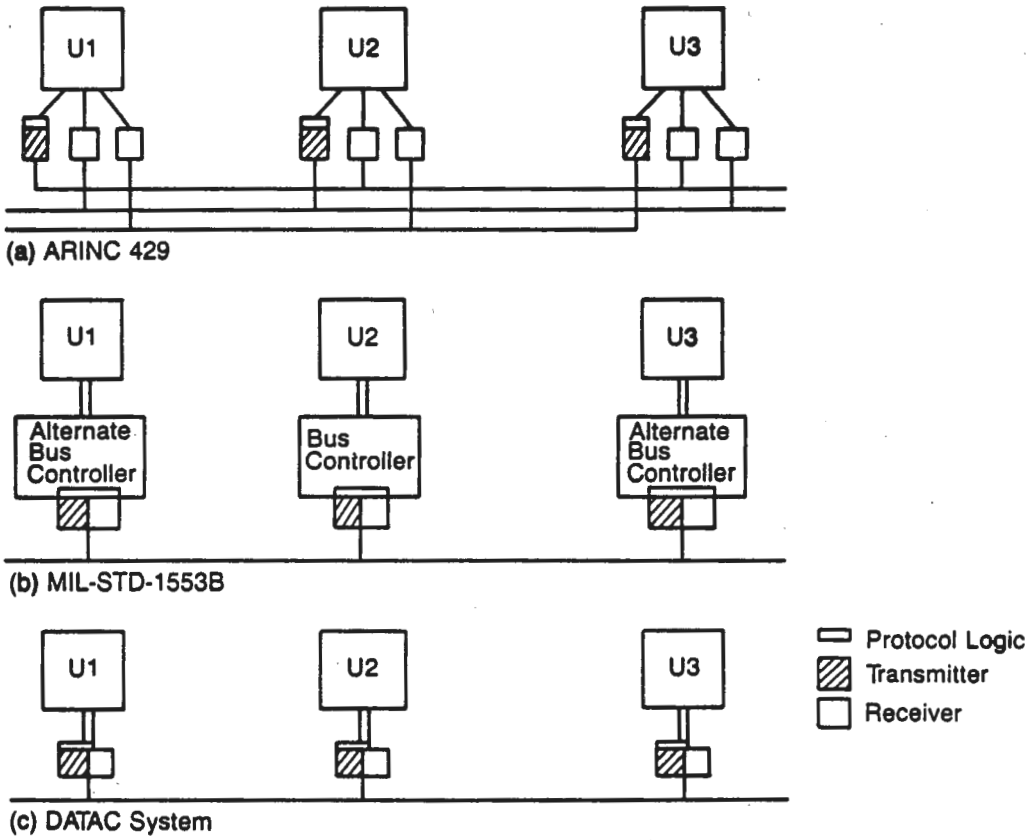



FIGURE 2.2-6. DIGITAL DATA BUSES

Possibly one of the most revolutionary changes to come about for electromagnetic compatibility will be the increased introduction of self-test and self-diagnostic capabilities in avionics allowing real-time readout or a history of equipment susceptibilities to noise (figure 2.2-7). Microprocessor evolution is speeding the development of these systems, and dramatic changes will occur in the immediate future (figure 2.2-8 and figure 2.2-9).

Hydraulic actuators powering control surfaces and other aircraft items such as landing gears and doors may be replaced by all-electric actuators now being developed. The actuator controllers contain "pulse width modulated" signals driving stepper motors. They are relatively high-voltage devices, are noisy, and the pulses need to be contained.

New composite and metallic alloy structural materials will mean greater efforts in the evaluation of the cornerstones of electromagnetic compatibility - the grounding and bonding designs that maintain current paths and stable electrical references (figure 2.2-10).

Many existing systems are being advanced and improved (table 2.2-1 and figure 2.2-11).

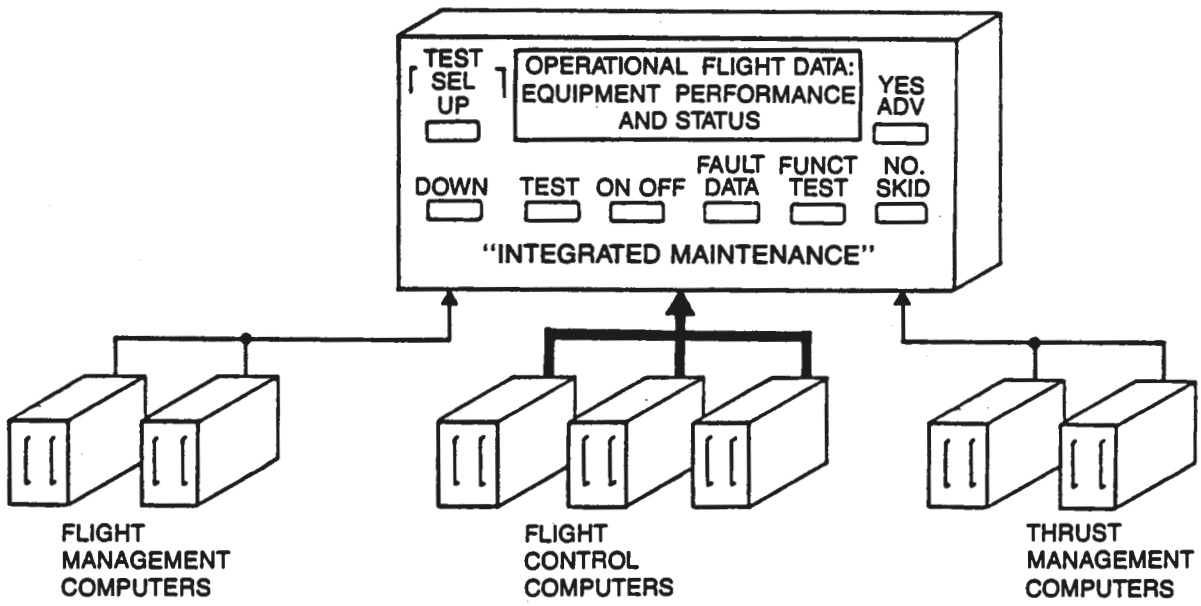



FIGURE 2.2-7. PERFORMANCE AND STATUS MONITORS

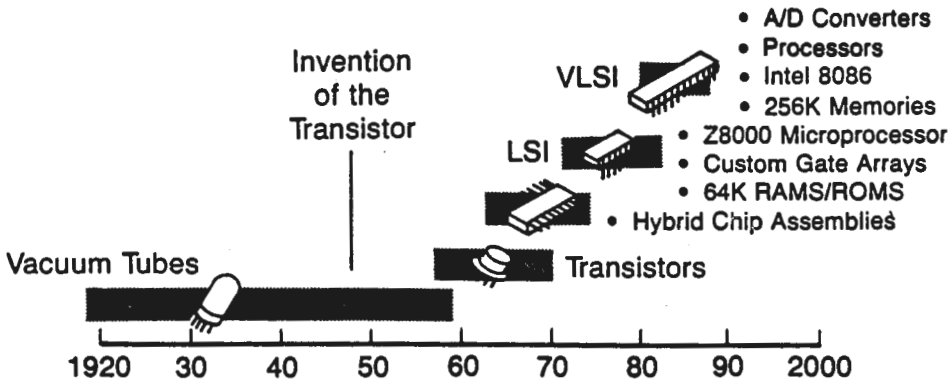



FIGURE 2.2-8. MICROPROCESSOR EVOLUTION

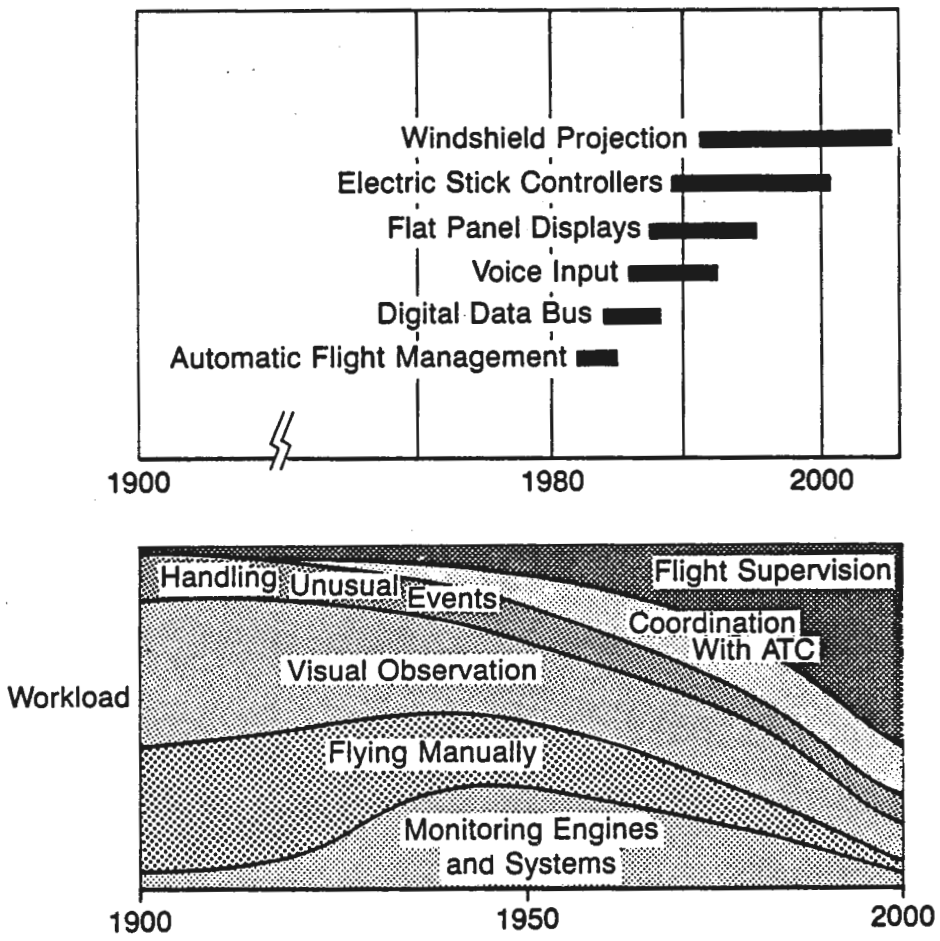



FIGURE 2.2-9. AVIONICS TECHNOLOGY PROGRESS

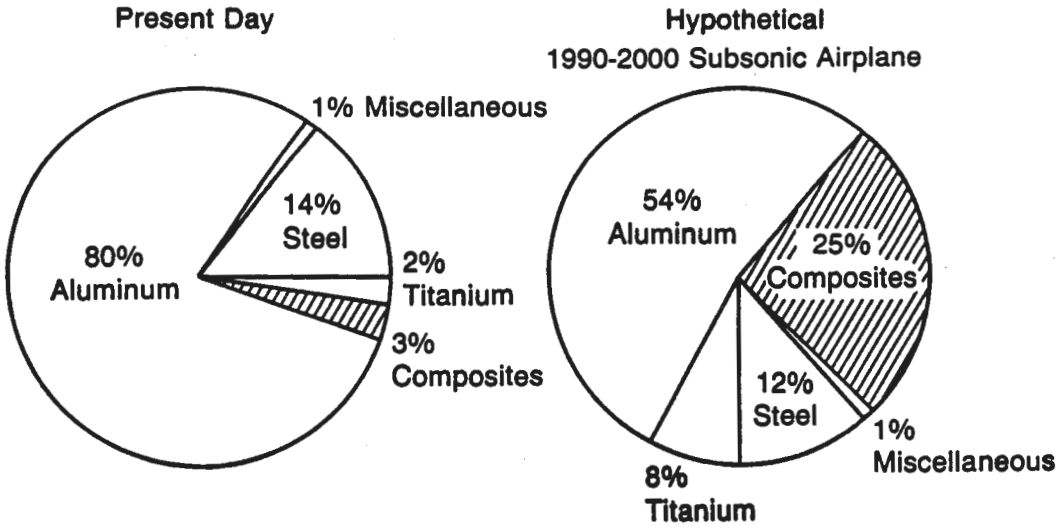



FIGURE 2.2-10. MATERIALS DISTRIBUTION

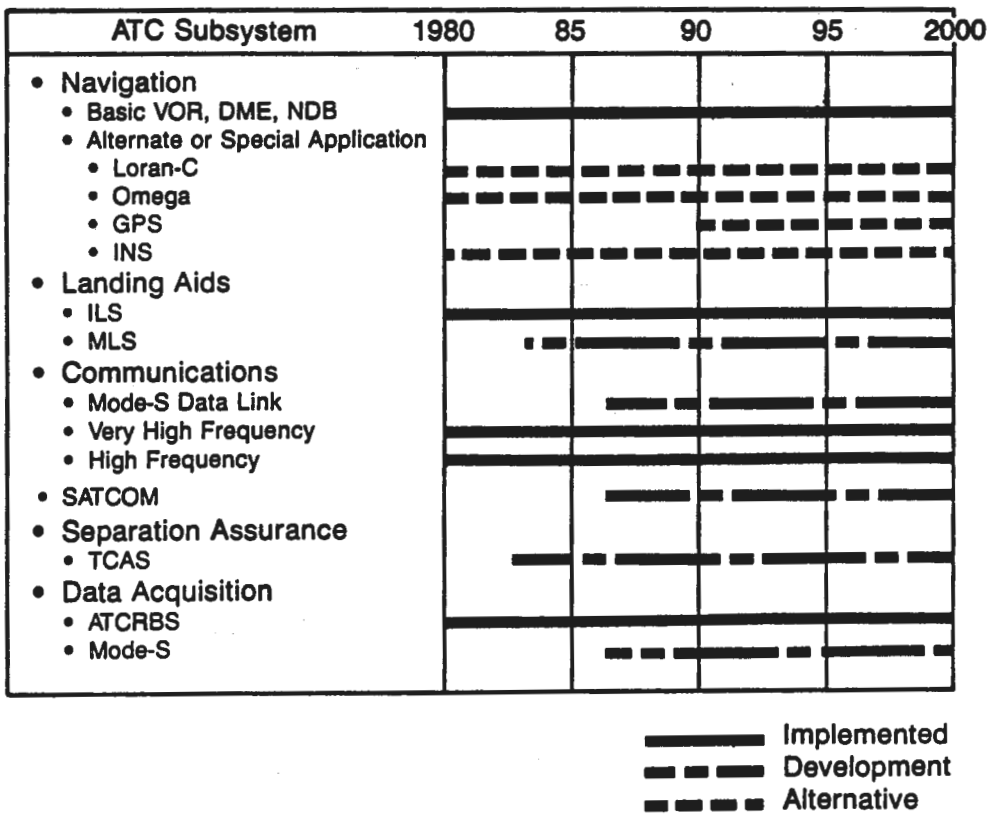



FIGURE 2.2-11. 1990'S ATC IMPLEMENTATION

TABLE 2.2-1. 1990's ATC IMPLEMENTATION

| SYSTEM | APPLICATION | AVIONICS |
|---|---|---|
| • DATA ACQUISITION<br>• ATCRBS WITH MODE-S | • Mode-S-Equipped Transponder Required on Air Carrier Aircraft | • Mode-S Transponder |
| • SEPARATION ASSURANCE<br>• TCAS | • Required for All Air Carriers | • Interrogator, Controls, and Displays |
| • COMMUNICATIONS<br>• VHF VOICE | • Most U.S. Domestic and Foreign ATC Operations | • VHF Transceiver |
| • MODE-S D/L | • High-Density U.S. Airspace | • Mode-S Data Link Modem and I/O Devices |
| • HF SSB | • Overocean and Lesser Developed Overland Air Routes | • HF SSB Transceiver |
| • SATCOM | • Overocean Voice and Data | • Data Unit and RF Unit |
| • NAVIGATION AIDS<br>• VOR | • Required for Short-Range Navigation | • Receiver |
| • DME | • Required for Short-Range Navigation | • Interrogator |
| • NDB | • Needed for Navigation and Approach Guidance in Some Areas | • Automatic Direction Finder |
| • INS | • Used for Long-Range Navigation Independently or With Other Systems (e.g., for Position Fixing) | |
| • OMEGA | • Used for Long-Range Navigation Independently or To Position-Fix INS in Either VLF or Omega Modes | One or More Types Needed for Long-Range Navigation; INS Installation Must Be at Least a Dual System |
| • GPS | • May Find use for Either Short- or Long-Range Navigation or To Position-Fix INS | |
| • LANDING AIDS<br>• ILS | • Required Until 1995 or Until All Destination Runways Have MLS | • ILS Localizer, Glideslope, and Marker Beacon Receivers |
| • MLS | • Required After 1995 but Needed Before To Obtain Improved Landing Guidance Available at Runways Where Implemented | • MLS Receiver |

The air traffic control radar beacon system (ATCRBS) operation (based on a ground interrogation of an airplane transponder to assess flight identity, altitude, range, and azimuth) is being upgraded with a new beacon mode, called Mode-S, which is a two-way digital link, to selectively address each aircraft. Surveillance en route and at terminals during approach starts at about 6000 ft (1830m) above ground level.

The airport surface detection system (ASDS), a primary radar system, is being implemented to transmit a pictorial presentation of the terminal surface area in order to expedite traffic.

The Instrument Landing System (ILS) consists of a 108- to 112-MHz localizer for horizontal, a 328- to 335-MHz glide slope for vertical, and two 75-MHz marker beacons for distance to provide position starting at 33 km from the runway (18 nmi). The system is used extensively and will be augmented with a new Microwave Landing System (MLS) to provide landing in zero visibility utilizing a ground-transmitted signal to the aircraft to establish elevation, azimuth, and distance and is now collocated with the MLS until replacing it.

Navigation systems will expand in extent and capability. The Traffic Alert and Collision Avoidance System (TCAS) requires a transponder trace, and interrogator to detect, track, and compute aircraft flight path projections and initiate selected levels of warning and avoidance maneuver advisories.

The Very High-Frequency Omnidirectional Range (VOR) establishes a magnetic bearing, is a short-range aid, and is generally collocated with a Distance Measuring Equipment (DME) station, which provides for more precise distance measurement from the VOR.

The Automatic Direction Finder (ADF) uses nondirectional beacons (200 to 415 kHz) to provide a bearing ($\pm$ 3 deg) at a range of 18 to 650 km (10 to 350 nmi).

Loran-C covers about 1850 km (1000 nmi) with pulses at 100 Hz with a positioning accuracy of 0.46 km (0.25 nmi), repeatable to 18m to 90m (60 to 300 ft) and is operated by the Coast Guard along the contiguous U.S. and Alaska, but is presently not being installed by scheduled air carriers.

Many of these systems will be supported or replaced by the Global Positioning System (GPS) which in future years will provide worldwide, accurate, all-weather positioning.

Communication systems will see unparalleled improvement in reliability and flexibility.

Very High-Frequency (VHF) voice communication, the continental U.S. air traffic control system to airport towers and control centers, operates between 116 and 136 MHz with projected bandwidth spacing of 25 kHz at a receiver sensitivity of around a microvolt.

High-frequency (HF) voice communication, for use over water, operates at 2 to 30 MHz with a receiver threshold of about 2$\mu$V and transmitter output of possibly 400W peak effective power.

## 2.3. Criticalities

### 2.3.1. Measures and Definitions

By any measure, airplanes are safe. Airplane avionics are a part of that safety picture. And electromagnetic compatibility is a part of the proper operation of avionics.

Although safety records have been built and established over the years, the criticality of avionics and their interconnecting wiring is not an unchanging situation. Of course, mathematically finite possibilities of avionics performance errors or malfunctions always exist, and it almost goes without saying that absolute, definitive resolution of the exact probabilities and reliability of critical circuits is elusive.

"The simultaneous failure of two reliable independent systems, each of which has dual redundancy," states FAA Advisory Circular AC No: 25.1309-1, "is expected to be extremely improbable." But still, the performance of critical avionics equipment cannot be allowed to be affected by the noise in the airplane.

Critical circuits need evaluation. Evaluation is needed on non-essential systems for their effect on critical circuits. Flight crews must not be given misleading information. Noise effects can be barred with shielding, rejected by balanced circuits, diverted and contained with filtering, or neutralized by software.

Table 2.3-1 itemizes some equipment criticality categories and definitions, and figure 2.3-1 charts a general relationship of probability and consequence.

TABLE 2.3-1. CATEGORIES OF CRITICALITY

| Category | Function | Impact | Probability of Occurrence |
|---|---|---|---|
| A | Critical | Prevent safe flight | Extremely improbable $\leq 10^{-9}$ (per 1 hr.) |
| B | Essential | Impaired ability to cope | Improbable $\leq 10^{-5}$ to $> 10^{-9}$ |
| C | Nonessential | No significant degradation | May be probable $> 10^{-5}$ |

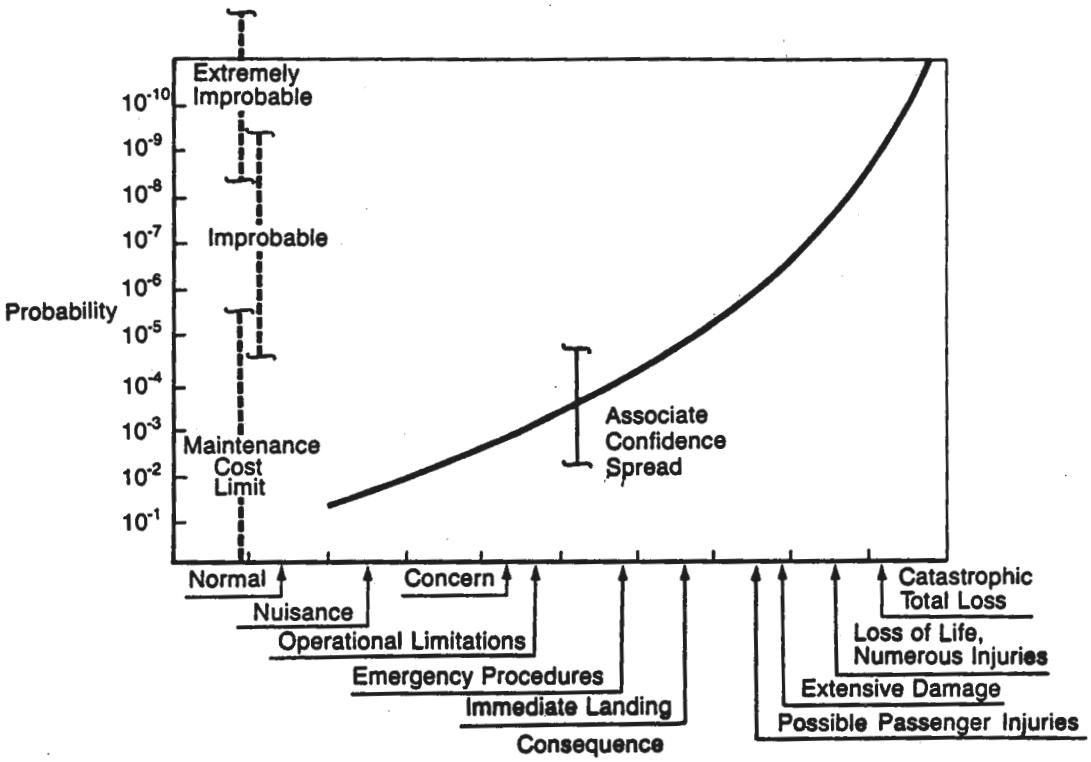



FIGURE 2.3-1. PROBABILITY VERSUS CONSEQUENCE

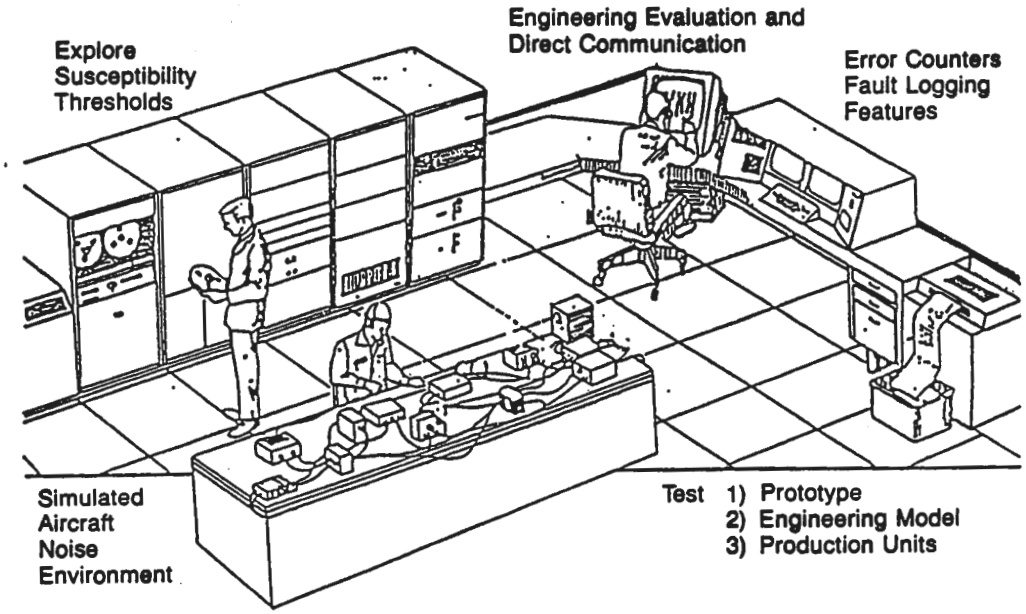



FIGURE 2.3-2. THREE-STAGE SUBSYSTEM TESTING

Toward the end of an airplane program, malfunctions or upsets can practically be brought to zero by repeated subsystem testing by the airframe manufacturer in conjunction with the avionics manufacturer of, first, prototypes, then engineering models, and finally production units. The avionics operation in a simulated noisy environment will be understood, and verified, and the data and experience will help contribute to aircraft validation (figure 2.3-2).

### 2.3.2. Critical Equipment

What makes equipment critical? What conditions influence criticalities? Here are some defining generalities:

Function:

- Does the unit support safety of flight or is it for convenience, comfort, work relief, or economy?
- Is the unit employed to maintain flight, or is it needed to proceed to the nearest airport, or to continue to destination?

Redundancy:

- Is the unit triple or quadruple redundant?
- If the first unit fails and the second unit fails, can loss of aircraft be averted?

History:

- What is the history, condition, and age of the unit or the aircraft?
- Are all units operating at flight dispatch?

Flight conditions:

- Is criticality based on phase of flight: takeoff, climb-out, cruise, landing?
- Is criticality based on type of flight: deficient or lack of navigation facilities, instrument flight, heavy weather, long flight, over water, nighttime operation?
- Is flight safety dependent on automated electronic controls, instruments and sensors?

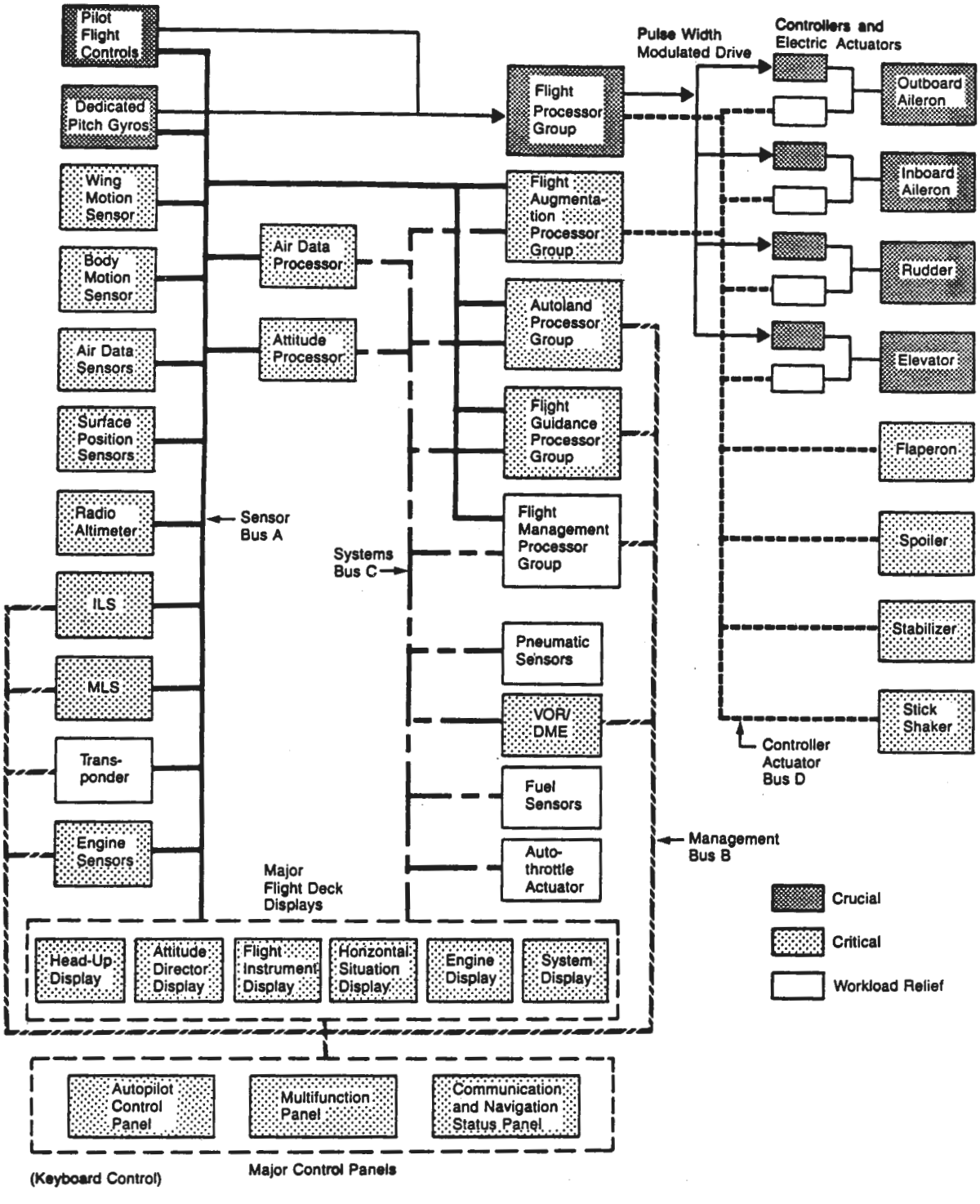



FIGURE 2.3-3. PROJECTED CRITICALITIES

What circuits are classified as critical? Certain circuits or functions must be extracted and given special attention for electromagnetic compatibility. Circuits that might be considered include these illustrative examples:

- Flight control/flight management: control surface actuators, displacement transducers, servo valves, position indicators, switches/valves; electric controllers/actuators; negative stability controllers; augmentation controllers; air data, attitude, altitude, airspeed, and situation indicators, displays, control panels and their backups; automated landing system.

- Navigation/communication: VHF transceivers, voice recorder, tape recorder, ADF, radio altimeter, instrument landing system, marker beacon.

- Power: standby power, instrument lights, standby instruments.

- Advisory flight instrumentation: actuators, position indicators, displays, test circuits, pressure, temperature, quantities.

- Fire detection systems.

- Landing gear: antiskid control.

- Engine: controller actuators, computers, displays, temperatures, speeds, pressures, quantities, restart/shutdown, thrust reversers.

Figure 2.3-3 delineates a block diagram of some hypothetical categories of future aircraft systems.

## 2.4. Packaging and Architecture

The aircraft is an electrical/electronic package. Right from the start, the aircraft packaging design must include a layout and topology that optimizes electromagnetic compatibility. Here are the dominant EMC desired designs:

- Major subsystems are grouped together for an in-line equipment design, input to output, to draw out the shortest possible wire bundle routing.

- Major incompatible wiring groups are separated: power feeders from electronics; analog (with single point ground) from digital; high voltage/high frequency from digital; low-level sensitive from power.

- The aircraft has a designed system-level shielding barrier combined with a designed, controlled, equipotential ground plane system.

- Every interface circuit, electronics and power, is filtered and protected. Controlled transmission line design techniques are employed for susceptible and digital interface circuits.

- Every installed unit of equipment meets the EMC qualification test requirements.

All of these designs and technologies need early conceptual consideration, planning, and lay-out. A mockup is invaluable. In-line design is the best. Locate equipment to minimize wiring lengths. Electromagnetic coupling increases with length. Make wire connections short. It's good for weight. It's excellent for electromagnetic compatibility.

Here is a more detailed checklist.

Grounding:

- Ground connections cleaned and burnished, no paint, must verify.
- Ground wires evaluated for resonance, capacity, and continuity.
- Avionics case grounded with a low resistance (and verified).
- Grounding paths and references designated on drawing.
- Trays, conduits, metal liners, foils designed for return currents.
- Graphite-epoxy not used as structure return.
- Analog circuits are single point grounded or isolation provided.
- Each avionic unit return/ground identified.
- Engine circuit return/ground analyzed for wire length and impedance.

Bonding:

- Structure electrically bonded and aperture bonds confirmed.
- Electrostatic conductive paints applied to external dielectric or nonconductive surfaces.
- Equipment bonding to structure verified.
- All isolated metal objects bonded.

Shielding:

- 360 degree, peripheral shield connections used.
- Backshell continuity defined and checked.
- All circuits evaluated for shielding.
- All low-level, sensitive circuits shielded.
- Shield noise not carried inside case confirmed.

Wiring:

- Landing gear wiring installed in conduit and flexible overbraid.
- No wiring routed under unshielded fairings.
- Wiring installed close to structure.
- Shielded, balanced, isolated interface circuits employed.
- Power feeders given a power line and a wire return.
- Signal lines given a signal line and a wire return.
- Twisted pair used for audio circuits.
- Oscillator harmonic frequencies emanating from clocks, switching regulators, pulse width modulators, and digital data lines shielded. Conducted and radiated emissions, close to receivers, may need to be controlled below DO-160 limits.

Packaging/installation:

- Test data dug out to verify equipment complies with specification.
- Equipment tested to up-to-date requirements.
- Transformers separated from CRTs and audio circuits.
- Environmental control, flight control/electric actuators, flight management, radio transmitters and receivers, power, and engine control systems unearthed, defined, grouped, and aligned.
- Mockup constructed to settle on equipment topology.
- Functional or subsystem grouping provided for environmental control electronics, switching units, valves, fans; flight control, electric controllers located next to stepper motors; air data, caution advisory computers next to their transducers; radio frequency units, power supply, electronics, amplifier, antennas; heavy power generators, converter regulators, transformer rectifiers, switching contactors, power switching unit installed together, and separated from avionics; power "single point grounds" positioned next to power regulator or distribution unit; power switching unit centrally located; dedicated avionic power supplies installed next to the unit they supply; electronic engine controllers positioned to shorten and minimize wiring to engine sensors and controls.
- All circuits evaluated for filtering.
- Equipment input and output wiring not doubled back upon itself.
- All circuits evaluated for categories and separation.

These are not all the answers but they will help.

## 2.5. Issues

### 2.5.1. An Early Start

The whole EMC scenario rises up and needs resolution at the concept of the program. This is an issue. The scope and depth are resolved first - is it a proposal? - proof of concept? - or full production program? What are the funding boundaries?

Environmental assessment quickly follows, all of the environment inside and outside of the airplane. This is an issue.

Early in the program, light must be shed on the "tailored" design requirements or waivers for subcontractors. New systems demand definition. Policy and schedule come into focus in a brief EMC plan. These are large tasks and they are all at issue. Bringing two or three people on board early is an added expense. It seems unnecessary. Electromagnetic compatibility design does not start at the design stage, it starts at the concept stage.

### 2.5.2. Design

#### 2.5.2.1. Shielding

Shielding is one of the best ways, the most all-around effective ways, to protect a circuit. Shielding diverts almost all of the energies of noise. Shielding stops transients as well as radio frequencies. It protects against electric fields. It helps to maintain controlled stripline impedance. It guards against arcing and sparking. But, it adds weight to the aircraft, requires maintenance, and costs more. Shields require shield ties - difficult to install, particularly for panel-mounted equipment. With adequate modeling and design effort, shielding can be judiciously applied.

#### 2.5.2.2. Power and Signal Returns and Wire Grounds

Sometimes a wire is good, sometimes bad. Many of today's circuits or case housings are referenced to structure ground with a wire, which may resonate. Many of the interface circuits between pieces of equipment do not have a wire return, which will open the circuit up to noise. Both of these practices are detrimental to EMC, but they save on weight. Wiring is good as a signal return and bad as a circuit reference.

#### 2.5.2.3. Dielectrics

By specification, wiring and connector insulations have dielectric withstand strengths at sea level of 1500V 400-Hz steady state. That 1500V weakens as one goes up in altitude. It can be down around 300V its lowest point, at an altitude of 250,000 ft. In the future, as power line voltages rise and aircraft altitudes increase, dielectric strengths and corona will become more of a concern.

### 2.5.3. Environment and Test

#### 2.5.3.1. High-Energy Radio Frequency Fields

Magnitude, pattern, frequency, polarization, modulation, and geographical location of high-energy radio frequency and radar transmitters are needed. To evaluate aircraft circuit immunity, a systems approach must be implemented to study shielding, induced voltages, circuit protection, and software correction techniques. An industry susceptibility test specification is needed. In areas such as this, compliance to DO-160 may not guarantee system-level compatibility. Efforts are underway to provide an update of high-energy radio frequency field environments and protection techniques. (See section 10, Bibliography, ECAC study on Electromagnetic Environment.)

#### 2.5.3.2. Fields From 400 Hz and Transients

Some say that the power system 400-Hz is the most troublesome electromagnetic environment on the aircraft causing 50% or more of the shortcomings in EMC quality. Audio 400-Hz "hum" derives from magnetic ("H") field. Electric ("E") fields can trigger a comparator circuit that has high-input impedance, and it is well known that powerline transients cause logic upset resulting in lockup or even equipment shutdown until the flaws are found and rooted out. Through better definition, analysis, and modeling techniques, avionics designers must be made aware that their equipment is operating in this environment. Higher design and test levels are appropriate in some instances, coupled with more attention to finding interface circuit thresholds during test. It is controversial.

#### 2.5.3.3. Test Conditions

In the laboratory during development or qualification testing, it is expensive, difficult, and meaningless to strive to recreate actual aircraft installation or production configurations. There are wire lengths, resonant conditions, and test coupling conditions that may not adequately simulate the aircraft and, because of this test deficiency, may ultimately lead to an upset occurring on the aircraft. Possibly one answer is to verify that test levels are high enough with an adequate safety margin. Subsystem tests offer information on EMI characteristics to help moderate this dilemma. Aircraft tests authenticate the final EMC design.

#### 2.5.3.4. Emission Variances

Today's conducted and radiated emission limits are restrictive under some conditions (see section 9). Computers, with digital clocks and switching regulators, sometimes emit harmonics in the HF megahertz region that defy total containment and consequently emissions may be a few dB above limits. Cases exist where it is uneconomical and unnecessary to go to extraordinary efforts to filter those emissions if it can be shown that they will not in any way radiate to local receiving antennas. Infringements may be approved and deviations granted without adverse effects on neighboring circuits. The present radiated emission limit in the VHF range is not too restrictive and, in fact,

2.5.3.5. Subsystem Testing

Subsystem testing is different in intent from equipment qualification testing. Subsystem testing, usually held at the airframe manufacturer's facility with support from the avionics supplier, provides insight into the susceptibility thresholds and emission levels while simulating noise environments of the airplane.

The tests are for engineering evaluation and they provide significant information on computer processing performance and interface data quality. Test software exercises the processing functions and the interfaces between avionics units, and further, it monitors processing, memory, and transmissions of data for any abnormal conditions. Error counters or fault logging features examine operation in real-time and provide capability for hard copy printout. Formal pass or fail standards for susceptibility and emission do not apply. Completion of the specific test procedures and the investigation is the gauge for determining success.

The features are that many interface circuits are per manufacturer's configuration, software is up-to-date, and observation is real-time. At this stage of the program, the equipment avionics engineer is available to provide rapid evaluations and judgments. Re-evaluation and decisions on rework or test changes are easy and flexible. Often these tests can be performed on a "non-interference" basis using informal procedures. They establish standards for the airplane test. Costs are low.

2.5.3.6. Aircraft Testing

Aircraft testing is the final authentic proof. It offers first hand "real-world" validation. Wiring, equipment, and installations are the final design. It is expensive, however. It is expensive beginning with the test procedure (step-by-step development and approvals), then the aircraft test itself (a labor intensive operation compounded on top of a costly unit of equipment), then continuing with the formal documentation of unplanned events during the test, and ultimately ending with an elaborate final report.

2.6. Variances in Electromagnetic Compatibility

2.6.1. Diagnostics and Troubleshooting

Flight test phoned project. Project called flight deck instrumentation staff. Then staff contacted the electromagnetic compatibility engineers.

A new aircraft on the flight line had a discrepancy in the left engine fuel flow reading. The fuel flow digital readout on the display screen of the criticality advisory system (CAS) was variable and erratic whenever the 400-Hz power to the engine Mach probe heater was energized. What was happening and what was the cause? A work authorization was quickly approved; time on the airplane scheduled; and laboratory test support called in.

On the airplane, in cramped quarters next to the electronics bay, investigators used an oscilloscope to troubleshoot the problem on the circuits running from the fuel flow meter on the left wing engine to the CAS computer located in the electronics bay. Connectors were hard to reach and remove with care. Knuckles got scraped. Equipment was difficult to move. Engine run time was expensive. As a part of the investigation, the left engine wire bundle was disconnected from the left CAS computer and reconnected to the right CAS computer to see if the problem would "follow" the bundle. It did, and that showed that the CAS computer itself was not totally at fault and that 400-Hz, 115V power in a wing wire bundle was being coupled to the fuel flow circuit.

The 400-Hz, 115V power wire to the mach probe heater, traced out on the wiring diagrams, starts in the electronics bay and runs out to the engine; but then the 115V heater return wire is taken from the engine back onto the engine strut; the wire is there connected to structure with structure being used as return back to the power source in the electronics bay. So the "high side" of the 115V power wire runs in the same bundle in the leading edge for 100 ft next to the unshielded, fuel flow circuit. There is 100 ft of "electric field" capacitive coupling.

On the other hand, the fuel flow circuit is three wire balanced, has the meter wires isolated on the engine, and has a high-resistance input to an operational amplifier in the CAS computer protected at the input by 5 kΩ resistors and diodes to ground. The two, 350-mV pulses generated by the fuel flow meter have a working period approximating the period of the 400 cycles per second. The 115V 400-Hz noise was induced right on the 350 mV.

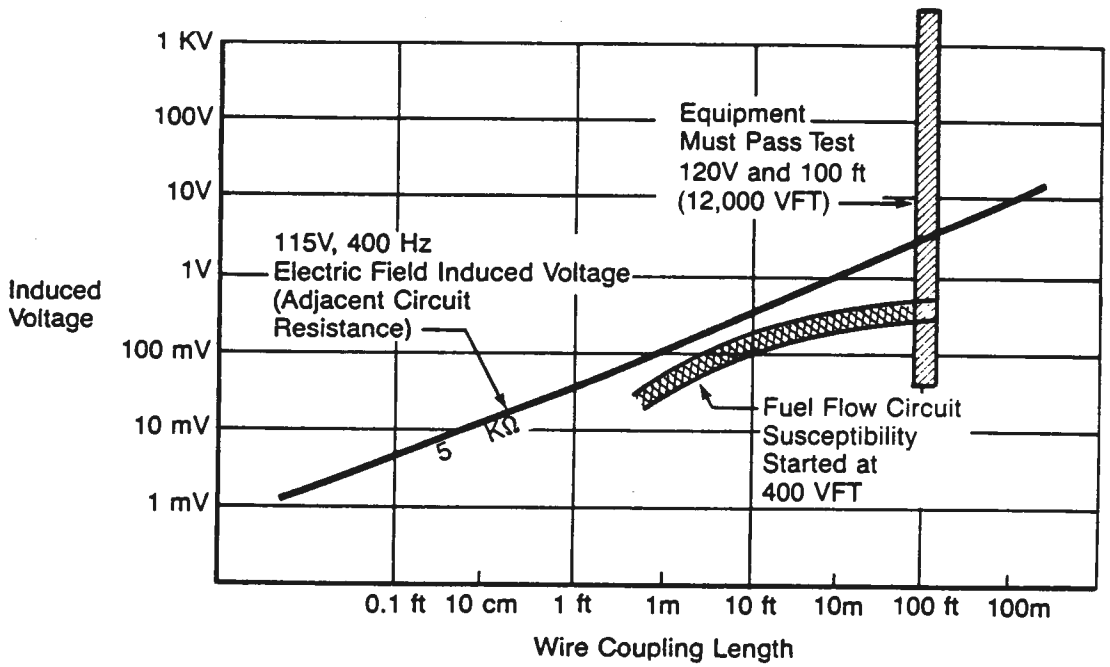



FIGURE 2.6-1. A DISCREPANCY

The fuel flow circuits were rapidly set up and simulated in the laboratory and they malfunctioned under the DO-160 power line "electric field" test just as had occurred on the aircraft. It took only 40V at 10 ft (400 Vft), equivalent to 4V at 100 ft, of coupling to cause upset (figure 2.6-1). The test requirement is 120V at 100 ft (12,000 Vft). Other DO-160 tests caused no upset. Once the simulated circuits with the proper pulses had been developed and the thresholds and boundary characteristics of the upset outlined, the test was scheduled, set up, and run in an avionics laboratory with a production configuration CAS system. With management approval, communication was established through Project with the subcontractor. He was able to duplicate the condition.

The long, 100-ft wiring run, all the way from the electronics bay through the wing pressure seal, then along the wing leading edge to the engine strut and down into the engine, had provided an extensive opportunity for electric field coupling into the operational amplifier wiring. What will correct electric field coupling? A low resistance or shield will.

Wire shielding, installed on the airplane fuel flow circuit, provided an excellent barrier against the electric field and was a quick solution.

The operational amplifier design in the CAS computer, although a balanced circuit, did not offer proper noise rejection to stop the upset. It turned out that that circuit design was inherently difficult to balance, partly because of resistor mismatch, partly because of capacitor imbalance, and possibly because of a phase shift occurring in the signal return.

This "variance" from electromagnetic compatibility on the aircraft took months to resolve. Flight test, project, manufacturing, management, and subcontractors: all were involved. Work authorizations, reviews, justifications, and documentation were invoked, processed, approved, and completed. A necessary, but costly, effort.

### 2.6.2. Table of Variances

#### 2.6.2.1. Commercial

Diagnostics and troubleshooting of electronics are continuously demanded because of avionic circuit susceptibility to transients, 400-Hz electric ("E") and magnetic ("H") fields, radio frequencies (HF-VHF) including clock and switching regulator harmonics, and power quality. Knowledge of past lessons may help save design and troubleshooting expense (table 2.6-1). The bottom line is adequacy of specifications. In table 2.6-1, under the DO-160 column, recommendations are made on possible increases in test levels to improve aircraft electromagnetic compatibility.

#### 2.6.2.2. Air Force

The Air Force has documented troubleshooting experience. "The coupling of fields through the unintentional antennas formed by aircraft wiring," Zenter, the author, says, "is the cause of many interference problems." (See table 2.6-2).

TABLE 2.6-1. VARIANCES IN EMC

| EMI CLASSIFICATION | SOURCE | EQUIPMENT RECEPTOR | PATH | SYMPTOMS | REMEDY | DO-160 | |
|---|---|---|---|---|---|---|---|
| Static: | 1 windshld | captr | dsch | dis captr | gnd wndshld | add test | ant-antenna |
| | 2 covers/pan | VHF rcv | " | aud | bnd covers | | aud-audio hum or tones |
| | | | | | | | cab-cabin |
| | 3 duct | ADF rcv | " | " | gnd duct | | cap-capacitor |
| | 4 ant cover | " | " | " | paint covr | | captr-computer |
| | | | | | | | CRT-cathode ray tube |
| Sw.trnsnt: | 1 powerline | captr | wc | del rud/ dis captr | lgnd | inc trnsnt | del-change or movement |
| | | | | | | | dis-disengage |
| | 2 | " | " | lock up | cap/softwr | | dsch-discharge |
| | 3 | " | " | false cmd | " | | "E"-electric field |
| | | | | | | | eq-equipment |
| 400Hz"E"field: | 1 powerline | captr | " | ind- FF | shld | inc "E" | FF-fuel flow |
| | 2 | N2 ind | " | ind- N2 | " | field level | fil-filter |
| | 3 | captr | " | ind-F/qty | " | | gl-ground loop |
| | 4 | aud cir | " | aud | " | | lgnd-single point ground |
| | | | | | | | "H"-magnetic field |
| 400Hz"H"field: | 1 powerline | intrphn | gl | " | lgnd | inc "H" | HUD-head up display |
| | | | | | | field level | inc-increase |
| | 2 | headset | " | " | " | | ind-indicator change |
| | | VHF rcv | | | | | intrphn-interphone |
| | 3 | videocoax | " | del CRT | Tfx iso,lgnd | | intrpt-interrupt |
| | 4 | CRT | wc | " | TP & sep | | iso-isolate |
| | 5 | tape head | " | aud | " | | lt-light |
| | 6 | aud cir | " | " | fil,sep | | N2-engine speed |
| | 7 | " | " | " | TP,sep,lgnd | | prx-proximity switch |
| | | | | | | | rad-transceivers |
| Clock hrncs: | 1 wx rdr | ILS rcv | " | " | sep | inc RF | rcdrs-recorders |
| | 2 portble rad/rcdrs | omega rcv | " | ind del: spd,course | shld | level | rcv-receiver |
| | | | | | | | rdr-radar |
| | | | | | | | RE-radiated emission |
| Sw.reg.hrncs: | 1 captr | VHF rcv | " | aud | sep, shld | inc RF level | sep-separation |
| | | | | | | | shld-shield |
| | | | | | | | sup-suppression |
| HF-VHF freq: | 1 VHF tx | handset | RE | " | fil micrphn | inc RF | tfx-transformer |
| | 2 VHF tx | headset | " | " | tfx iso | level | TP-twisted pair |
| | 3 HF tx | captr | wc | cab press | fil, cap | | trnsnt-transient |
| | 4 " | aud cir | " | aud | dublshld | | tx-transmitter |
| | 5 " | " | " | ind | " | | wc-wire coupling |
| | 6 HF eq | VHF eq | " | aud | " | | wx-weather |
| "Crosstalk": | 1 VHF, DME | aud cir | " | " | sep,lgnd | | |
| Radar: | 1 airport rdr | prx sw | RE | ind lts | shld | | |
| Power qualty: | 1 powerline | captr | 28vac | ind | fil | | |
| | 2 | " | 115v | no-land | " | | |
| | 3 | " | " | pwr intrpt | cap, softwr | | |
| | 4 | " | " | del CRT | softwr | | |

TABLE 2.6-2. AIR FORCE VARIANCES

| EMI CLASSIFICATION | SOURCE | EQUIPMENT RECEPTOR | PATH | SYMPTOMS | REMEDY | MS461 |
|---|---|---|---|---|---|---|
| Static: | 1 canopy | UHF rcv | dsch | aud | sep | add test |
| | 2 " | " | " | " | bnd,gnd | |
| | 3 ant | " | " | " | " | |
| | 4 " | " | " | " | " | |
| Sw.trnsnt: | 1 powerline | captr | wc | ind-rdr warn | shld,fil | add test |
| | 2 | | | IFF code tx | diode sup, | |
| | 3 | | | ind-HUD cam op | softwr | |
| | 4 | | | bomb disarm | | |
| | 5 | | | ind-flare | | |
| | 6 | | | del-captr memry | | |
| | 7 | | | ind-terain fol rdr | | |
| | 8 | intphn | | aud | | |
| 400Hz"E"field: | 1 powerline | omega rcv | RE | ind-del | sep | add test |
| | 2 | VLF rcv | | aud | fil | |
| 400Hz"H"field: | 1 powerline | intrphn | wc | aud | lgnd | inc "H" |
| | 2 | captr | | del CRT | sep | field |
| | 3 | " | | ind oxy | " | level |
| Clock hrmncs: | 1 captr | ant | RE | aud | fil,shld | |
| Sw.reg.hrmncs: | | --none | | | | |
| HF-VHF-UHF: | 1 tx | captr | RE | del-cntrl surfce | fil,bnd,shld | inc RF |
| | 2 | wiring | | del-steer gear | | level |
| | 3 | " | | del-rdr ant | | |
| | 4 | " | | del-winch | | |
| | 5 | " | | del-eng speed | | |
| | 6 | " | | ind-altimeter | | |
| | 7 | " | | aud-intrphn | | |
| | 8 | " | | del-A/C headng | | |
| | 9 | " | | del-nav flag | | |
| | 10 vid sig | ant rcv | | aud | | |
| | 11 strb sig | " | | aud,del-stick | | |
| | 12 dig sig | " | | aud | | |
| Radar: | 1 rdr | prx sw | RE | lose antiskid | fil,bnd,shld | inc RF |
| | 2 HF tx | captr | | ind-lts | | level |
| Power qualty: | 1 powerline | captr | 26v | del captr | fil,bnd,shld | |
| | 2 | " | 115v | dis captr | | |

### 2.6.3. Corrective Action and Modeling

A comprehensive study of troubleshooting is needed before accurate corrections to the aircraft system specifications or the RTCA DO-160 equipment specification can be recommended. Documented, statistical groupings of troubleshooting experience concerning historical EMI characteristics and cost factors do not exist, but one might postulate some possible categories and hypothetical percentages as shown in the accompanying pie charts (figure 2.6-2).

A study of categories such as these might reveal that analysis and modeling of analog/audio circuit susceptibility is most beneficial, or possibly better models of aircraft wire coupling might help, or even modeling and documenting of the most popular fixes.

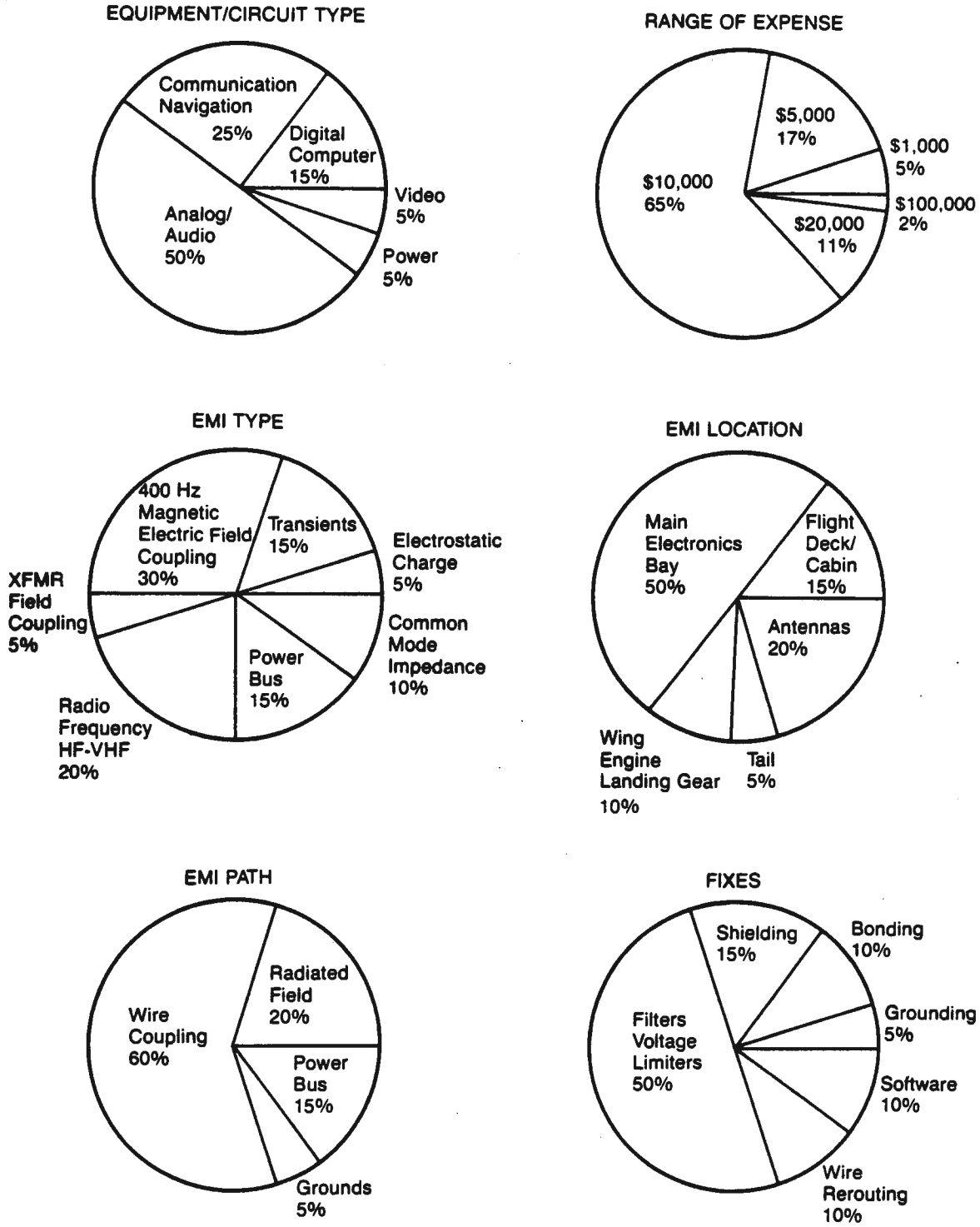



FIGURE 2.6-2. POSTULATED HALLMARKS OF VARIANCES

# 3. AVIONICS THRESHOLDS AND PROTECTION

## 3.1. Hardware Tolerance

Recognizing the conditions of proper interface wiring design, circuit protection, and the contributions of aircraft structural shielding protection is of course important in a "top-down" electromagnetic compatibility design of an aircraft, but these recognitions must also be tied closely to a knowledge of the noise tolerances and noise thresholds of transistors, micro-circuits, and logic. The lines can be drawn: environment, protection, threshold.

Airplane environmental noise voltages flourish far above the transistor and micro-circuit thresholds of damage, upset, or offset, which means that micro-circuits demand protection in every case. Transistor-Transistor-Logic (TTL) gates and microprocessors do not have even a modest tolerance to the run-of-the-mill aircraft noise types, such as electrostatic pulses, lightning induced transients, inductive switching transients, or even some high-energy radio frequency signals.

A TTL logic gate will "change state" at a threshold of about 800 mV when radio frequencies are injected starting at low frequencies and on up into the megahertz range, but in the tens to hundreds of megahertz the threshold rises to greater than 5V before upset occurs. Figure 3.1-1 maps that threshold. Whereas a gate will operate or change state at 800 mV it may be damaged if subjected to much greater than 10V at low frequencies and can usually survive about 100V transients that are of microsecond or nanosecond duration depending on thermal dissipation. As shown in the figure, a transistor or semiconductor "PN" junction will detect power levels as low as 100 $\mu$W under certain conditions.

Microprocessor chips are high density and high speed. Microprocessor circuits may be upset, change state, or change performance when signals with noise power levels down to 10 $\mu$W are injected on signal, address, or clock lines. As frequency is increased beyond the operational range of a microprocessor, the power required to cause upset and damage increases. These levels change with conditions such as loading, circuit geometry, radio frequency paths, and sometimes software design. Operational factors such as address or memory or timing or process changes affect the definition of upset. Under the onslaught of steady state, low-frequency signals, microcircuits can be damaged at power amplitudes of less than 1W, but high frequencies or fast narrow pulses require much higher wattage levels - ten to hundreds of watts.

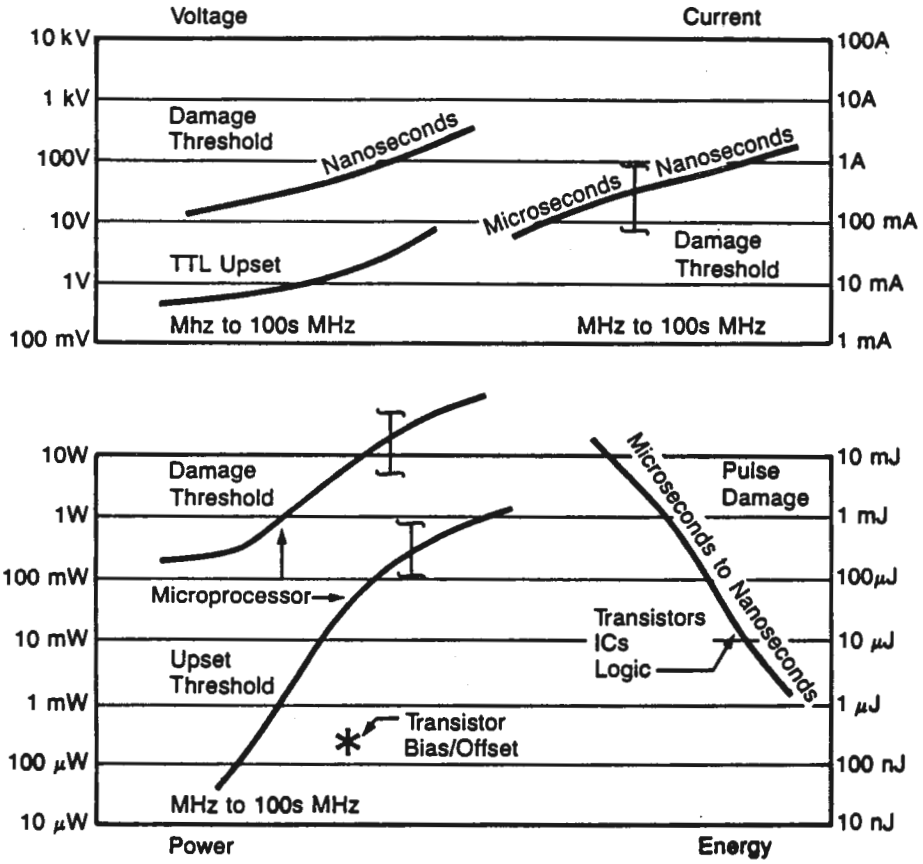
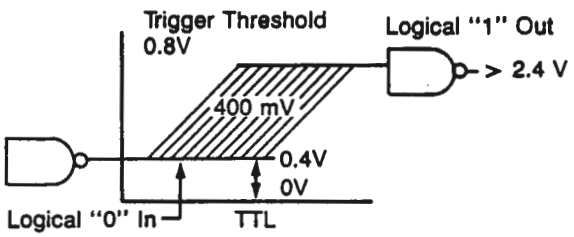



FIGURE 3.1-1. MICROCIRCUIT TOLERANCE




| | Transistor-Transistor Logic TTL | Emitted Coupled Logic ECL | High-Threshold Logic HTL | Complementary Metal Oxide Logic CMOS |
|---|---|---|---|---|
| Average | 1.2V | 100 mV | 7.5V | 2.2V |
| Minimum | 400 mV | — | 5V | 1.5V |

FIGURE 3.1-2. NOISE MARGIN

Individual pulses, measured by their energy content, require anywhere from 10 mJ to 1 $\mu$J to cause damage as shown in figure 3.1-1. Pulse durations are microseconds to nanoseconds. Awareness of these levels is important when designing the shielding for the aircraft or avionics wiring to protect against HF transients or resonances.

Shielding protection or voltage limiting must be provided to reduce transient noise signals below the damage threshold. Protection must also be designed to keep continuous wave radio frequencies below the TTL or integrated circuit operational thresholds (figure 3.1-2). Software can be designed to correct for random, nonrepetitious transients.

Extensive effort in testing, modeling, measuring, and graphing operational upset and damage levels has been documented in the industry. Minimum and maximum spreads are available. Voltage damage amplitudes may vary by a factor of two or three, say from 100V to 300V from one manufacturer to another, or from one lot to another. The threshold of operation may vary from 800 mV to 1.5V from one transistor to another. Damage thresholds change by a factor of ten or more with the number of transient pulses and the rate of applying them. The spreads and averages are interesting in the study of susceptibilities, but at the bottom line is this: protection must be built in to account for the minimums - for example, 100V for transient damage, 800 mV or less for TTL gate state change, and 10 $\mu$W for microcircuit performance alteration. It is necessary to constrain radio frequency power and stored transient energy access to avionics by diverting and/or blocking the noise with balanced circuits, filters, or shields on every input-output interface circuit.

When two conductors are spaced 100 mils (2.5 mm, 2500 $\mu$m) apart, and the voltage between them is slowly increased, an arc will start (at sea level, atmospheric pressure) and establish itself at around 2000V or 3000V. If there are only 20 mils of spacing (0.5 mm, 500 $\mu$m) between the conductors, the arc will start at 1000V. Circuit card conductors have these close spacings. So it is easy to understand when microcircuits, chips, and thin-film devices with substrate circuit separations of a few microns fail at 100V.

## 3.2. EMI and Software

It is difficult to know where to start in the treatment of electromagnetic interference relative to software in an aircraft context. What is special about an aircraft? How does aircraft electromagnetic interference uniquely relate to software?

Not by component failure or damage: damage may appear anywhere, anytime - broken parts - vibration - faults - mishandling.

Not by errors or deficiencies in software itself: this is the purview of the software designer; he must compensate for these regardless of aircraft electromagnetic interference. And also, not especially by the internally generated noise from power and switching regulators, or clocks and data lines: noise sources found in any electronic package.

The aircraft associated electromagnetic interference arises from power line 400 Hz, radio frequencies, electrical transients, and power bus momentary interruptions. They are very different in their characteristics, their occurrence, and their threat of upset. Two of these noise threats must be eliminated from software concerns right away.

First, 400 Hz is simply 400 cycles per second of a noise voltage from the power line imposed on a neighboring circuit. Cycle duration time is 2.5 ms; with zero to peak being 625 us; the positive risetime repeats every 2.5 ms. If a balanced circuit is slightly unbalanced and responsive for any reason, the 400 Hz will trigger operation of the balanced circuit continually or, much worse, in an intermittent fashion. If there is an error or disorder from 400 Hz, then there has been an error in the original design that must be fixed. Four-hundred Hertz must be controlled and kept out of interface circuits.

And second, radio frequencies can also be eliminated from software concerns. Aircraft wiring cannot be allowed in a radio frequency field stronger than the original design specification. If a digital circuit sees an overlay of an unwanted radio frequency (not damaging, but causing loss of data), the software will not be able to correct upset. The radio frequency environment must be known, documented, and immunity designed in to the interface circuits. Transmitter radio frequency noise and digital data are in the same frequency range, the most important being HF-VHF, 1 to 300 MHz.

Transients are the problem. Electrical transients have existed in the past, they exist today, and will continue to exist on the airplane as well as in the laboratory. Electrostatic discharge, lightning, or powerline inductive switching transients are usually of large magnitude (hundreds to thousands of volts). The amplitudes are reduced below circuit damage level (100V; a few amps) by design; but transients are sometimes not totally rejected, and they result in short-term destruction of data words. It is the responsibility of the EMC engineer to supply threat transient levels and repetition rates to hardware/software designers to assure proper programming of fault tolerance and detection. What, then, are the time characteristics of transients relative to data words that the software designer must know to override noise or make software tolerant to noise?

Some inductive switching transients have ringing frequencies of 10 MHz recurring at a 1-MHz repetition rate and lasting for around 1000 $\mu$s (figure 3.2-1). During test, the transient is repeated every two seconds. (DO-160 specifies 8-10 pulses per second for 10 seconds.) A 12-kHz, ARINC 429 signal has a bit width of 80 $\mu$s, and the 32-bit word has a duration of 256 $\mu$s. A 1000 $\mu$s, inductive switching transient can decimate a 256 $\mu$s data word.

An electrostatic discharge, on the other hand, is a very fast 100-ns event (figure 3.2-1) and probably does not recur until after sufficient time, possibly 4 or 5 sec, to recharge the object originally collecting the charge. This transient might only affect one bit.

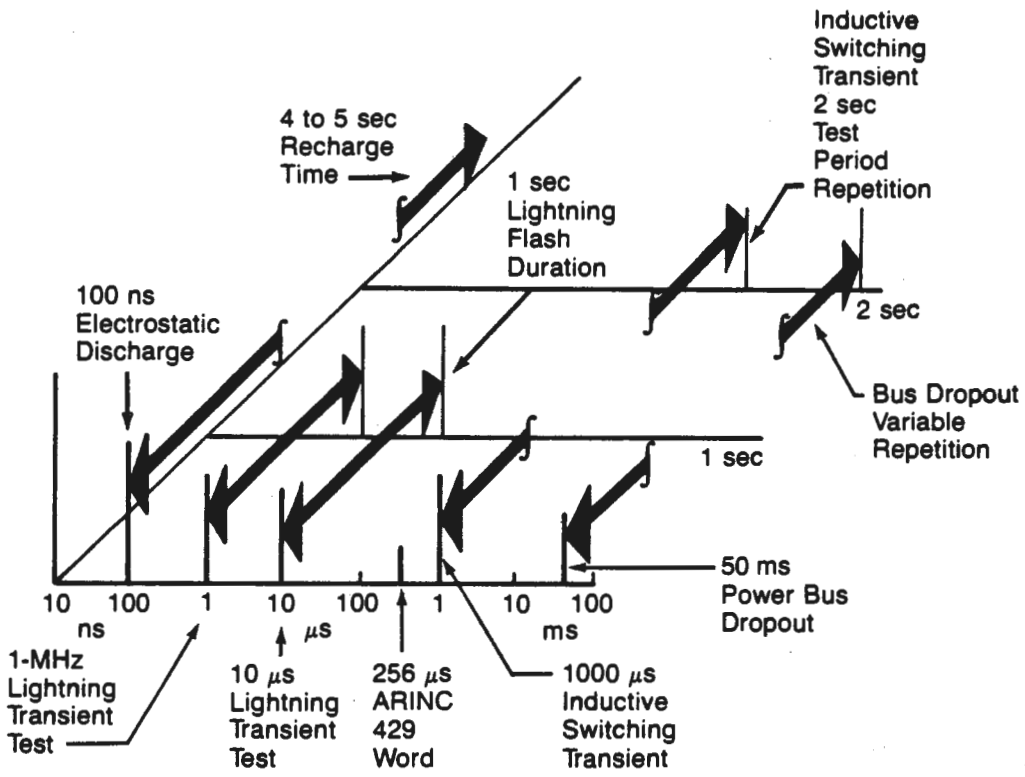



FIGURE 3.2-1. TRANSIENT EVENTS

Lightning transients have low frequencies, 10 μs or longer, and ringing high frequencies, for instance at 1 MHz, 3 MHz, and 7 MHz, set up by the electrical resonant lengths of an aircraft and damped out in a few microseconds; but then they may recur again and again under multiple strokes of a total lightning flash lasting for possibly one second (figure 3.2-1). A lightning flash might result in disorder in a number of words.

Of important concern is the momentary interruption of the power bus, referred to as a "bus switching" or "dropout" transient, where power drops or decays to zero for up to 50 ms (figure 3.2-1) when the power supply is transferred from ground power to engine power or from one engine generator to another. (DO-160 has a test requirement of a 200 ms interrupt for ac equipment, and a 1-second interrupt for dc equipment.) Software control may be required to ensure a graceful shutdown, proper data storage, or even continued processing.

So the software designer institutes techniques to control equipment operation, override errors, and make software tolerant to noise when transients and power shutdown occur.

There are a number of methods employed for data correction and resetting, such as:

- Microprocessor reset to initial state ("backward")
- Microprocessor forced to known state ("forward")
- Functionally equivalent, but dissimilar backup system
- Data word repetition or redundant data supply

There are also a number of means of detecting errors of data, flow, or hardware operation, such as:

For data checks -

- Read after write on output data line
- Data bus activity, reasonableness check, data averaging
- Bits per word, parity, status bits
- Check sums: averages, spreads, maximums, minimums

For flow checks -

- Out of sequence, out of loop
- Excess or deficient time
- Event record of activity with respect to total program execution ("state activity")

It may be postulated that the aircraft has five layers of protection: (1) structure shielding, (2) circuit shielding, (3) balanced circuit, (4) voltage/current limiting, and (5) software. Software is the last line of defense.

## 3.3. Digital and Discrete Circuits

### 3.3.1. ARINC 429 Drivers and Receivers

The ARINC 429 bus, a digital transmission interface system, fans out from the main equipment bay to the flight deck and to external sections in the wing, engine, or empennage. With two or more 429 circuits per unit, there could be two- to three-hundred individual buses. They may reach 100 ft (30m) in length, and are routed in bundles where they encounter transients, radio frequencies, and power line noise. The ARINC 429 "MARK 33" Digital Information Transfer System (DITS) offers immunity to noise by using a well designed combination of a balanced circuit, a relatively high-trigger threshold, a "high-resistance" input, and finally, wire braid shielding. Parity, status bit, and "bits per word" software checks help to extend protection even further.

The output signal of the 429 transmitter measures at the high level, 10 ±1V "line-to-line" and at the low level, 0 ±0.5V. At the other end of the line, the receiver must operate with an input signal at a high level of 6.5V to 13V and be at a low below 2.5V. The margin from 2.5V to 6.5V is undefined.

The trigger threshold of operation, therefore, can be 2.5V.

The transmitter driver output resistance is 70Ω to 80Ω "line-to-line" and the receiver input resistance is 12 KΩ or greater in each line so that there is at least 12 KΩ resistance in the circuit from transmitter to receiver. That is an important resistance for protection against transients. The high resistance provides immunity. Just ignoring the shield for a moment, if a 600V transient occurs on the wire and appears at the 12-KΩ input, the resulting current amounts to only 50 mA, a very low "transient" current and not enough energy for damage.

The ARINC specification does not define the circuit ground or case ground. For shields, the specification states that: "The circuit should be twisted pair shielded from data source to sink with the shield grounded at both ends at an aircraft ground close to the rack connector." Shield tie length is not defined. The 429 system has been subjected to the RTCA DO-160 electromagnetic interference tests and passed. Three factors in implementing this digital bus design demand extra effort in manufacture to ensure quality: (1) electrical tolerances of components (avionics supplier responsibility), (2) shield tie to ground (airframe manufacturer and supplier), and (3) case ground (airframe manufacturer and supplier). Future digital buses may have similar noise characteristics and rejection capabilities.

The 429 is a balanced circuit, and balanced circuits are important; the installation of shields is also important, but grounding decides effectiveness of receiver immunity: the grounding of the box, the grounding of the circuit, and the grounding of the shields.

### 3.3.2. Circuit and Shield Grounds

Noise on a single wire in space with no connection to a ground plane cannot be measured, and no current flows for an "electrically short" wire.

Connect one end to a ground plane and an induced noise voltage in the wire of 1V can be measured at the other end, the ungrounded end. This is a single wire over ground (figure 3.3-1a) and is the technique used to install the "discrete" circuits on the aircraft. It is susceptible over the entire frequency range.

If the ungrounded end is left unconnected, practically no current flows and no energy is dissipated. That's an incredibly significant fact when analyzing for protection against damage. If no current flows, components cannot be damaged. Where circuits are exposed to high-level transients on the wing, isolate them at one end if possible.

Now, two wires over a ground plane with resistors between them at each end to form a circuit and one end connected to ground, say the source end, sets up the same condition - all the voltage will be measured at the load end relative to ground, with practically no voltage across the load resistor at low frequencies.

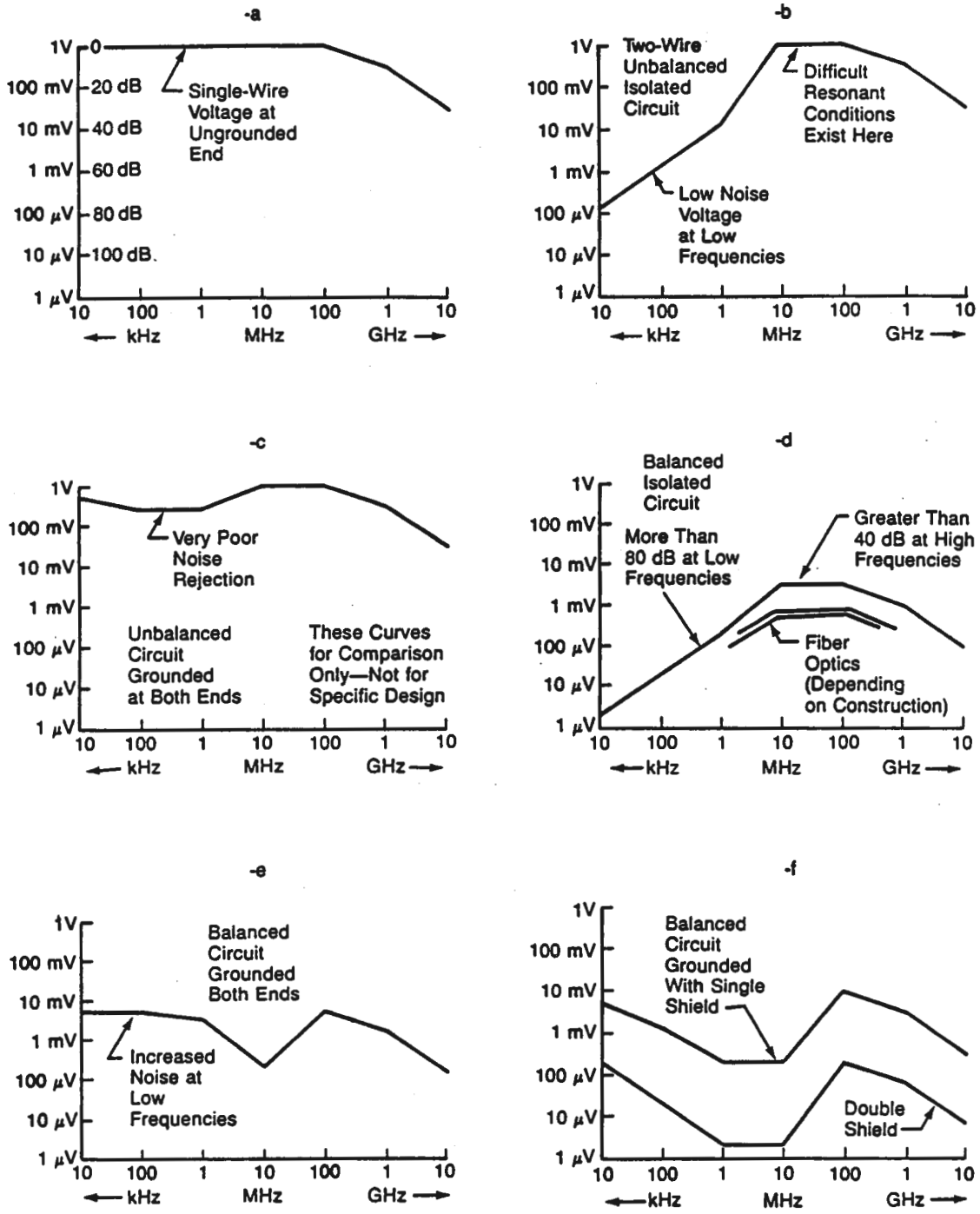



FIGURE 3.3-1. CIRCUIT NOISE REJECTION (A B C D E F)

This is a two-wire, unbalanced circuit (figure 3.3-1b). At the higher frequencies, resonance exists and voltages appear.

A two-wire, unbalanced circuit grounded at both ends, source and load, offers practically no noise rejection, maybe 10 dB or so (figure 3.3-1c). The noise rejection is lost at low frequencies.

Make the circuit a balanced circuit, as the 429 is, and it can establish more than 40 dB of noise rejection for "differential mode" line-to-line voltages (figure 3.3-1d) at the higher frequencies. The line-to-ground noise, called "common mode," remains the same as the unbalanced circuit.

In an aircraft installation, it is optional whether or not the balanced circuit is grounded. Grounding loses some of the low frequency protection (figure 3.3-1e). The balanced, isolated design can offer 80 to 100 dB at low frequencies, just about equivalent to fiber optics or transformer isolation under practical installation conditions.

Figure 3.3-1f shows a comparison of one shield (like 429) and two shields grounded at both ends with the circuit and equipment case grounded. If either the circuit or case were lifted from ground, the circuit would have much more low frequency isolation and protection.

A balanced circuit (either grounded or isolated) that is double shielded with shields tied to the case connector and the base surface of the enclosure case grounded is almost always the best. The circuit can be isolated with a transformer, optical device, or fiber optics. This is the optimum practical design for low and high frequencies.

### 3.3.3. Discretes

A "discrete" circuit is designed and installed to indicate events, such as on or off, engaged or open. Sometimes it is constructed using a single wire (figure 3.3-1a) from the electronics bay out to a unit on the wing where a switch, to provide indication, grounds it to structure using the structure as the return. Their thresholds are high and, therefore, radio frequency noise is usually not a problem. But these circuits can be hit with the full force of transients and need to be protected with shielding, voltage or current limiting, or increased power ratings.

## 3.4. Equipment/Wiring Isolation and Separation

### 3.4.1. Quality of Wiring Design

If there are no wires, there is no electromagnetic interference. Wiring takes on different characters in its role as a conductor of signals. It is a signal conductor, driver to receiver. It acts as a transmitting radiating antenna or a very efficient receiving antenna; it is a party in transformer action to participate in voltage-current-energy transfer when in a cable bundle; and it also acts as an intermediary or a transfer agent to import electrical energy, then retransfer that energy to other wiring - sometimes called secondary coupling.

The wiring design is a fabric, a multi-weave electrical mosaic. There is a mosaic of conditions: wire size, grounding, returns, shields, shield ties, separation, and wire length. There is a mosaic of properties: metals, dielectrics, resistance, inductance, capacitance, impedance, and nonlinear effects. Careful analysis and development of the design will bring about an acceptable level of electromagnetic interference and a cost-effective electromagnetic compatibility.

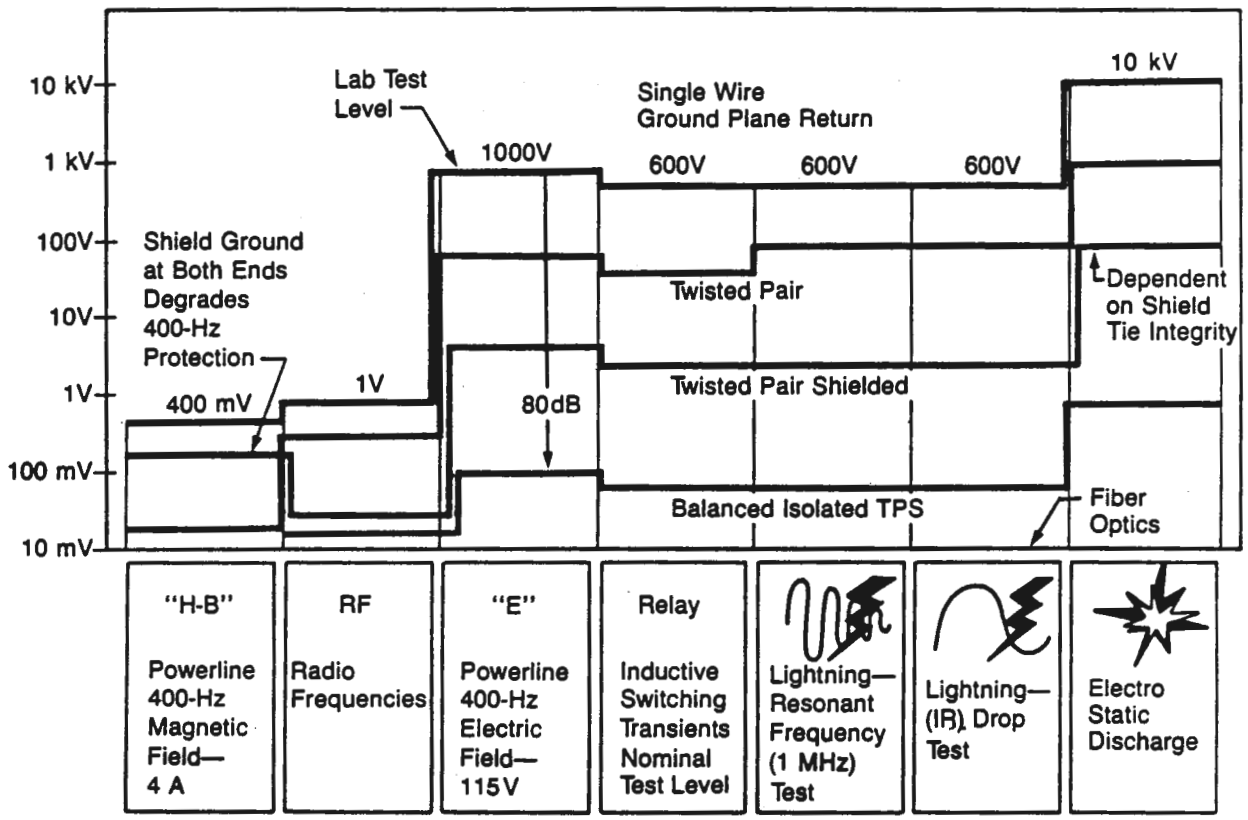



FIGURE 3.4-1. CIRCUIT RESPONSE TO EMI

The complexities of the mosaic could be largely dispelled through the simple use of a twisted-pair-shielded (or coax), balanced-isolated circuit design that reduces noise, and sustains signal quality. See in figure 3.4-1 how the twisted pair, shielded, balanced circuit has reduced noise to an acceptable level. This design provides a winning combination. It is the king of avionic interface design, boasting a possible noise rejection of 80 dB (10,000 times) and finding widespread use in digital communication or control circuits where stability, quality, and signal fidelity guarantee performance and confidence. Figure 3.4-1 was constructed to illustrate the general levels of improvement for comparison, but is not appropriate to be used or adapted to specific designs.

The building blocks to signal wiring quality are threefold: twisted pair for minimum magnetic field induction, down by 60 dB; shield for minimum electric fields, down by 30 dB or more; and balanced isolated design for a minimum common impedance, down by 40 dB. For future aircraft, it will be necessary to accurately model types of circuits on an aircraft to understand the induced noise levels and assure desired signal quality. Determination of the actual voltage, current, power, or energy on the wire circuit is the goal. Current and power become important when designing the rating or sizing of circuit protection.

There is no easy answer to avionic interface wiring design. Interface circuits can be built (by careful engineering evaluation, assessment, and construction) to offer optimum circuit compatibility through four fundamental approaches:

- Equipment and wiring location/grouping/organization design.
- Connector choice and wiring assignment.
- Return current rule.
- Coupling and modeling analysis.

Definition of the total circuit is required: the driver, the wiring, the receiver, operating frequency, and intercircuit connections. The following tasks are applicable: identification of all identical circuits; identification of all common circuits, common returns, every ground connection, capacitor, or resistor connection; determination of output and input impedances, balanced and unbalanced impedances to ground. A check of other current paths that are not intentionally designed is often appropriate, such as, circuit paths from ground wires through mutual capacitances (leakage capacitance), through other ground conductors or transformers, shields, and circuit card grounds.

Identical or common circuits are best assigned to the same connector. Circuit currents should exit and return in the same connector. Low-frequency signals, less than 1 kHz, may return in a well-signed aluminum or copper grounding structure where transmission line design techniques are not necessary.

These rules are constructive:

- Separate power and signal (if power must be in the same connector as signal, separate power and signal by ground pins).
- Separate families of circuits in individual connectors by frequency: audio, digital, pulse width modulation (PWM), video.
- Position wires for shortest practical route to other equipment.
- Assign connectors for optimum wire routing to other equipment.

Current paths returning in structure are designed to follow immediately adjacent to the cable (an image path). Currents should not be forced to take a wide path

through distant connectors and structure. This is sometimes called the return current rule.

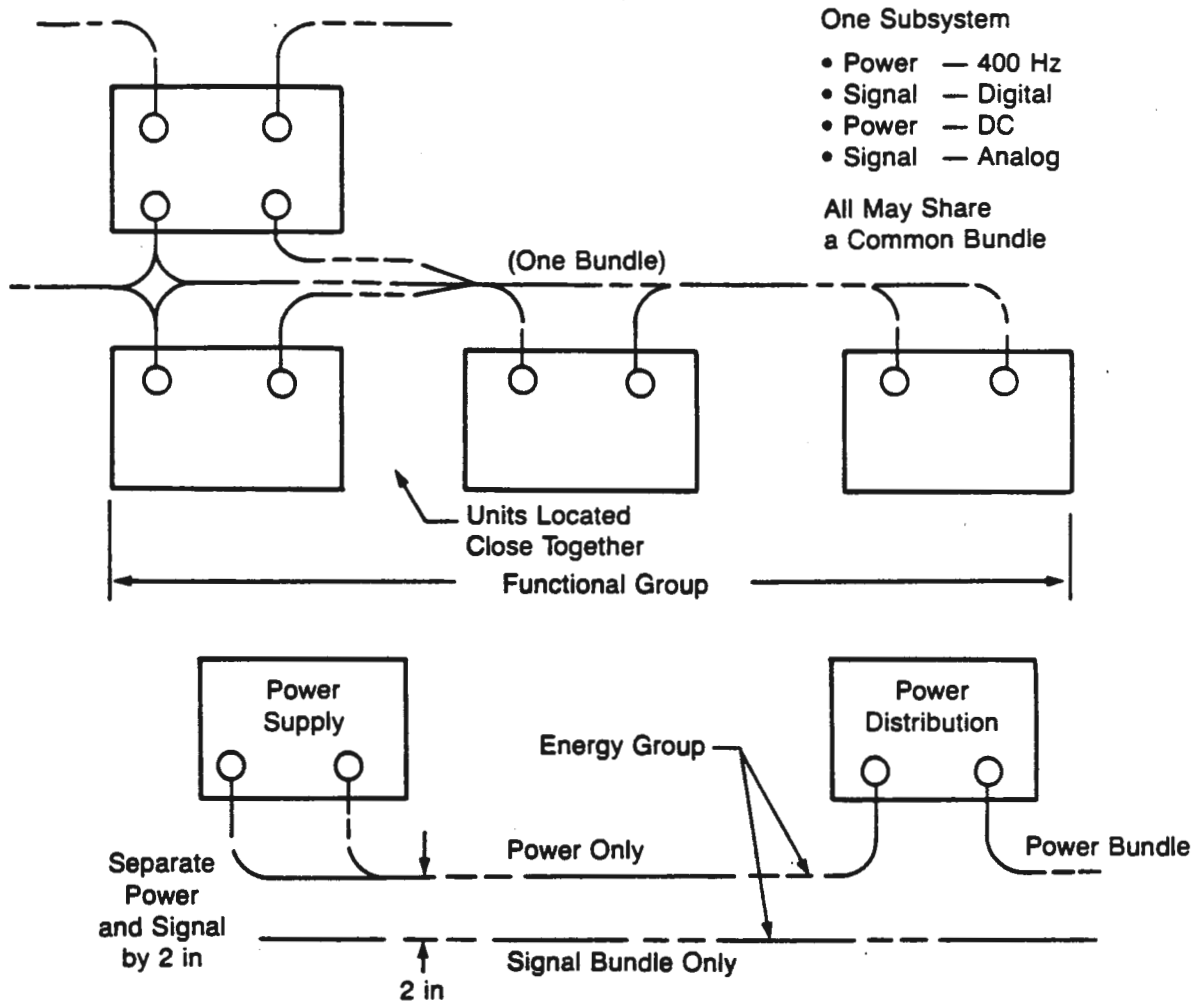



FIGURE 3.4-2. WIRING CATEGORIES

A simple way to achieve electromagnetic compatibility in wiring or in equipment is by functional or subsystem grouping (figure 3.4-2) - that is, keeping units or equipment of the same subsystem close together. An early definition and visibility of the system design, configuration, and function must be obtained. Equipment and subsystems and their locations must be known:

- Primary power, secondary power, distribution boxes, heavy switching (solenoids), lighting.
- Electronic flight control, electronic flight instrumentation, engine electronic controllers.

- Navigation, VOR, ILS, and DME.

- Radio transmitter/receivers, HF, VHF.

An early developmental wiring mockup is a requirement for good wiring design. The best design is grouping the subsystems close together in an in-line design. Generator to receiver, input to output, and source to load. Individual connectors can be dedicated with common groups of circuits. Functional grouping can often be extended in the aircraft between units that are widely separated. Here, engineering judgment and analysis is important. Where functional grouping is used, bundles are more easily separated to comply with redundancy and safety requirements or a separation of wiring to protect against physical damage to redundant critical equipment such as engine-mounted electronic controllers.

On long wire runs, functional grouping may result in excessive coupling between noisy lines and susceptible or sensitive lines. Where space is available, it is recommended that wiring be separated by "signal/energy" categories as follows:

- Ac feeder power bus; large ac control circuits; heavy current dc or secondary ac switching circuits valves, motor, or actuator drives; large inductive loads.

- Standard 115V and 28V power; signal circuits and regulated power circuits.

- Low-level, sensitive circuits such as audio, analog, dc reference, or dc secondary power.

- High-power radio frequency circuits (coax) or high-level pulse width modulated (PWM) circuits.

Wire separation is not the best way to reduce noise between circuits, but it is often necessary. Separation is very effective when used for isolation, redundancy, and safety (figure 3.4-3). Independent computer controls, instruments, power sources, or standby instrumentation are isolated to increase reliability. Left and right engine power lines and circuits, as well as engine indication signals, are separated and isolated. Communication, EICAS systems, ILS, LRRA, DME, and computer control devices, FMC, ADC, are separated and isolated. Primary controls (for example, roll, pitch, yaw, and stabilizer or spoiler controls as well as position sensors) are isolated.

### 3.4.2. Power and Energy Levels

A twisted pair shielded, balanced, isolated circuit, properly designed into the system as an interface circuit, minimizes the possibilities of noise-induced damage and establishes a basic high-quality design. But not all circuits are balanced and use a return wire; some interface circuits on the airplane have a ground return (structure return).

It is helpful and instructive in the initial design stage of a program to have a concept of the magnitudes of not only the noise voltage amplitudes but also the power and energy levels of radio frequency signals, 400-Hz power, and

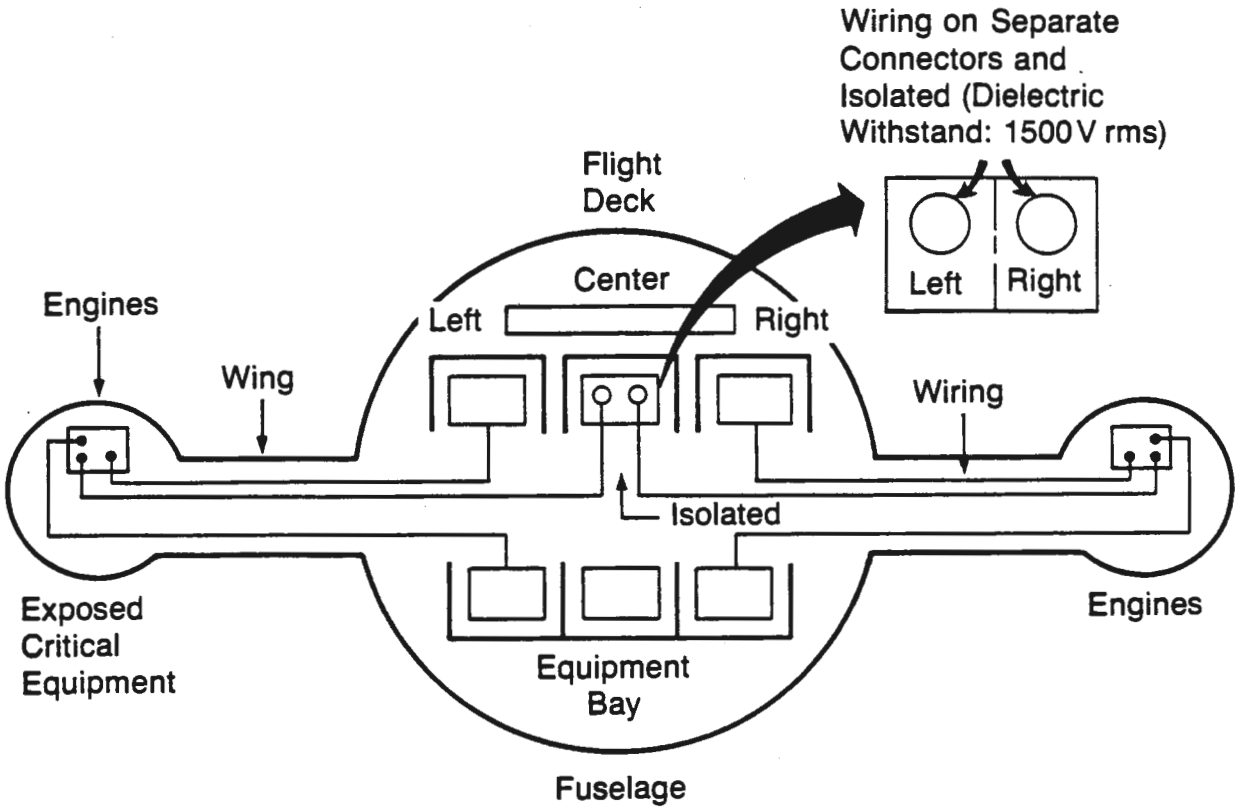



FIGURE 3.4-3. CRITICAL CIRCUITS WIRE SEPARATION

transient noise that can be imposed on a circuit which uses structure as the return path. It is important to analyze and define the power level (for continuous signals) and the energy levels (for transients).

One way to do this is by looking at selected electromagnetic interference laboratory test levels (figure 3.4-5). The test levels shown here are the avionic equipment test levels seen by equipment itself and are not the actual environmental magnitudes that may exist on the aircraft that the airframe manufacturer must consider, such as, higher levels of radio frequency signals or lightning transients.

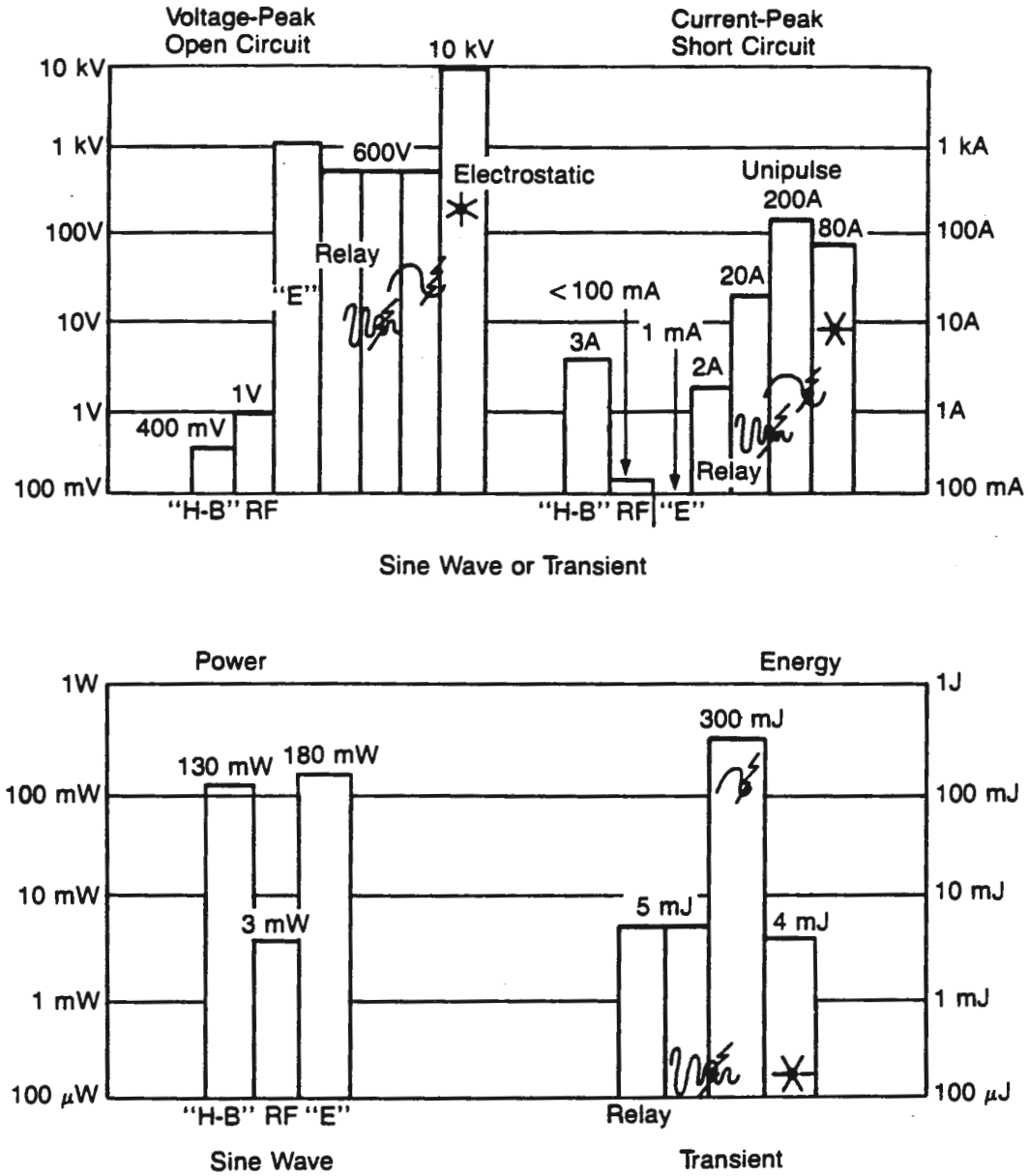



FIGURE 3.4-4. SELECTED EMI LEVELS

Seven electromagnetic interference types can be listed for the purposes of identifying, summarizing, and comparing their voltage-current-energy and power profiles. In these selected laboratory test setups, certain circuit parameters and conditions must be arbitrarily defined. Therefore, the levels shown in the figure are illustrative and helpful for making comparisons, but are not appropriate for detail design conditions and must not be used or adapted to specific designs. The electrostatic discharge test is not an industry standard

test level and is shown here for comparison to the other forms of noise. The amplitude is arbitrarily set at 10 kV, a nominal value out of an actual range of about 3 kV to 50 kV. Referring to figures 3.4-4 and 3.4-5:

- The radio frequency test induces a low voltage, 1V and a low current, approximately 100 mA, on a single wire that has a ground plane return. Observe on the bar chart the radio frequency voltage level of 1V and then the radio frequency current level of less than 100 mA. Now move down to the power bar chart and see the radio frequency power level of 3 mW. The power level is shown because it is a continuous signal. This test is performed in the laboratory using a current probe to induce the radio frequency voltage measured as an open circuit voltage (figure 3.4-5). The test level here is chosen as 1V (in the HF-VHF range) induced into a wire that is 5m (15 ft) long. The calculated current is the short circuit current and the load resistance is adjusted for maximum power.

  These radio frequency signal products come from aircraft microprocessor clock and switching regulator stray noise signals as well as HF-VHF aircraft communication, HF-VHF television, and FM radio broadcast transmitters. Radio frequency carries with it the ability to sneak through avionics internal circuit capacitors or diodes connected to power supplies, and on a circuit card it will pass from circuit trace to trace. The radio frequencies are not damaging, but they can alter circuit operation and performance.

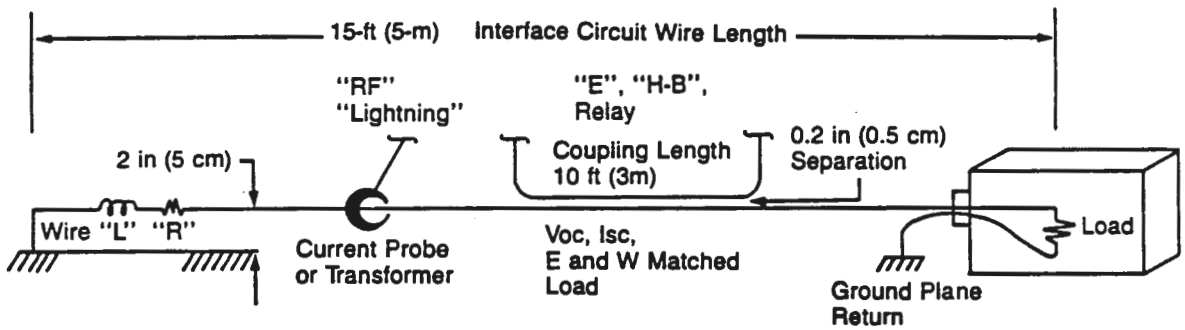



FIGURE 3.4-5. LABORATORY TEST SETUP

- The power line 400-Hz magnetic field, designated as "H-B" on the chart, delivers a low-frequency effect into analog circuits or operational amplifiers. Mark on the chart how the magnetic field might impose 400 mV with a power of 130 mW.

  For this laboratory setup, a 40A current is coupled into a 3m (10 ft) section of wire, with a 0.5-cm (0.2-inch) separation, and positioned 5-cm (2-inch) above the ground plane. This simulates a 4A, 30-m (100A-ft) maximum coupling condition on an aircraft. The power is calculated for a load resistance equal to the 5-m wire resistance.

- The power line 400-Hz electric field ("E") also delivers a low-frequency effect into analog circuits or operational amplifiers. See how the electric field ("E") is about 1000V, but note the extremely low current, less than 1 mA (off the chart). The power level is low but easily large enough for upset or alteration of equipment performance. Notice here that the power line test is performed at 10 times the 115V line level, or 1150V, in order to use a 10-ft coupling length. This simulates a 115V 100-ft maximum coupling condition on an aircraft.

- The electrical switching transient test (relay induced transient) may impose voltages of 600V or higher with currents of just a few amps and energies in the range of millijoules - large enough to damage sensitive solid state devices. For this test, the electrical parameters were estimated with the following values chosen: 600V peak-to-peak voltage; 2A current; a 1-MHz damped sine wave, repeated 1000 times to simulate a transient of 1-ms total duration. This is an arbitrary electrical switching transient for illustration only.

- The 1-MHz lightning-induced transient (designated by the damped sine on the bar chart) is a damped cosinusoid, has a set 600V amplitude, is transformer-induced into the wire, has an initial fast risetime of about 100 ns (3 MHz), and is to simulate lightning resonance on the aircraft. The induced current is high but the energy is fairly low.

- The so-called "ground potential test" (designated by the pulse waveform on the bar chart) is a 600V 10$\mu$s unipulse or "double exponential" waveform to simulate a controlled lightning induced current transient in structure. Observe on the figure the dramatically higher current, 200A, and much higher energy of the unipulse. The unipulse is a very powerful noise source. It carries a considerable amount of energy and can easily damage and destroy electronic components. A balanced circuit design or substantial protection is needed to divert or block this transient. The test setup is different from that shown in the figure. The 600V waveform from the transient source is pre-established on a 5$\Omega$ load, then the voltage is applied with a "source" transformer connected between ground and the avionics case, housing (and any signal return) where the resulting waveform will vary depending on avionic equipment circuit loading and wiring design. Any circuit using structure as return will receive the full impact of this transient.

- The profile of the electrostatic discharge transient (designated by the "star" on the bar chart) shows an exceedingly high-voltage pulse of 10 kV, which is capable of causing dielectric breakdown of insulation rather than the usual condition of thermal burnout of a semiconductor or microcircuit. The short circuit current can be high but the width or duration of the pulse is so narrow or short that it results in a low energy level. The application is different from that shown in the test setup figure. The transient is applied directly to the circuit wire or connector pin. These electrical parameters apply: 150-pF, 50$\Omega$ source through a 1-$\mu$H, 1-m wire to a simulated 50$\Omega$ load for maximum power transfer.

### 3.4.3. EMC Quality in Maintenance

Structural shielding, wire shielding, and interface circuit protection must be maintained through the life of the aircraft. Here are some of the most important items:

- Keep the controlled wire routing and the wire separation design intact. The original design of wiring is formulated to ensure that faults will not propagate, that critical functions are redundant, and that electrically noisy circuits are separated from vulnerable circuits.

- Concentrate on maintaining short shield ties to the structure or to "line replaceable unit" (LRU) box if structure is graphite-epoxy. Shield ties or "pigtails" do not usually receive the attention they warrant. Shield tie length is extremely important in the effectiveness and quality of shielding. A shield tie that is short, less than 2 or 3 inches, is highly desirable; longer lengths, 6 inches or more, degrade the entire shield. The best termination is to the connector backshell. Future connectors will have backshells and filter pins that will need maintenance.

- Maintain electrical bonding and grounding quality. Careful surface preparation, proper joining techniques, care of bonding straps, and finally adequate conductive sealing of joints and seams ensure the continuation of the excellent shielding and electrical grounding provided by structure.

- Be aware of electrostatic discharge. Electrostatic discharge is a key intruder in handling and maintenance procedures. Microcircuits may be impaired or destroyed by a pulse from a hand or an item of clothing. The event can go unnoticed. Conductive materials and grounding procedures, along with a training program, will provide techniques for failure prevention.

### 3.4.4. Shielding and Shield Ties

Shields may be single braid; double braid; braid and foil; shields inside an overall bundle shield; solid conduit, tray, or cableway; and aircraft structure.

The shielding effectiveness (SE) of shields is based on materials, dimensions, circuit connections, impedances, and shield tie lengths and has been one of the most elusive electromagnetic compatibility protective defenses to pin down. (A significant issue for future aircraft needing to be addressed is that of the wiring lengths installed during laboratory tests versus the actual aircraft installed length.) Length influences SE and SE can dictate length or design. A single braided shield with a 2-inch shield tie may offer less than 25 dB of protection over the span of 10 kHz to 100 MHz. A long wire, 100 ft (30m), may only show 10 dB above 1 MHz at some resonant frequencies under certain conditions. A solid 360-deg connection to a backshell can improve protection. Conditions that establish the grounding of shields vary, but there are some that need emphasis:

- Ground audio or analog shields at receiver end only.

- Ground digital or wideband signal circuit shields at both ends.

- Ground shields subjected to high frequencies (greater than 50 kHz) at both ends.

- Ground shields that contain or are a barrier to transients at both ends.

- When audio and HF requirements conflict then the circuits and installation must be evaluated. (Solution may be rerouting of wires or double shielding.)

Carrying a shield tie through an avionics unit connector, into the internal wiring harness, and to a circuit card connector is poor wiring practice. A 360-deg, peripheral shield connection to the backshell is the best (figure 3.4-6).

## 3.5. Aircraft Protection Measures

### 3.5.1. Structure Conductivity

Conductivity is the predominant electromagnetic compatibility consideration of materials in the basic steps to a unified aircraft structure design (figure 3.5-1). What are the separate electrical functions so highly dependant on conductivity (figure 3.5-2).

- Electrical stability: A low noise ground reference plane - a stable zero reference foundation for electrical and electronic circuit and shield ties (may have less than 500-mV ground noise). This electrical "ground" embodies structure, shelving, skin, spars, equipment chassis, and possibly uniquely installed grids, sheets, and foil.

- Shielding: Aircraft structure, skin panels - foils, flame spray, plating, paint-shelves, equipment enclosures, and wire shields. Shielding affords a barrier to external and internal radio frequencies, 400-Hz electric fields, and 600V transients.

- Fault path: Structure, skin panels, cable shields, safety wiring (green wire). The engineered fault paths divert currents to assure safety of passengers and personnel, prevent hazardous voltages, avoid ignition of combustibles (fire prevention), and limit equipment failure and upset.

- "Diverter": Structure, skin panels. The aluminum aircraft inherently offers the current control paths and bypass to eliminate shock hazard and damage from electrostatic charge and lightning.

- Signal return and power return: Cost and weight savings accrue through the use of structure as a return instead of the installation of wire.

- Reliability and redundancy: Parts of the aircraft may be employed as a baffle or wall to provide separation of wiring or equipment.

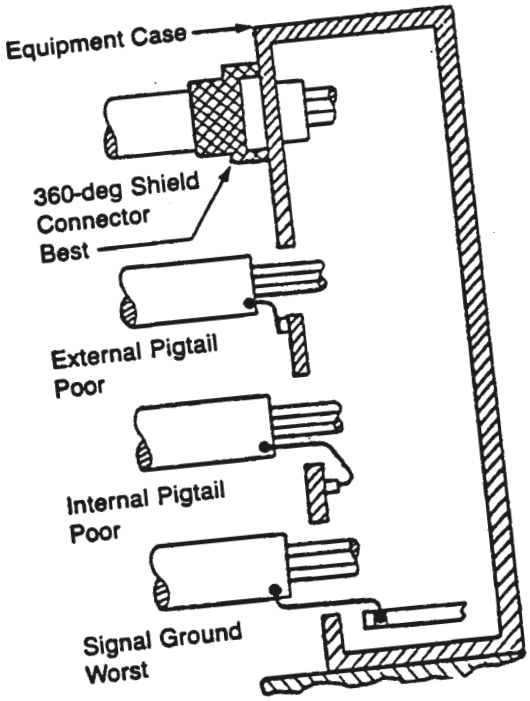



FIGURE 3.4-6. SHIELD TIES

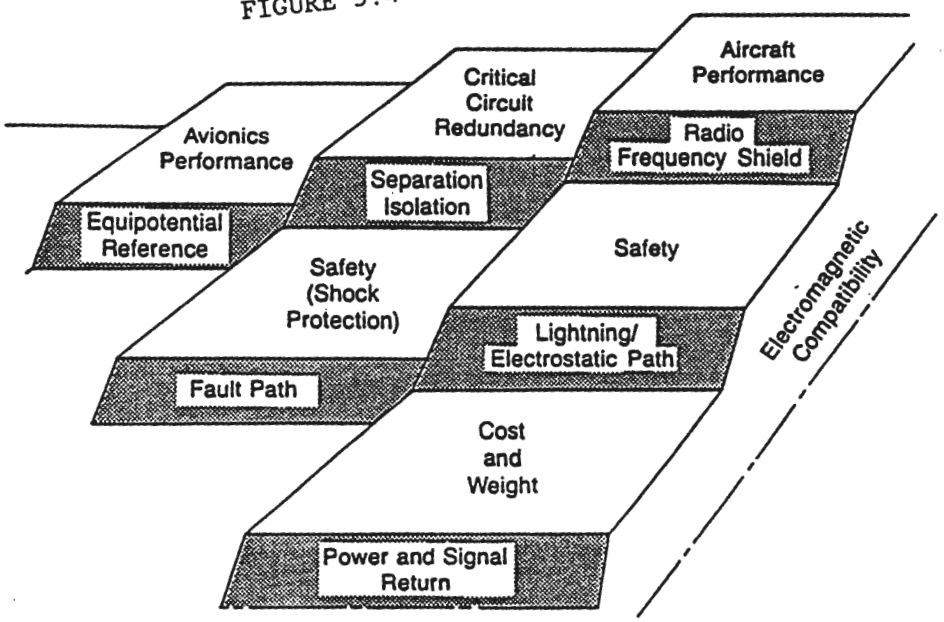
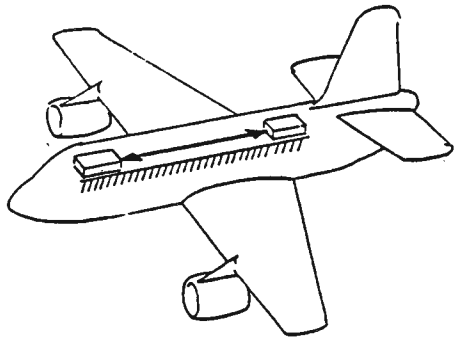
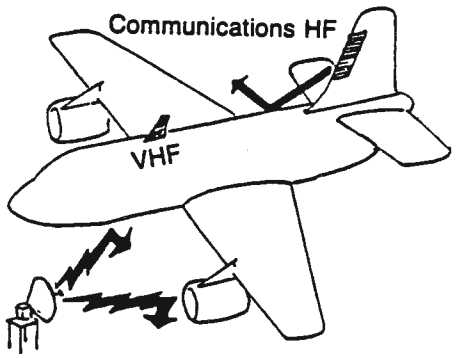


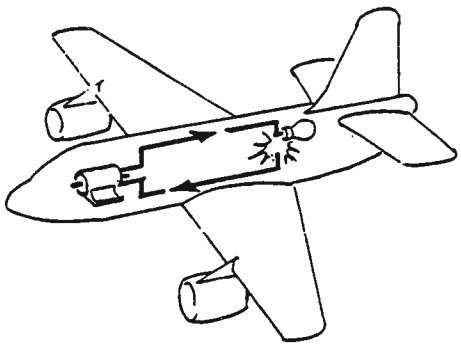

FIGURE 3.5-1. THE GROUNDING STEPS



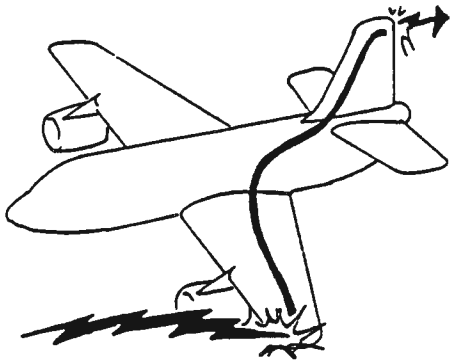
Equipotential Ground Plane
Reference (Stability and
Performance)



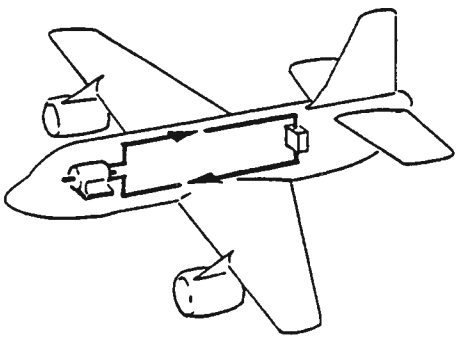

Radio Frequency Shield
- Radar (Critical
- Radio or Equipment
- Television Operation)



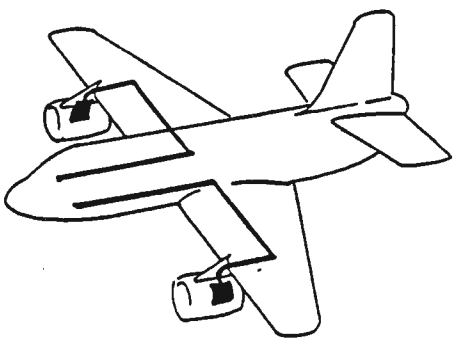
Safety Fault Path
(Passenger and Personnel
Protection)



Lightning Diversion Path
Electrostatic Drain
(Passenger Safety and
Equipment Protection)



Power and Signal
Return Path
(Weight and Cost Reduction)



Separation-Isolation Redundancy
(Critical Equipment Operation)

FIGURE 3.5-2. STRUCTURE EMC ROLES

The engineering of the structure, etc., to accommodate all of the electrical functions, entails detail design of electrical interfaces, bonding straps, foils, paints, etc. (figure 3.5-3). Bonding resistance tests can be made during the aircraft EMC test (see section 7).

It is illustrative and instructive to compare an aircraft with an electronics facility to help recognize the significance of the aluminum structure as a substantive electrical component (figure 3.5-4). Except for framing, many of the facility building materials are not conductive.

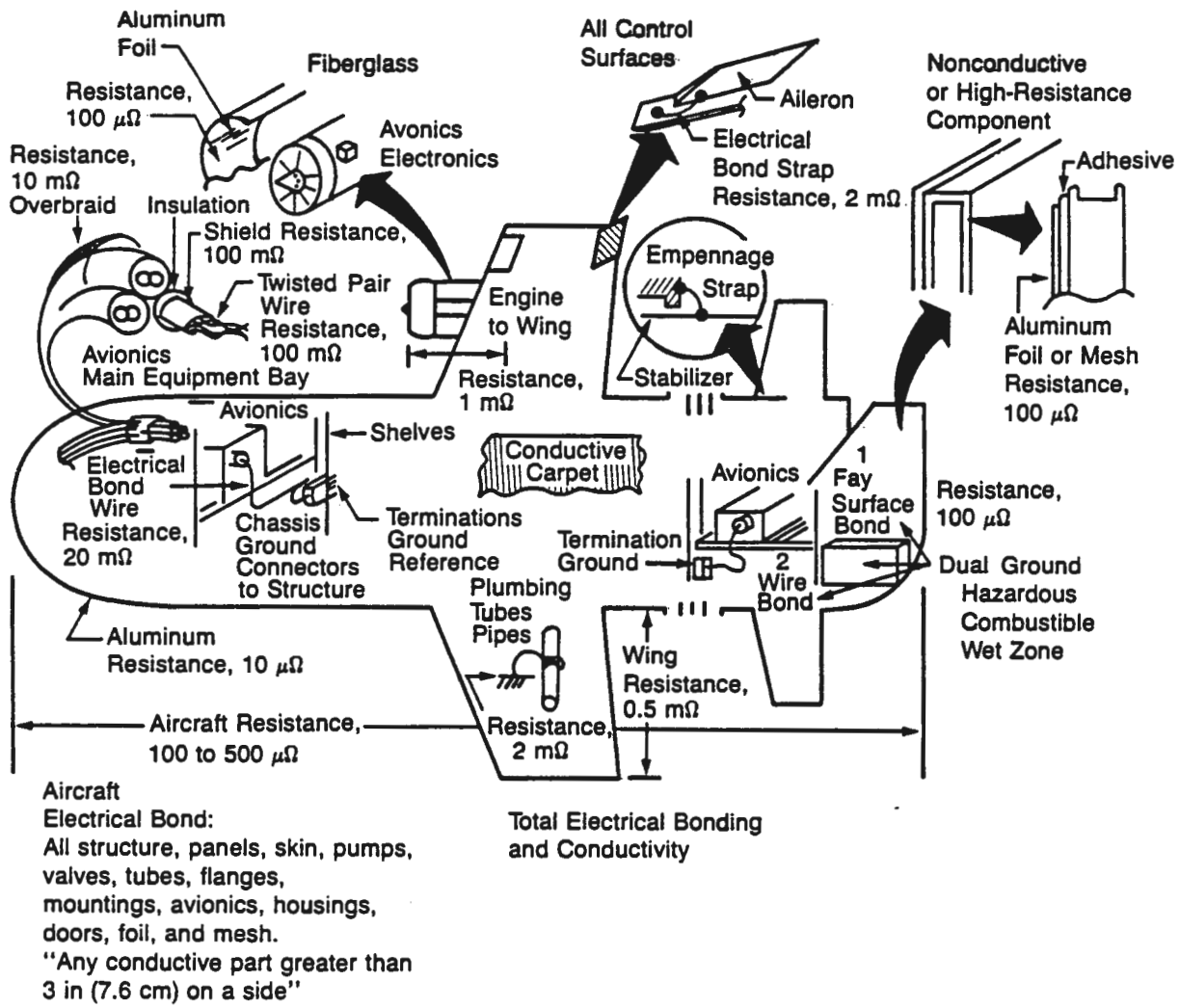



FIGURE 3.5-3. DETAILS AND INSTALLATION

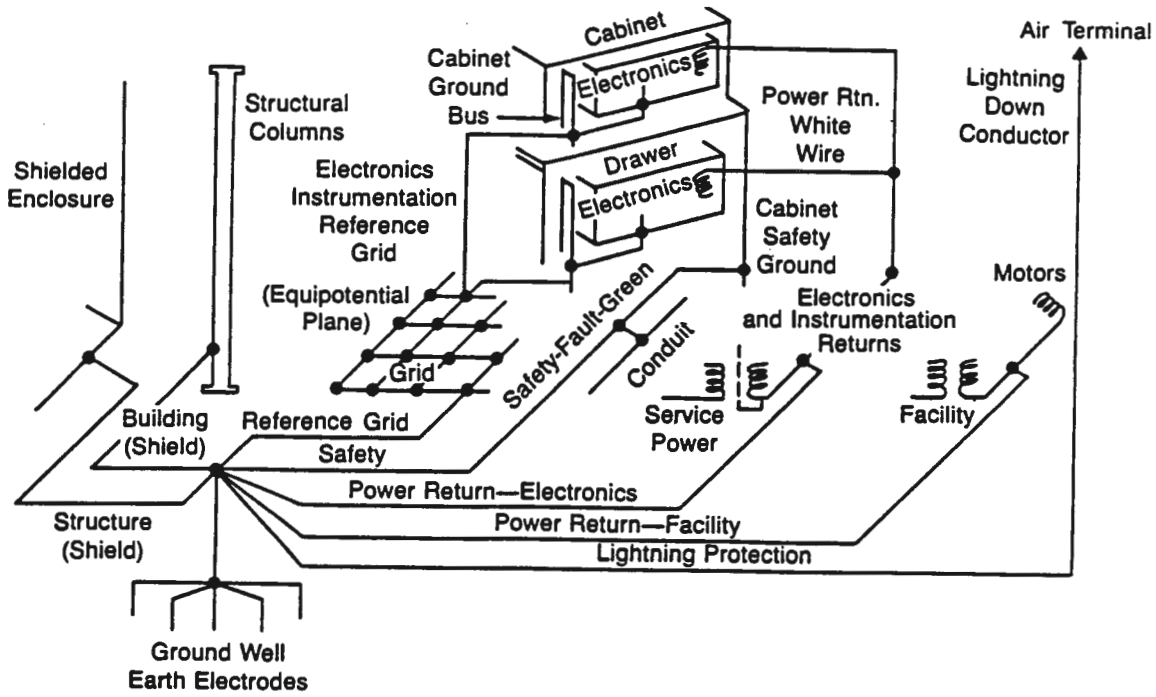


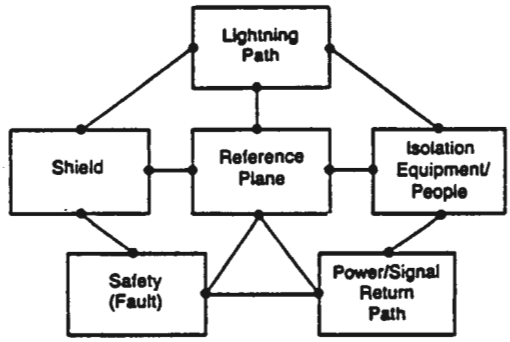

FIGURE 3.5-4. THE SEVEN EARTH GROUND CONNECTIONS

The electrical functions so freely supplied by the airplane structure must be built into a properly constructed facility using extra materials and supports. Whereas the airplane unifies the functions, in the facility they are separate (figure 3.5-5).

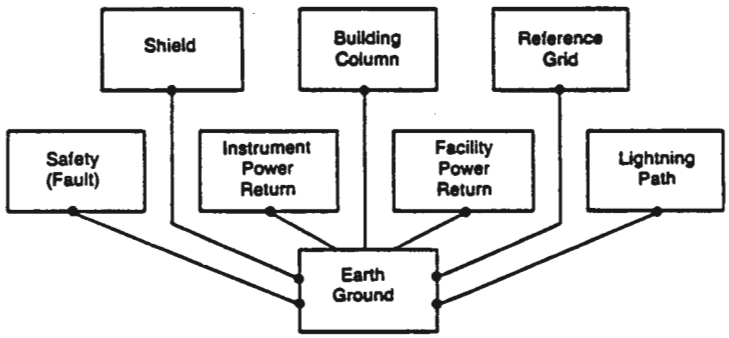
3.5.2. Shielding

Aluminum, after copper, is one of the best electromagnetic shields. Its effectiveness varies with thickness and frequency. One or two thousandths (1 or 2 mils) is good (figure 3.5-6).

Wherever shielding, grounding, or conductive properties are lost at joints and seams (figure 3.5-7) or are not available from structure (therefore compromising continuity), shielding must be added. Aluminum foil, flame spray, plating, and paint are candidate solutions (figure 3.5-8).




A. Aircraft Structure Unified EMC Functions




B. Facility Individual Installations and Connections

FIGURE 3.5-5. EMC ANATOMY BLOCK DIAGRAM

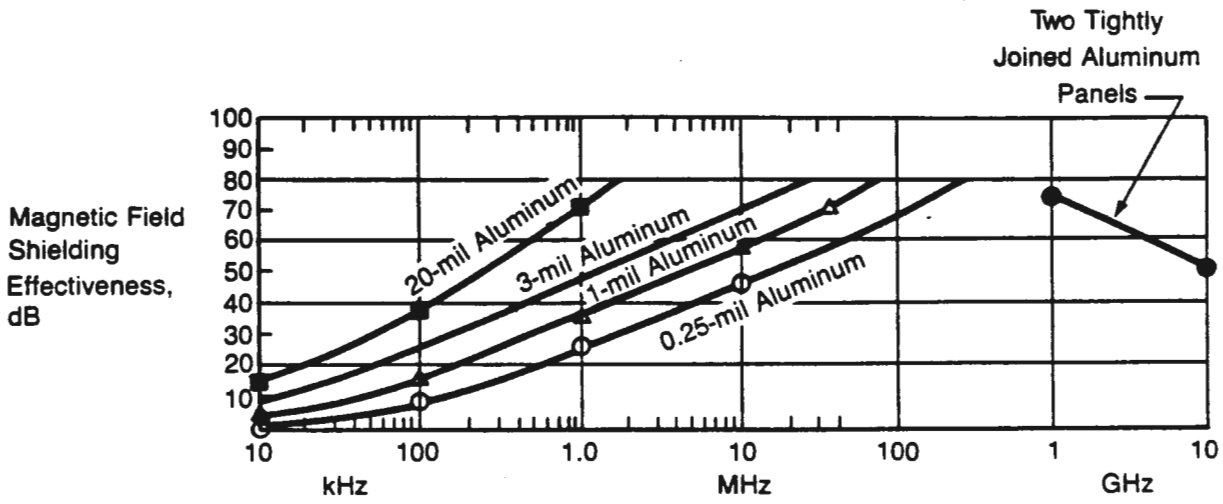



FIGURE 3.5-6. MAGNETIC FIELD SE OF ALUMINUM

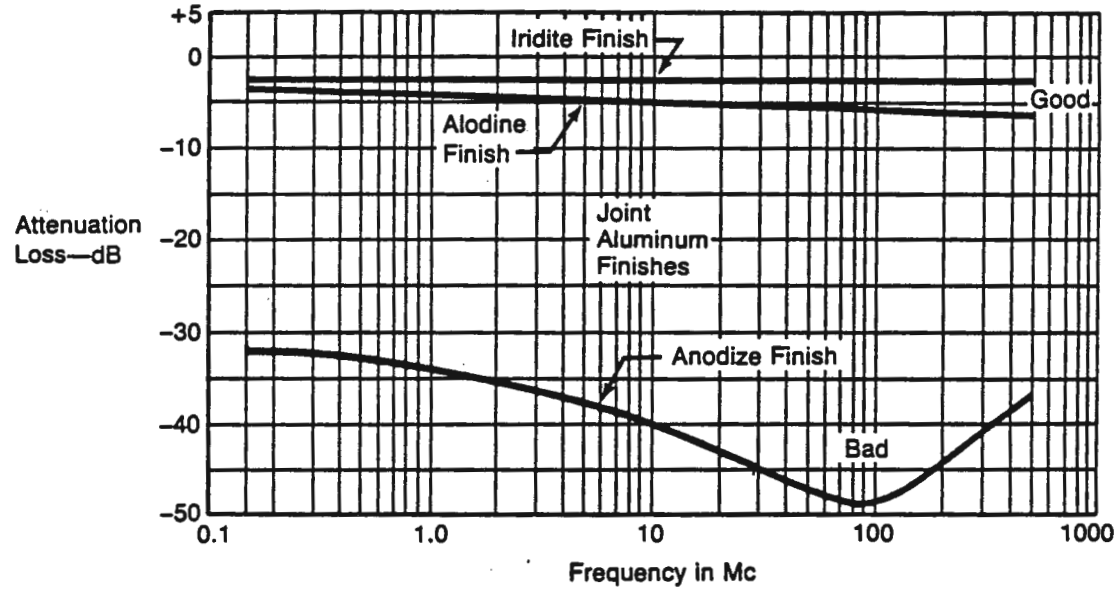



FIGURE 3.5-7. LOSS OF SE WITH JOINT FINISH

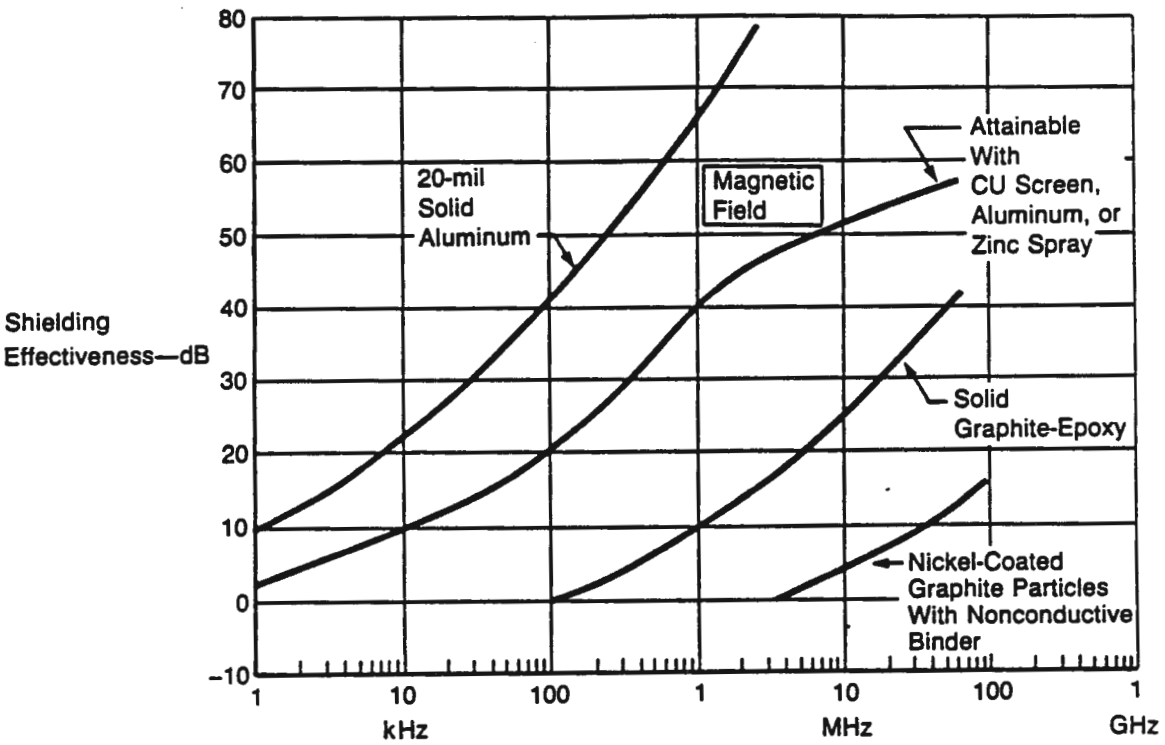



FIGURE 3.5-8. SE COMPARISON

Foil (aluminum or copper) is a good shield, conductor, and reference plane. Foil may bring with it an increased effort to establish quality of bonding to adjacent parts, reliability under environment and vibration, and integrity and durability. Moreover, the foil size, thickness, and geometric configuration dictate its economy of installation. Foil can make a good wire shield.

Flame spray applications (aluminum, copper, or zinc), even by skilled operators, can be difficult to apply in a controlled fashion. The resistance of flame spray coatings is higher and shielding effects lower even with greater thicknesses than foil. Its dense coat and coarse finish on a substrate can lead to flaking or cracking under vibration and moisture conditions with obvious loss of qualities.

Paints (metal or containing conductive agents or fillers) provide good shielding and are used successfully. Insulating or moderately resistive material (graphite-epoxy) may be painted to provide shielding or conductivity or they may be overlaid with aluminum mesh or foil.

Wire shielding provides another layer of protection (figure 3.5-9). Aircraft structural shielding is an important companion to wire shielding. The levels of shielding effectiveness are computed for each wiring run location and configuration to determine the accurate values and these can then be combined with modeled and calculated values of structural shielding. Figure 3.5-10 shows some SE levels for silver or gold film that might be deposited on glass and aluminum or copper screen that could be for shielding ventilation ports on equipment. The front and rear spar areas exhibit very poor shielding. The figures depict nominal or typical estimated levels. Shielding exhibits such wide spreads and variations around these levels (dependant on conditions), and especially as frequency varies that these summary representations are useful as a tutorial tool, but are not appropriate as a design tool.

### 3.5.3. Safety in Grounding and Returns

Structure is a ground and a return. Seven separate grounds or returns on electrical/electronic equipment can be identified. Some of these ground/returns are interconnected and perform common functions. Certain individual grounds are always separated to avert mixing noisy and sensitive circuits.

- Case enclosures or housings (electrical/electronic equipment) ground: May be a wire or the case surface. This case enclosure ground acts as a safety ground (fault return), a static ground, and a radio frequency ground. It sometimes is a signal return and possibly a 115V, 400-Hz return.

- 115V, 400-Hz return: Isolated and brought out of the equipment on a separate wire (figure 3.5-11). A twisted pair power supply circuit will improve electromagnetic compatibility.

- 28V, 400-Hz return.

- 28V, dc return.

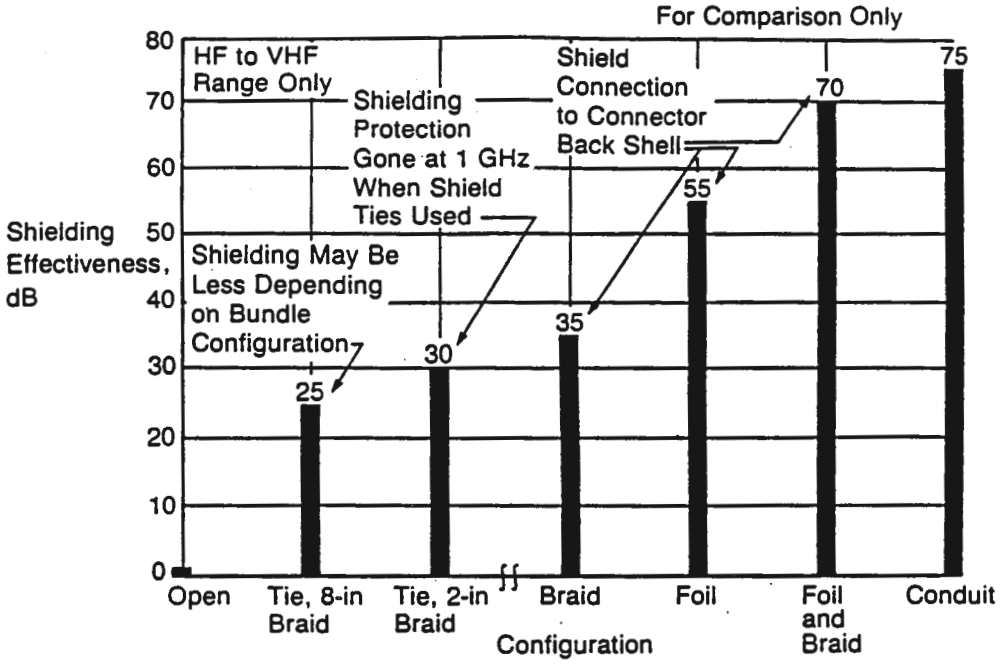



FIGURE 3.5-9. WIRE SE COMPARISON AT HF-VHF

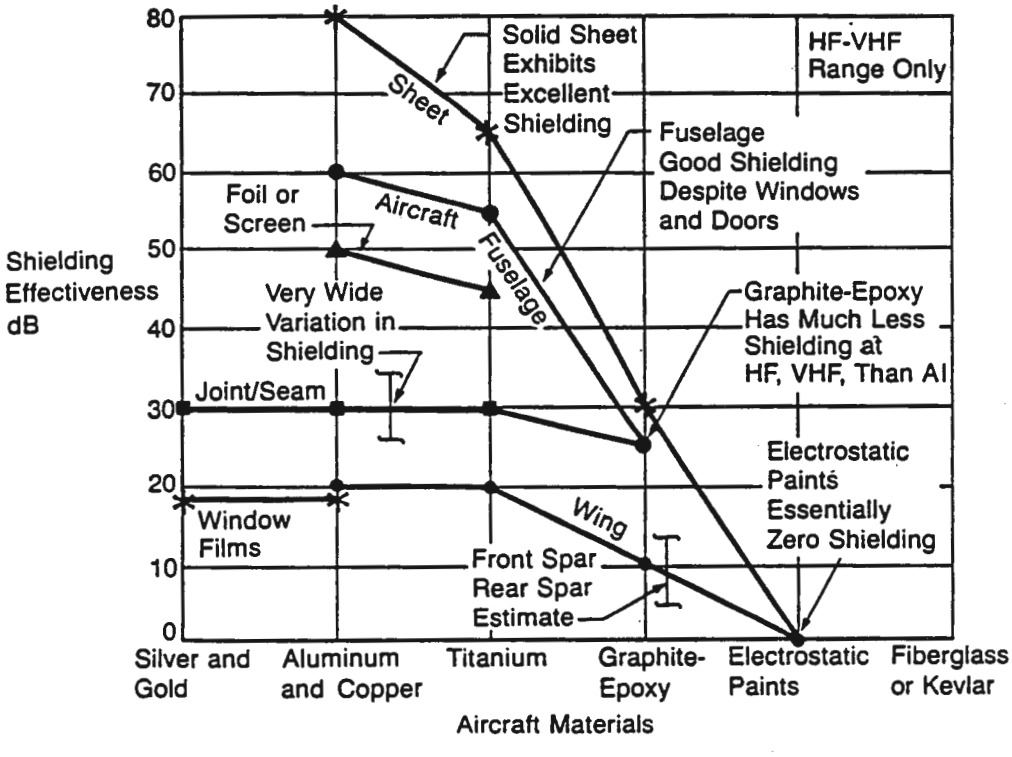



FIGURE 3.5-10. MATERIAL/CONFIGURATION SE OVERVIEW

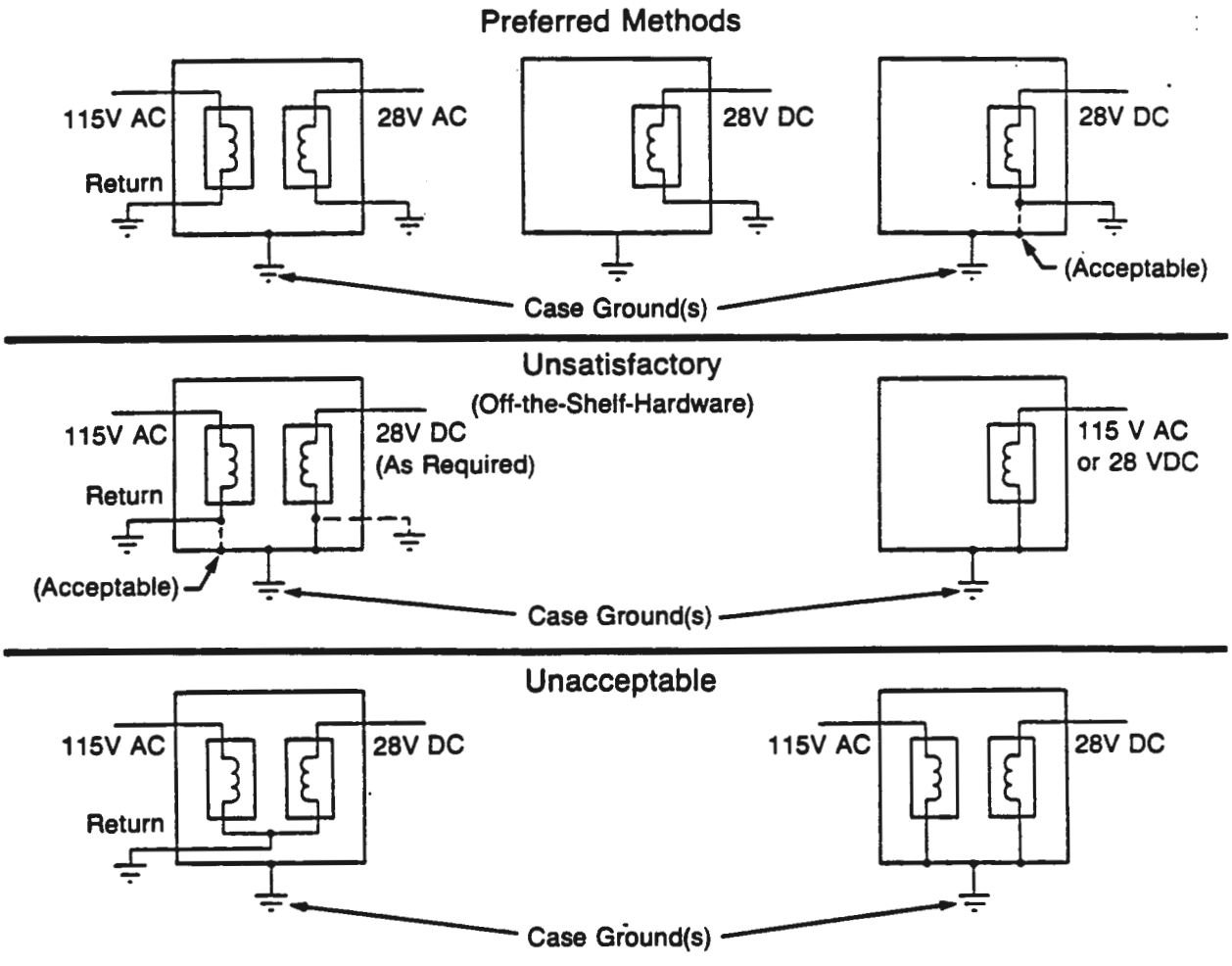



FIGURE 3.5-11. POWER RETURNS

- Audio or analog circuit return.
- Digital circuit return.
- Shield ties (pigtails).

Certain case enclosures require dual grounding in a flammable leakage zone or water exposure area, possibly around wing, cargo, and tail locations, to offer redundancy and safety in case of electrical fault. Power returns are brought out of these zones before being terminated to structure. Each circuit is exhaustively reviewed for voltage, current, resistance, and potential voltage drop at each connection. MIL-B-5087 gives resistance limits of 6 mΩ to 11 μΩ maximum when computed fault currents are 50A to 7 kA.

3.5.4. Resistance

Two and one-half milliohms (0.0025Ω) is the most often quoted resistance maximum for a termination or connection when designing for electromagnetic compatibility. It is applied to wire terminations, case enclosures, mounts, shelves, panels, doors, and radio frequency components. The 2.5-mΩ limit is appropriate wherever any component or structure forms a part of the ground plane, shielding, fault path, or power or signal return.

Panels, rails, frames, access doors, and all conductive items in the flight deck are bonded especially to reduce electrostatic discharge.

Pumps, valves, flanges, lines, vents, and penetrations in the fuel tank are bonded especially to prevent arcing from lightning. Transport aircraft undergo thorough research and assessment of conductivity or resistance. It is sometimes difficult to attain 2.5 mΩ. The resistance of aluminum is a good basis for comparison of other materials (figure 3.5-12 and figure 3.5-13).

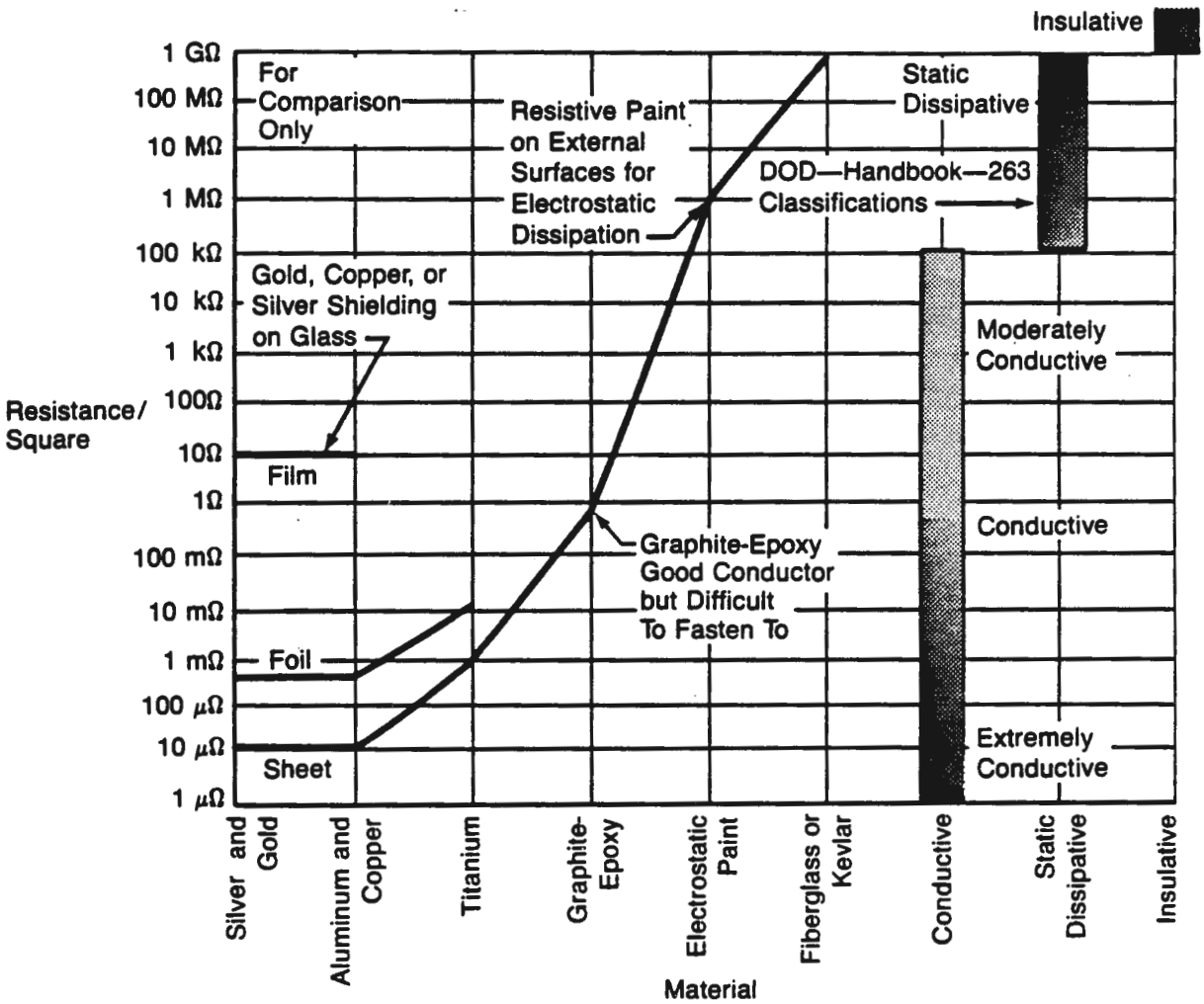



FIGURE 3.5-12. MATERIAL/RESISTANCE OVERVIEW

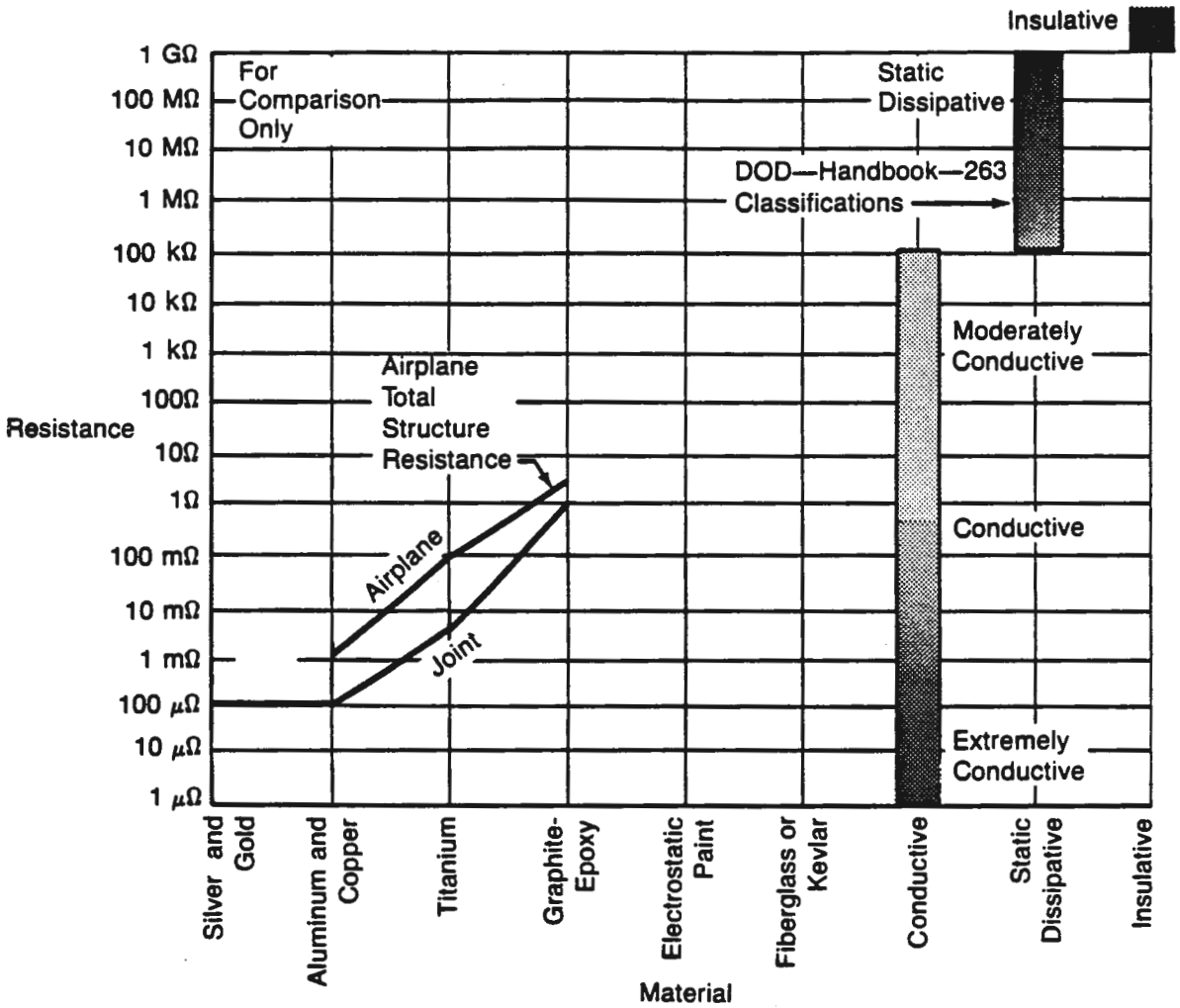



FIGURE 3.5-13. CONFIGURATION/RESISTANCE OVERVIEW

Ground planes establish the electrical foundation for any system. Copper is practical, affordable, and is usually employed in the electromagnetic compatibility laboratory (figure 3.5-14). Currents flowing in copper have a low IR drop and therefore provide a low noise system. Aluminum usually offers good conductivity through faying surface bonds or permanent joints. A number of factors affect resistance: pressure and surface finish are significant (figures 3.5-15 and 3.5-16).

When components do not form a part of the electrical design but must be grounded for safety, a 1Ω resistance is often adequate. Where electrostatic charge buildup is of concern on nonconductive materials such as on external dielectric surfaces, the resistance limit may be allowed to be much higher. Military Handbook 263 gives 500 kΩ to 100 MΩ as the range to adequately drain away electrostatic charge.

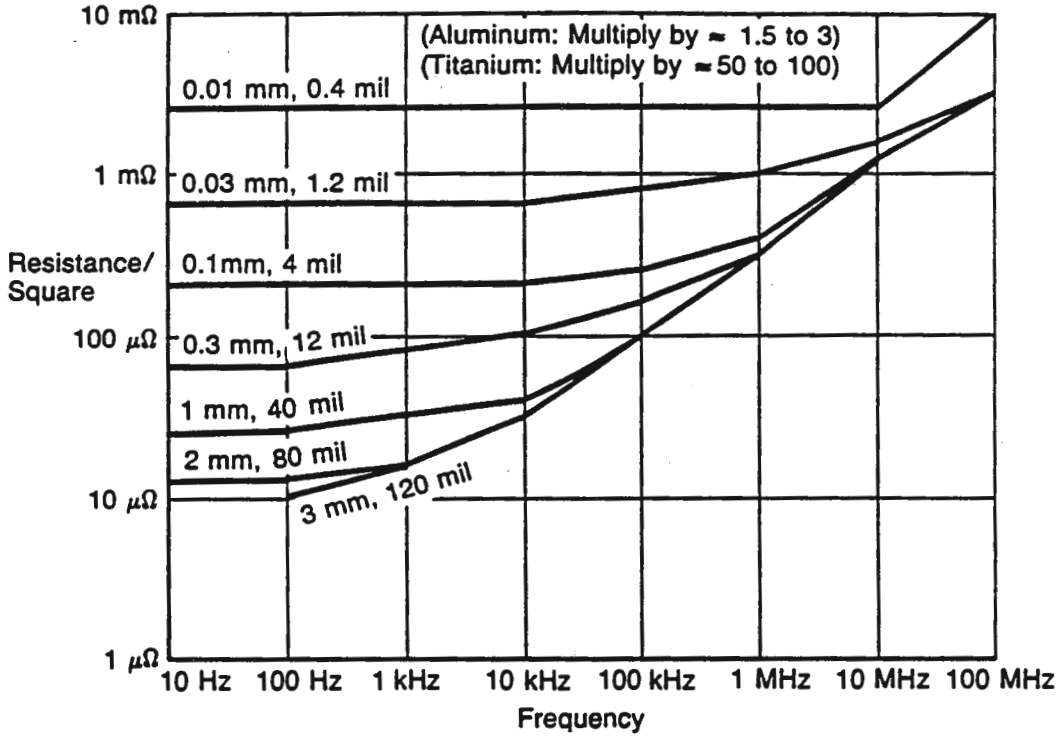



FIGURE 3.5-14. RESISTANCE PLOT - COPPER PLANE

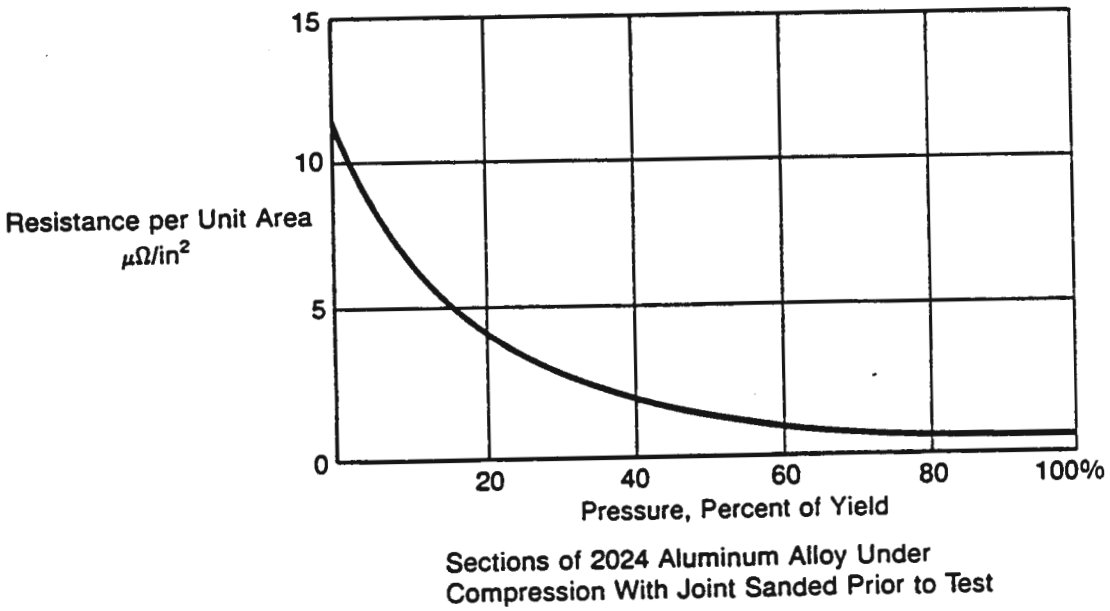



FIGURE 3.5-15. RESISTANCE VERSUS PRESSURE

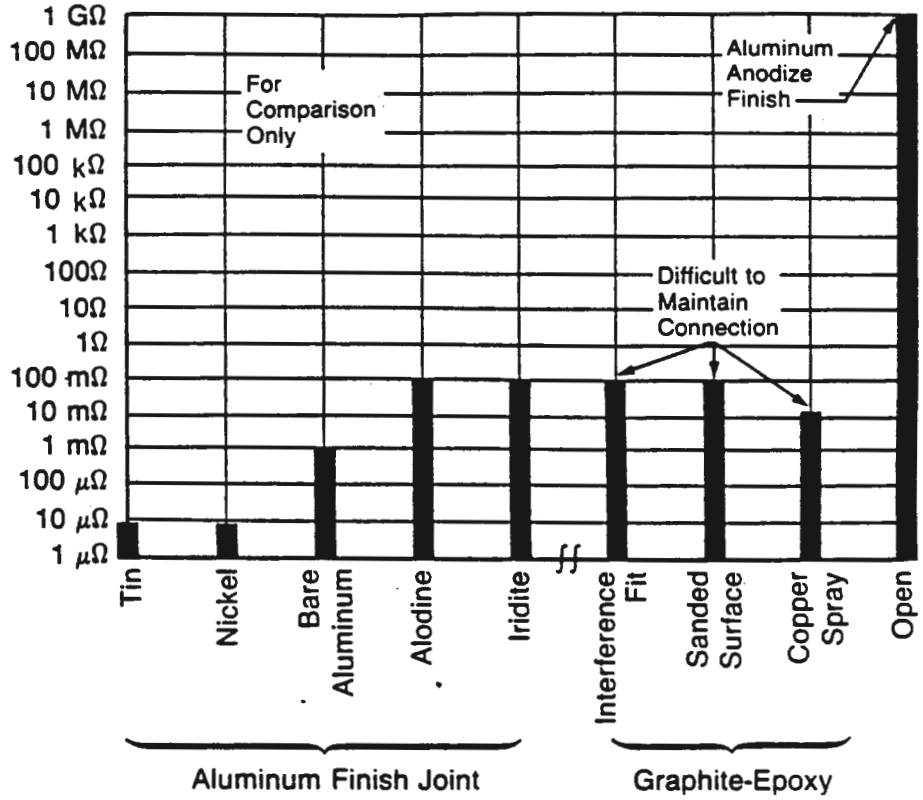



FIGURE 3.5-16. JOINT FINISH RESISTANCE

In summary, the complexity of the joining process and its maintenance is contrasted by the simplicity of the electrical bonding resistance limit: 2.5 mΩ. Aluminum is the grand conductive foundation. The part that it plays cannot be overstated. It stabilizes, shields, conducts, isolates, and protects.

3.5.5. Resonance

Aluminum structure and single wires as conductors have a drawback. They resonate (figure 3.5-17). Structure and single wires must be avoided in high-frequency circuit design.

3.6. Composites

Graphite-epoxy is a fair to good conductor and shield.

Kevlar and fiberglass are insulators and have no affect on radio frequency fields. Dielectric structural materials need to be modified to provide electromagnetic shielding.

Nonmetallic materials form many parts in today's aircraft (figure 3.6-1) and are expected to increase in the future. When varieties of materials are brought

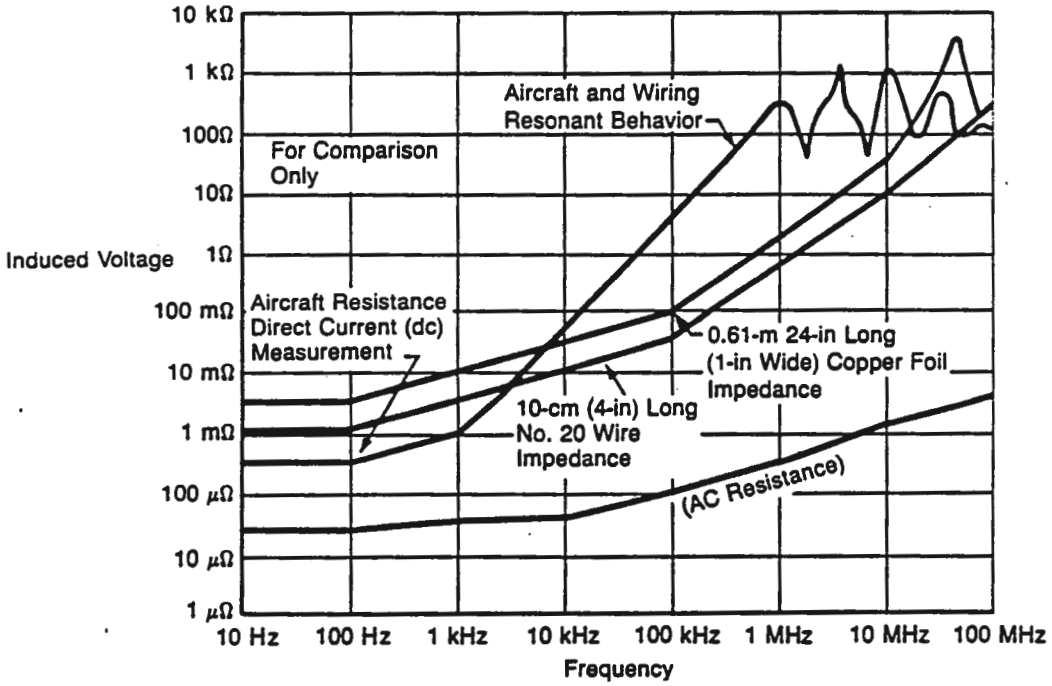



FIGURE 3.5-17. AIRCRAFT AND STRAP RESONANCE

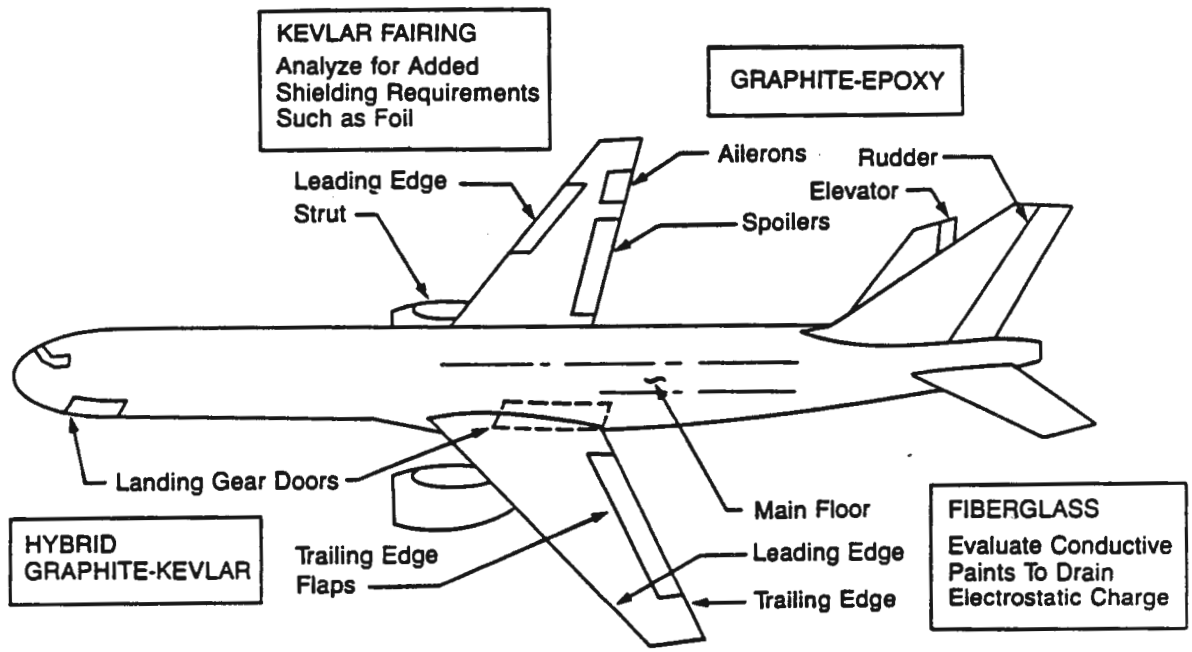



FIGURE 3.6-1. TYPICAL NONMETALLIC APPLICATIONS

together, the interfaces lead to escalating possibilities of material mismatch between finishes, fasteners, and adhesives with possible adverse implications for the quality of electromagnetic compatibility.

Graphite-epoxy will conduct and will shield. Electric field shielding is very good. Following is a comparison of appropriate electromagnetic compatibility-related characteristics:

- Dc resistivity: more than 1000 times greater than aluminum in longitudinal direction and 100 thousand to 1 million in the transverse direction.

- Joint resistance: 30 mΩ to 1Ω or higher. It is difficult to create an electrical connection to graphite-epoxy, and once the connection is made, it is difficult to maintain. New techniques and better procedures are being developed. Today, graphite-epoxy cannot be used as a power or signal return.

- Magnetic shielding starts at about 1 MHz, rises to 60 dB at 100 MHz with about 35 dB in the HF-VHF range.

If graphite-epoxy is not well bonded at seams and joints, it will not act as a shield.

If composites are used extensively for structure, the greatest impact is in providing for a ground plane, the signal return system, and the power return system, which will require their own installations of wire, conductive foils, strip, or cableways. Graphite-epoxy resistance is too high to provide an adequate ground plane (except for antennas). Figure 3.6-2 shows the general level of resistance of a large cylinder or tube, and indicates that its use as a power return path and signal return path is unsatisfactory.

If graphite-epoxy or insulating materials are built into the structure, loss of shielding may open up a path for noise from fluorescent lights, switching regulator and clock oscillator harmonics to reach the ADF, HF, or VHF receivers. Or, in turn, HF frequencies may enter wiring and the avionics. The effects of radio frequency electromagnetic interference are becoming more of a concern (figure 3.6-3). Figure 3.6-4 illustrates some of the resistance levels that might be encountered and shows the use of twisted pair or twisted pair shielded wire.

In the world of electromagnetic compatibility, our protection and safety lies in an integrated design: architectural or structural shielding, shielded wiring, interface circuit voltage limiters, and software correction techniques. The soul of electromagnetic compatibility is a balanced, isolated, shielded interface circuit: it is practically impervious to conductive, inductive, and capacitive attack of electrical noise.

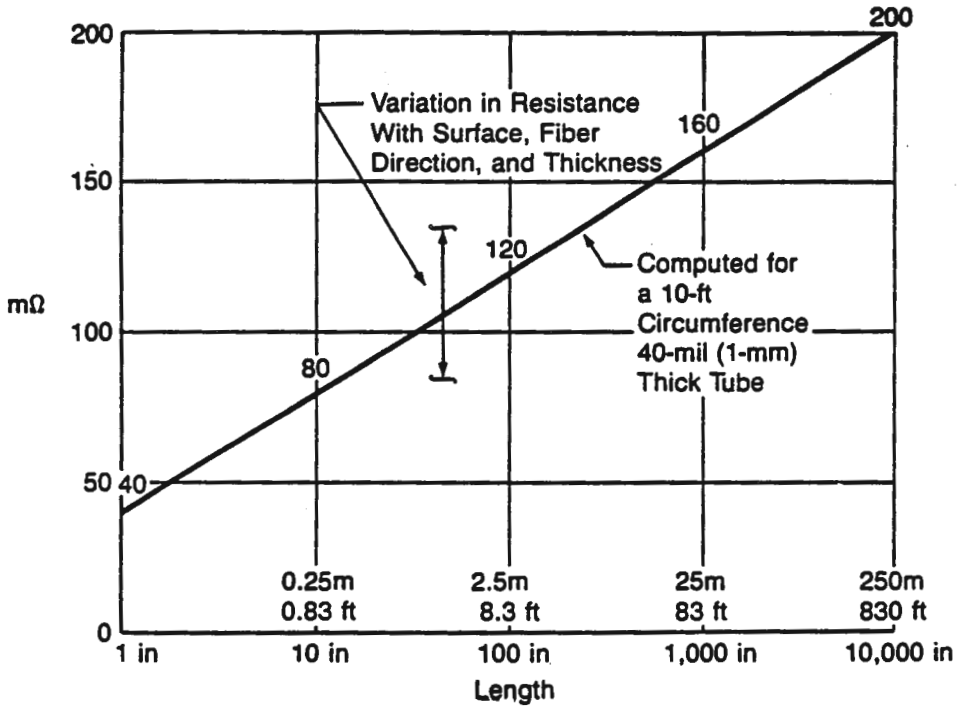



FIGURE 3.6-2. GRAPHITE-EPOXY RESISTANCE

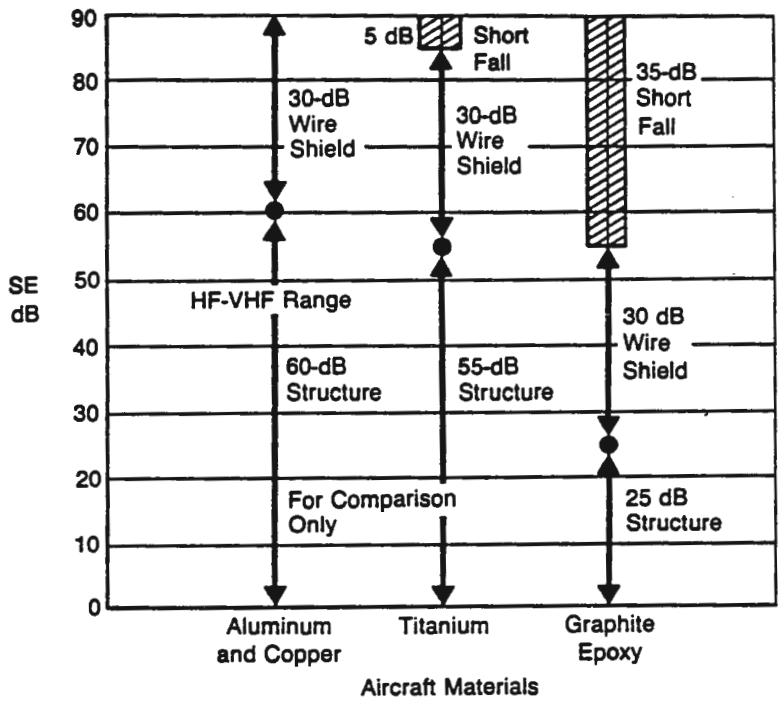



FIGURE 3.6-3. GRAPHITE-EPOXY SE SHORTFALL

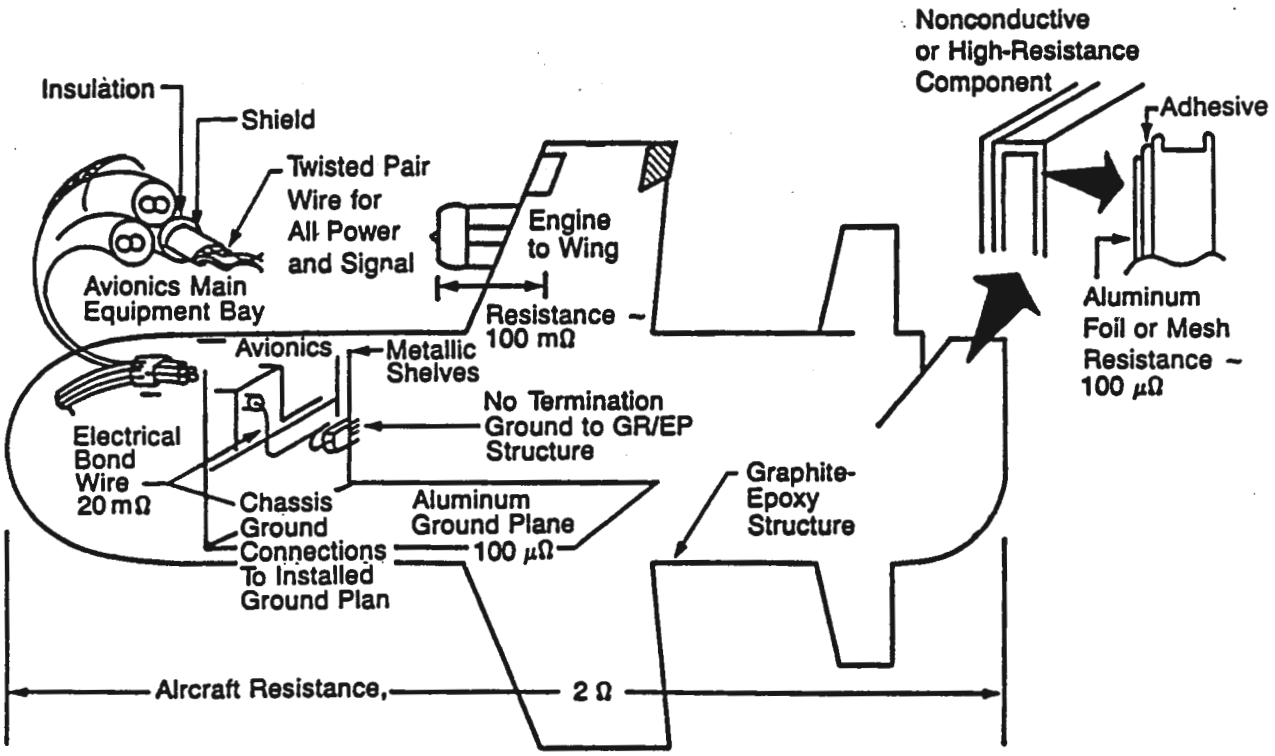



FIGURE 3.6-4. GRAPHITE-EPOXY DETAILS AND INSTALLATION

# 4. EMC ANALYSIS AND ENVIRONMENT

## 4.1. Radiated Environment

### 4.1.1. Radio Frequency Field Distribution

#### 4.1.1.1. Environment

The electromagnetic radio frequency spectrum is vast. Electromagnetic radio frequency field strengths - measured in volts per meter - vary widely. But how do we bind them and treat them so we understand their significance? Well, for the purposes of electromagnetic interference simplicity, one might say we, as humans, and the aircraft, as an object, exist in a dimensional world of roughly 300m to 1m which, converted to radio waves, is 1 MHz, to 300 MHz.

That is very important for electromagnetic compatibility.

The American National Standards Institute (ANSI) radio frequency protection guide for personnel has the strongest field strength limitation over the span of 3 to 300 MHz. People, airplanes, wiring, and the avionics equipment itself are most susceptible to the frequency spectrum spanning the range of 1 to 300 MHz. This is the HF-VHF range (figure 4.1-1). The conditions and behavior of radio frequencies are sometimes so complex that radio frequency measurement and design efforts have often been given the title of "black art." Stray fields, cavities, diffraction, absorption, resonance, constructive interference, destructive interference: all of these interact to make the details of radio frequency studies conceptually difficult over some of the transitional frequency ranges.

The spectrum of radio frequency fields that penetrate aircraft wiring and enter into the avionic inputs to attack integrated circuits can be bounded and summarized over three basic regions:

- A low frequency below 1 MHz where radio frequency noise is normally less of a concern because fields are very inefficiently received on a wire.
- The HF and VHF region, 1 to 300 MHz, where radio frequency fields are very much of a concern and aircraft wiring acting as an antenna is efficient.
- An upper frequency range above 300 MHz where induced voltages in wiring drop off rapidly with increasing frequency and are easier to control.

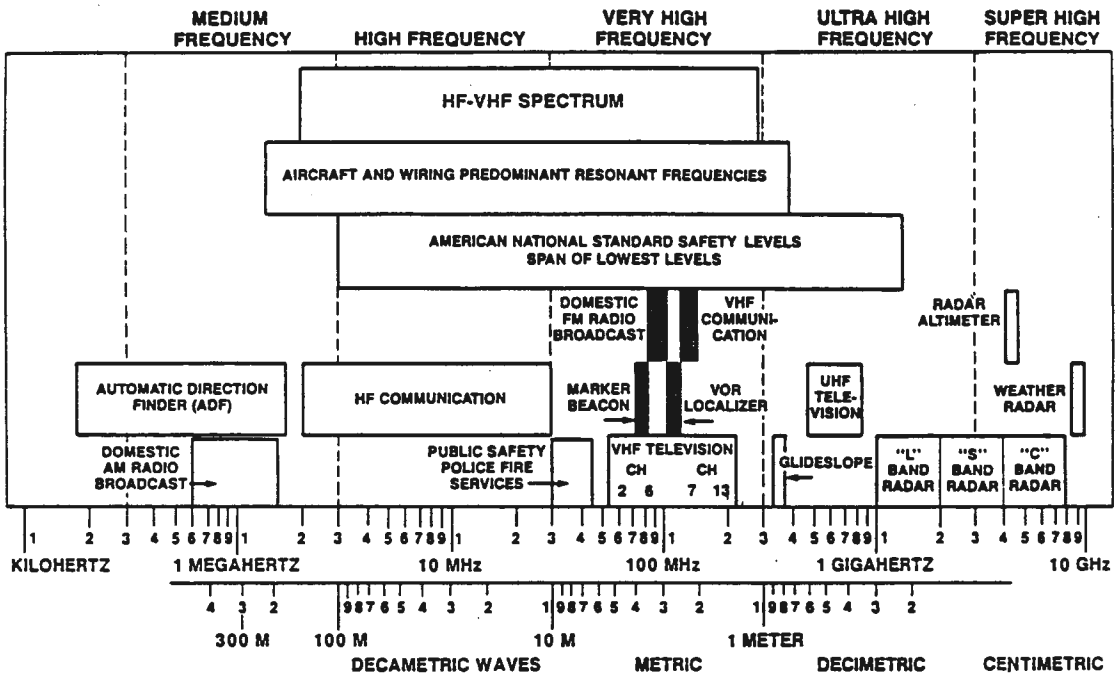



FIGURE 4.1-1. HF-VHF RANGE

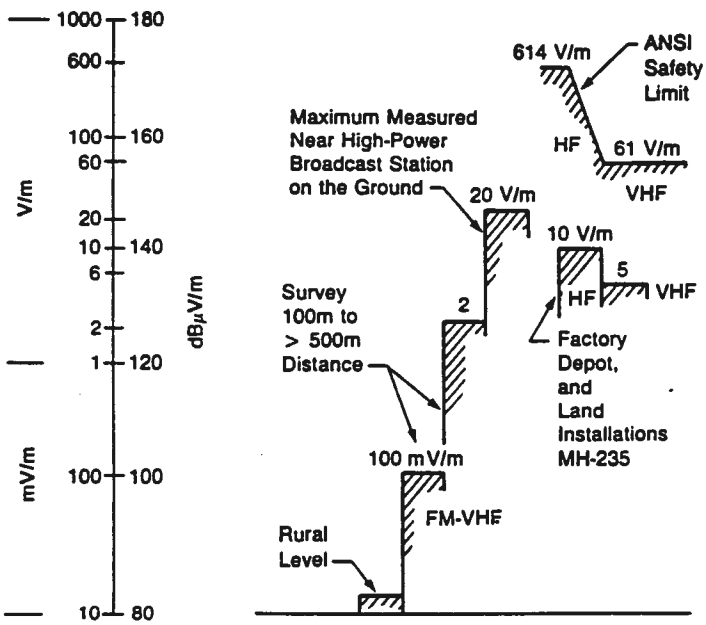



FIGURE 4.1-2. SELECTED RF FIELDS: HF-VHF RANGE

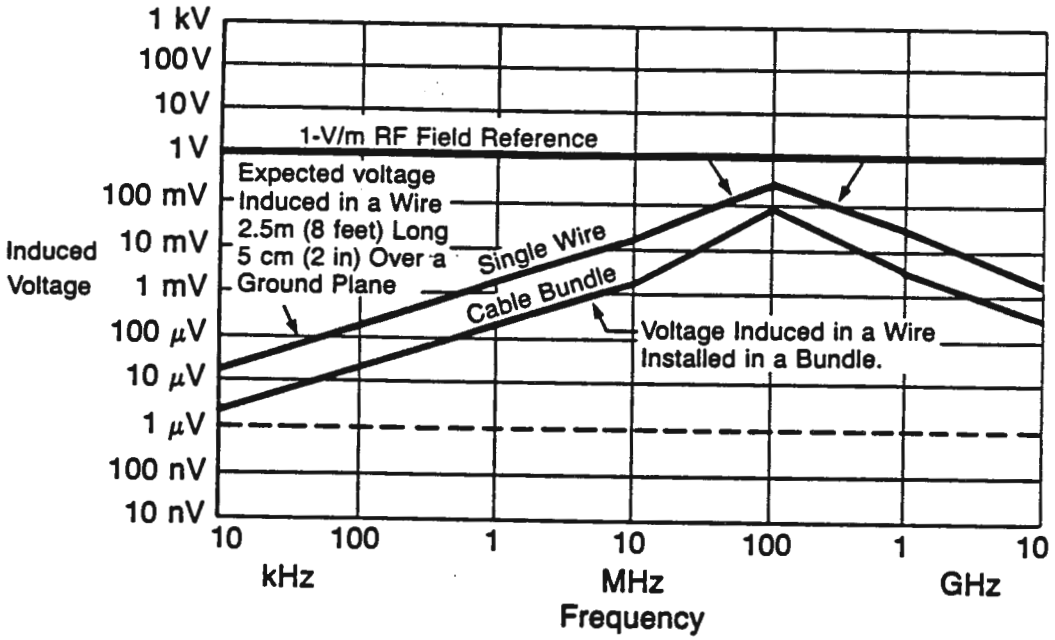



FIGURE 4.1-3. EXPECTED WIRE VOLTAGE - EIGHT FEET

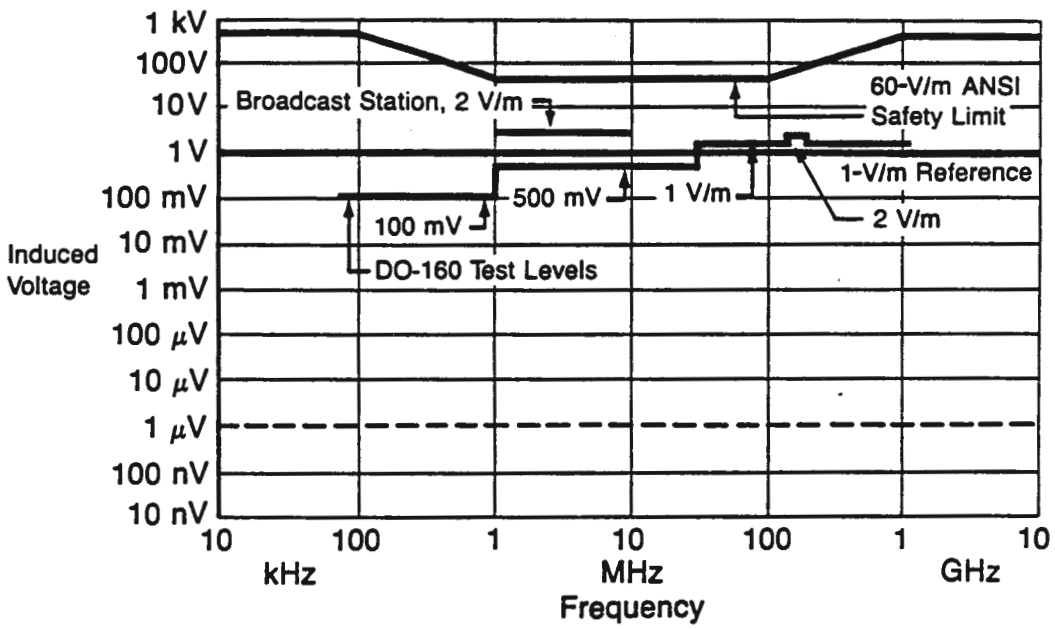



FIGURE 4.1-4. ENVIRONMENTS, SAFETY, AND TEST

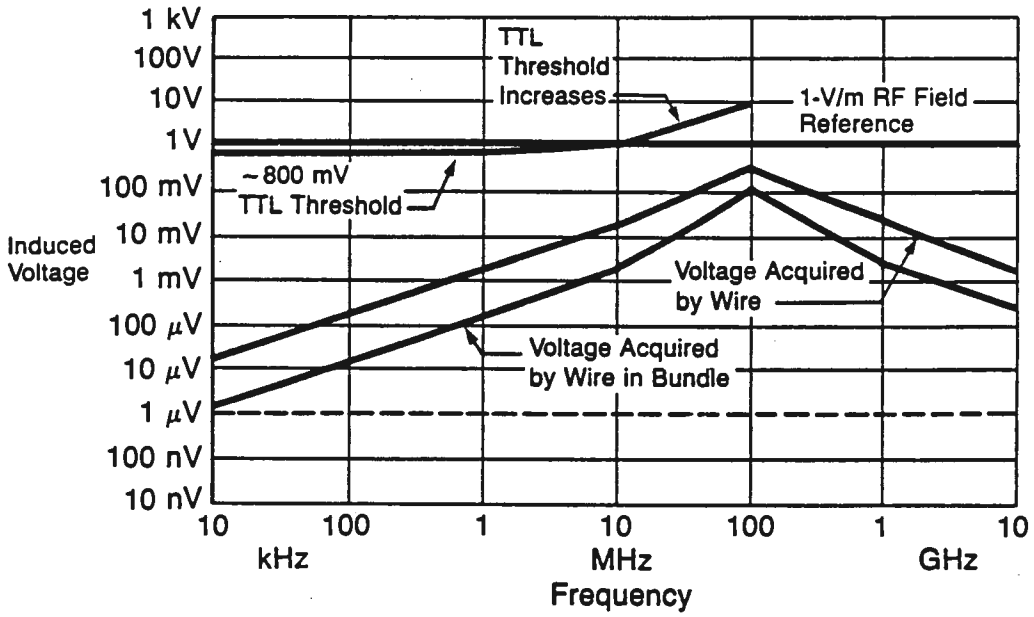



FIGURE 4.1-5. TRANSISTOR THRESHOLD

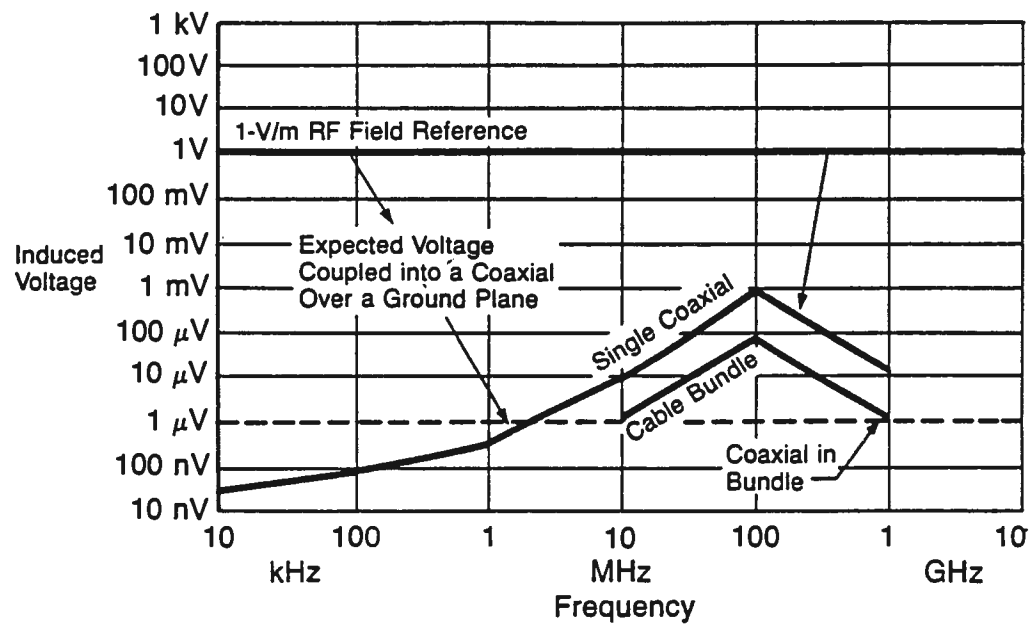



FIGURE 4.1-6. REPRESENTATIVE COAX VOLTAGE COUPLING

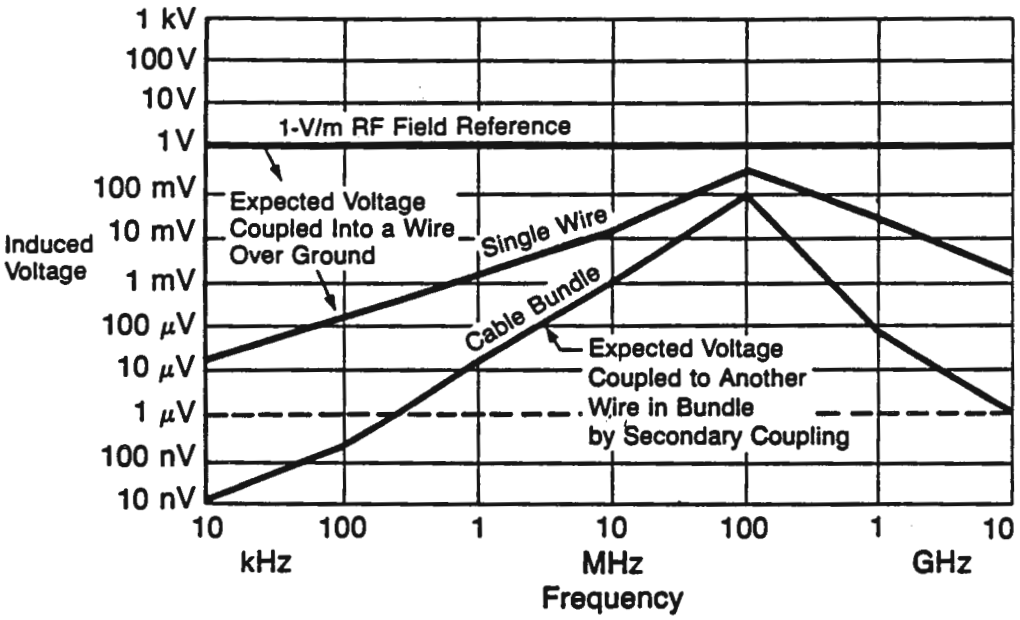



FIGURE 4.1-7. REPRESENTATIVE SECONDARY COUPLING

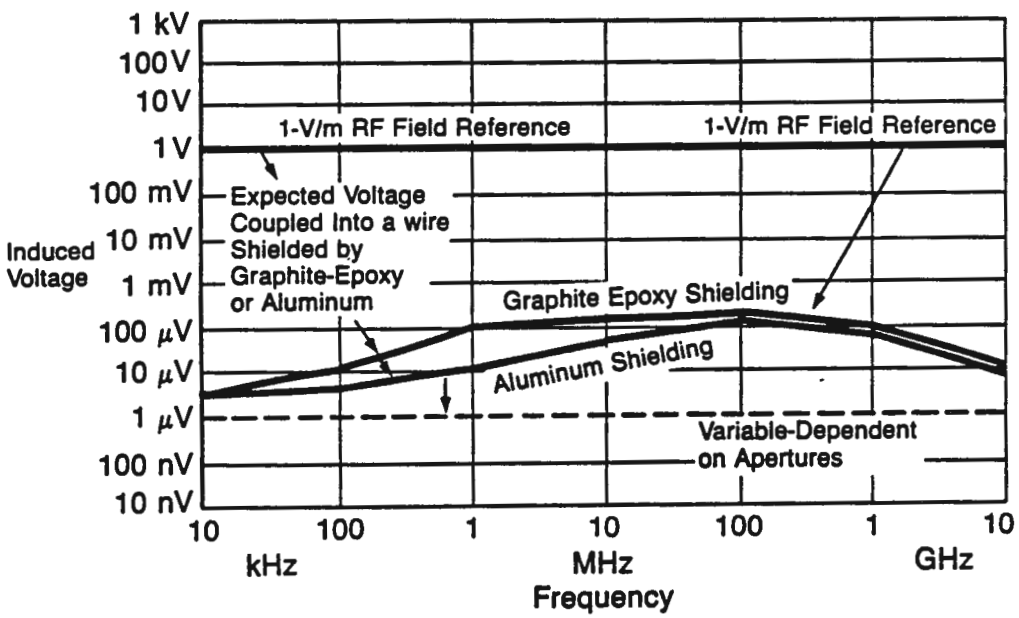



FIGURE 4.1-8. GRAPHITE-EPOXY OR AL SHIELDING ADDED

HF and VHF, ground-based FM radio and TV broadcast signals join with their airborne companions, the aircraft HF and VHF communication links, to induce significant voltages on aircraft wiring in the 1- to 300-MHz range. HF-VHF field strengths measured at ground level for selected urban, suburban, and rural areas are plotted in figure 4.1-2. At altitude, stronger fields can exist (see section 10, Bibliography, ECAC study). The ANSI personnel safety limit and the Military Handbook 235 Environmental Test Specification are plotted for comparison.

Figures 4.1-3 through 4.1-8 give a general view of the expected induced voltages (in some specific wires under selected conditions) and also a comparison of related environments. This information and data is illustrative and advisory to resolve significance and make comparisons of parametric properties and behavior and is not for use or adaptation to specific designs.

Aircraft structures and avionics are engineered and designed by the airframe manufacturer and subcontractors to limit and control these fields. Some lines are shielded, some filtered, and others incorporate a balanced circuit design that will very effectively reject most radio frequencies.

Another main defense against HF-VHF radio frequency signals is a comprehensive and all-inclusive electrical bonding system of all aircraft structural shielding materials to eliminate resistive seams and open seams or apertures. During a specific design program, it is becoming more and more necessary to develop a sophisticated set of models and computations to characterize and define the particular installation and uncover the behavior of the radio frequency fields and induced voltages.

So, in summary, the HF-VHF frequencies are important to aircraft designers because the wiring, equipment, structure, and the aircraft itself, having the same dimensions as the radio frequencies, become efficient antennas.

4.1.1.2. Resonant Behavior

Resonance is the ingredient that brings the arcane aspect to electromagnetic interference. Resonance, standing waves, and reflections share a commonality with black magic in their illusory and unseen character. Electrical parameters, voltage, and current, under the influence of the infinite variability of resistance, capacitance, and inductance, depending on the number of electrical poles, and when viewed over the frequency spectrum, may start out with high impedance and fall quickly to a short circuit minimum. Then they curve back to a maximum. On the other hand, the impedance may start low and rise to open circuit values. Voltage and current in RLC circuits may rise and fall in a periodic fashion or in one simple cycle. They may reach constant highs and lows, or may vary with inconsistency. Resistance, capacitance, and inductance vary endlessly with material electrical properties and length, size, and height above ground. But, they are bounded by maximum and minimum envelopes and the spreads, although large, are limited. In a shield room, standing waves on wiring may rise up at around 10 MHz.

Shield room resonance itself usually is around 50 MHz depending on size. On a large transport aircraft, resonance may occur around 1 MHz and continue into the higher frequencies. The fuselage may resonate at 1 MHz, the wing and other structural details at higher frequencies.

In a balanced transmission line circuit, resonance is kept to a minimum; signal stability and quality are maintained.

Single wires and braids (at their resonant frequencies in the 10s and 100s of MHz) offer very high impedances. A 40-inch, or 24-inch, or 9-inch wire or braid at 70 MHz (the VHF range) may have an impedance that has risen to 10 kΩ. A 4-inch long braid or a #20 wire exhibits about 500Ω at 100 MHz, roughly the same frequency range. With this kind of impedance increase and resonant condition, adding a wire connection between a circuit and ground may alter the surrounding field strength levels by 10 or 20 dB. The significant consideration here is that they are not stable conductors. As soon as appreciable capacitance and inductance are added, conductor quality is lost in a single wire or braid.

In unencumbered space, radio frequencies behave linearly, constantly, and predictably. Introduce conductors, metallic planes, and a range of materials, resistance, and dielectrics; and vary the structure, cabling shape orientation, and location; then only modeling and measurement techniques can focus on and map the radio frequency fields with any accuracy and reliability.

Even though there is a wide range of variation in the HF-VHF radio frequency region, the testing and measuring is performed in an equivalent shield room, metallic environment, complementary to the aircraft fuselage metallic environment. Increased field strengths from resonant and reflection activity in the shield room have been a rough emulation of the final installation.

### 4.1.2. Magnetic and Electric Field Distribution

From the 400-Hz powerline, those ubiquitous, unidentical twins, electric field and magnetic field, although easy to contain or block, nevertheless crisscross the length and breadth of today's aircraft and are present in every bundle and on every circuit. The magnetic field permeates all materials. It may couple to other wires. A wire with 4A may cause noise voltages up to 400 mV (figure 4.1-9) depending on wire length. These noise voltages can mar the signal fidelity in the input circuitry of analog sensors, passenger entertainment, radio headsets or interphones. The electric field charges all materials. It can induce high voltages. A 100-ft wire with 160V peak, 115V rms, may induce over 100V peak (figure 4.1-10) onto high-resistance circuits. These noise voltages will pester operational amplifiers or comparator circuits, although they will ignore low-resistance analog or digital signal drivers.

These twin marauders, almost always together, are strengthened with increasing bundle length and encouraged by proximity of common wire routing. The magnetic field and electric field induced voltages of figures 4.1-9 and 4.1-10 have been plotted together on figure 4.1-11 and shown with the added variable of wire separation - the other dominant wire coupling parameter. See in figure 4.1-11 how the induced voltage drops with wire separation.

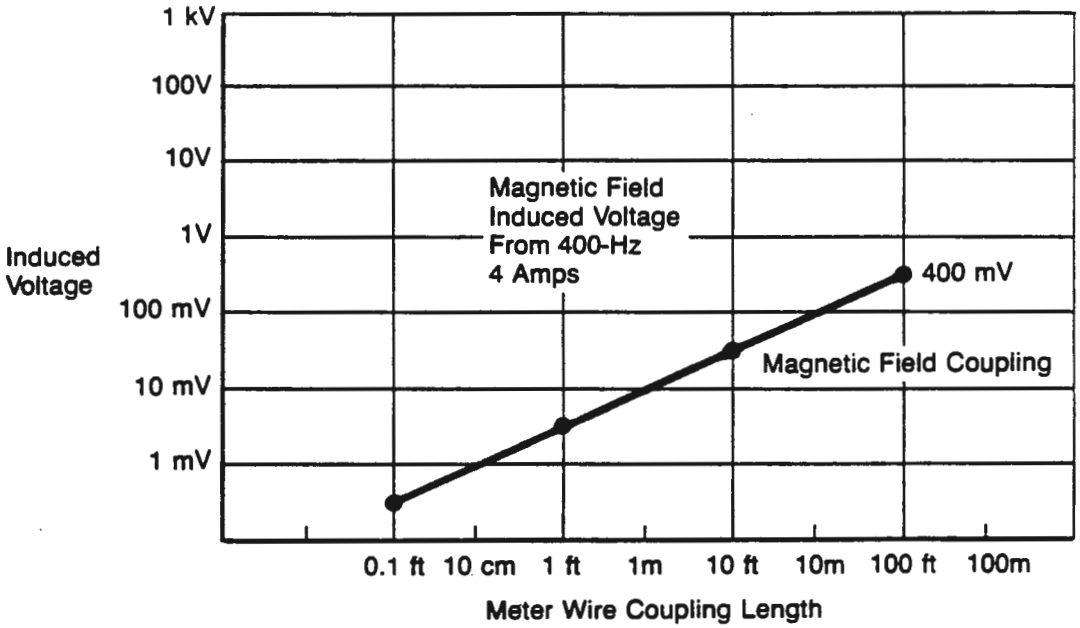



FIGURE 4.1-9. "H" FIELD COUPLING

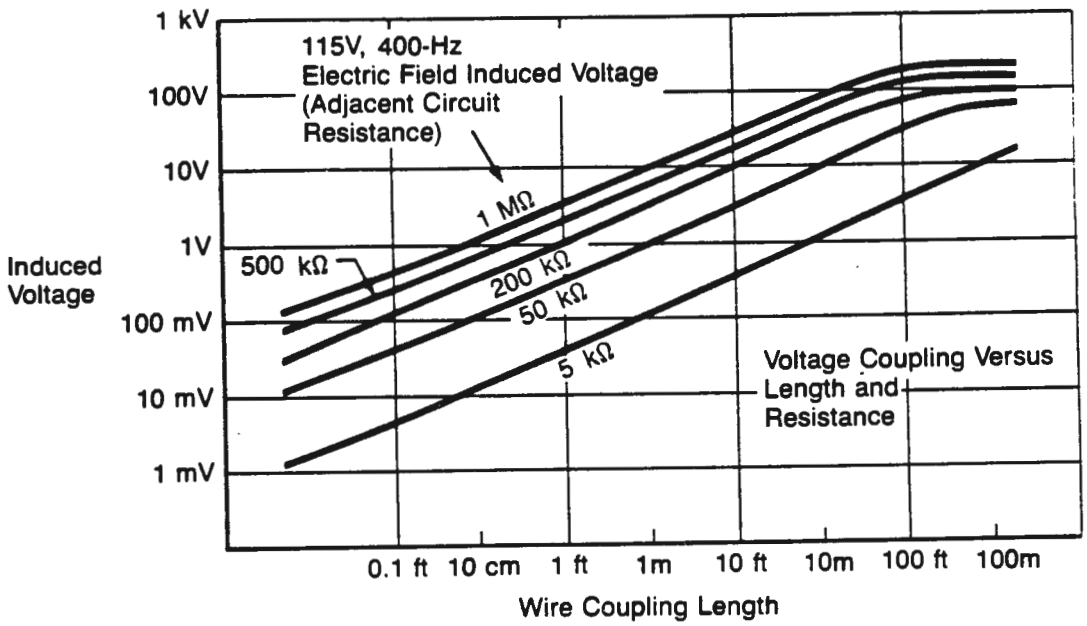



FIGURE 4.1-10. "E" FIELD COUPLING

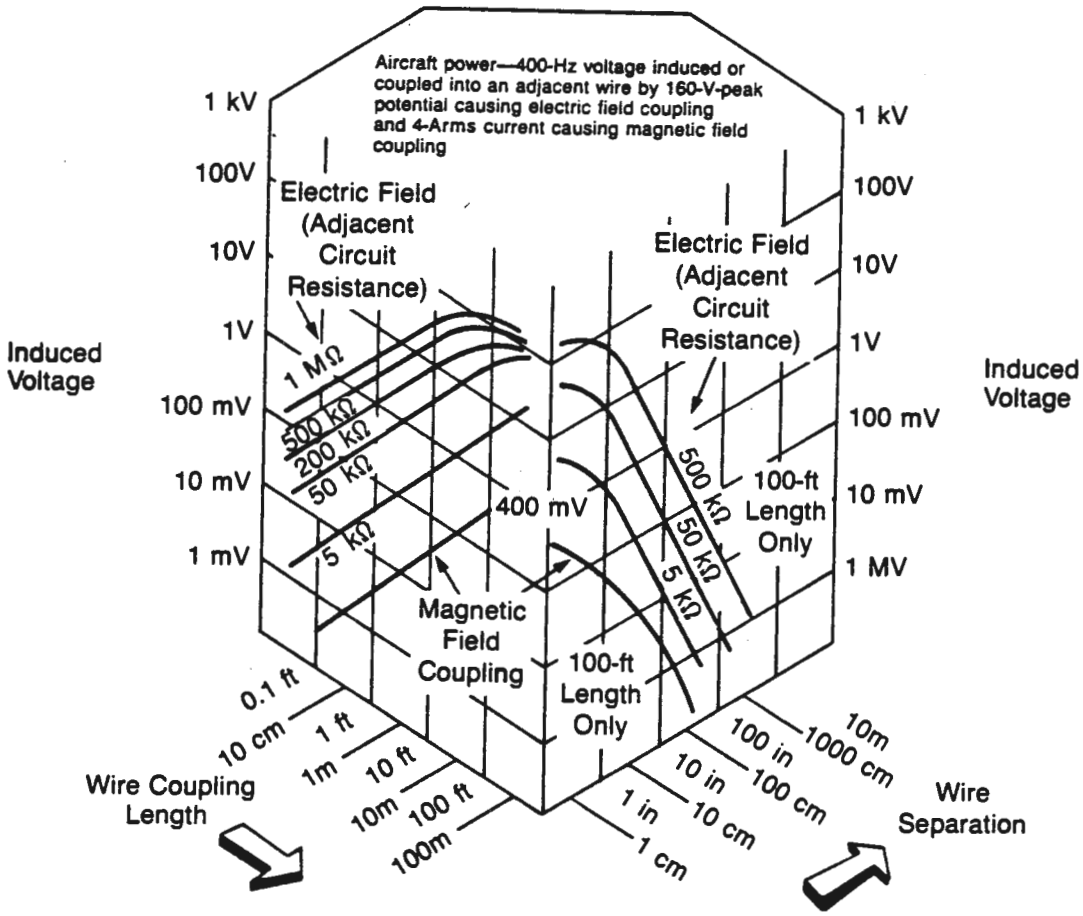



FIGURE 4.1-11. COUPLING VERSUS SEPARATION

Now, there is one exception to the design panacea of using a twisted pair, shielded circuit as the perfect defense against electromagnetic interference. On present day aircraft, a circuit, even though it is twisted pair, will be subject to magnetic field induced voltages if it has a shield and the shield and circuit are grounded at both ends. (Aluminum aircraft structure carries the 400-Hz power line return currents. The 400-Hz current will thus travel in shields grounded at both ends. The shield current then transfers a voltage to the internal circuit.) A balanced isolated circuit design or double shielding will compensate. The first twin, magnetic field, induces voltages on other wires in a bundle, but it also radiates out from the bundles into the aircraft interior space and into equipment (figure 4.1-12). The radiated magnetic field emanating from the power lines varies in field strength from microgauss to gauss (milliamps per meter to amps per meter). The fields are of interest when flight-deck CRT displays, hand-held transceivers, and microphones (whose operational functions rely on the use of magnetics for control or display processes) are present. The magnetic field will easily cause distortion or "banding" on display tubes or will develop the sound of a 400-Hz "hum" in a speaker. Magnetic fields drop off in magnitude very rapidly with distance.

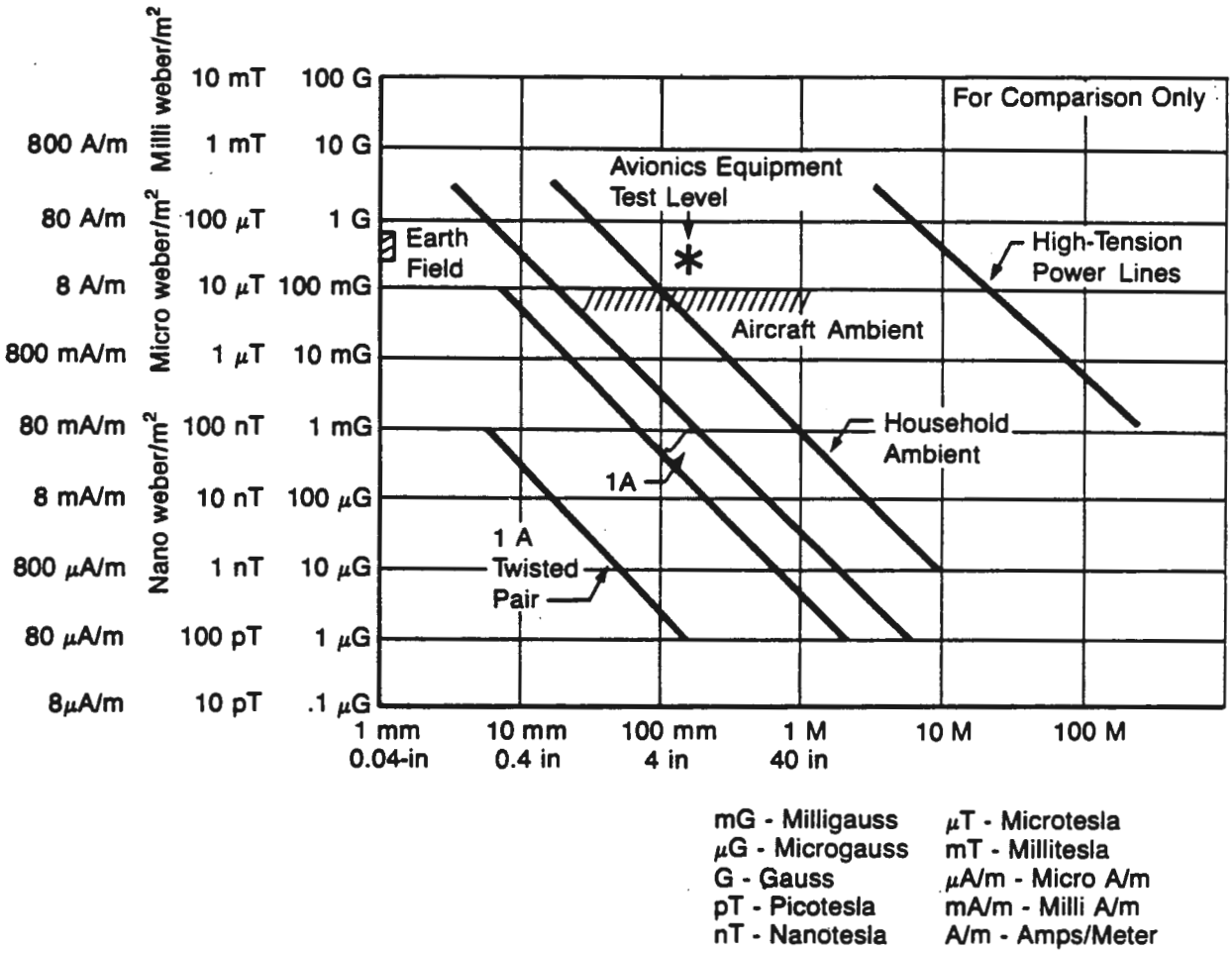



FIGURE 4.1-12. MAGNETIC FIELDS

The second twin, the 115V electric field from 400-Hz power lines, is in almost every aircraft cable bundle. The power line wire has a 115V rms (160V peak) potential. Another wire circuit brought into proximity of the 115V wire holds the 115V potential unless it is destroyed by the "electrical resistor divider" action of a low resistance on the adjacent or "victim" circuit. Figure 4.1-11 shows the reduction of voltage with reduced resistance on the adjacent circuit. The resistance is the parallel resistance of source and load. Either a low resistance or a shield will stop electric field coupling.

Careful attention to wiring details and design such as the use of twisted pair wire, shielding, balanced circuits, or fiber optics helps protect the aircraft against electric and magnetic fields.

4.1.3. Transients

A 115V power line supplying an inductive load, when interrupted by the opening of a switch or circuit breaker, will create an electrical switching transient

often called an "inductive switching transient" or a "relay-induced transient." Electrical switching transients from lights, fans, pumps, or control surface actuator operation reach levels of 600V. Their duration can be over 1 ms, and their rise times nanosecond to microseconds. They harbor repetitive ringing waveforms or pulses (figure 4.1-13). These recurring ringing frequencies span the 1 to 10-MHz range. The switch opens, an electrical arc is started, and an inductive coil unloads stored energy through the arc in repetitive pulses onto a wire - a wire that may be bundled with other circuits carrying digital data or analog signals. The high-frequency energy is transferred capacitively and inductively to those circuits and directly into the internal harnessing and etched circuit card traces and microcircuits of an avionic unit. The electrical transient distributes to microprocessor clock, timing, or data lines.

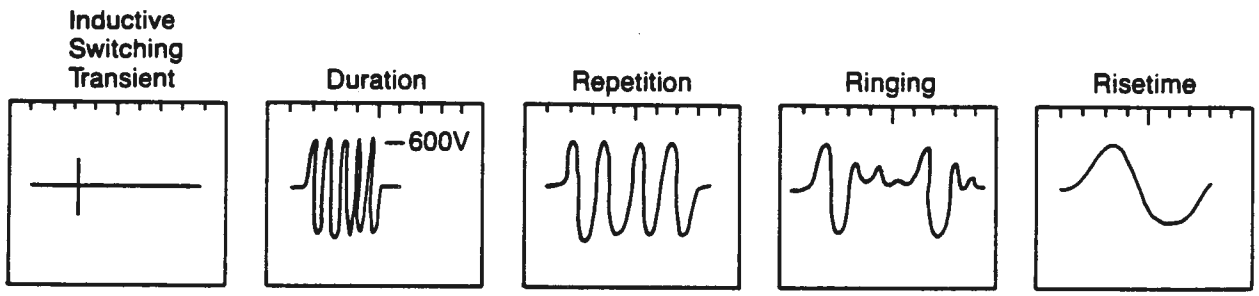



FIGURE 4.1-13. TRANSIENT RINGING

Very extensive measurements have been made of transients and their characteristics. L. Bachman states in his 1981 report: "This paper summarizes the results of the most comprehensive study ever conducted of U. S. Navy shipboard power line transients. Transient data was acquired on 13 ships - over 9400 hours of monitoring time - 2300 transients were encountered." Figure 4.1-14 is a summary. This study is remarkable in its extent and completeness. These transients show very similar, if not identical, characteristics of magnitude, risetime, and duration to those measured on commercial aircraft. In contrast to the high-level, inductive switching transient, the power bus may also experience momentary power interruptions, where the voltage drops to zero for as long as 50 ms.

Electrostatic discharge is another form of transient that can suddenly erupt under the right conditions of humidity, air or fluid flow, and juxtaposition of materials. Electrostatic discharges have much higher voltages - up to and over 10 kV, but much narrower pulse widths, possibly a 100-ns duration. A person on a dry Arizona-like day touching an avionic unit may discharge 10,000V onto the housing or wiring.

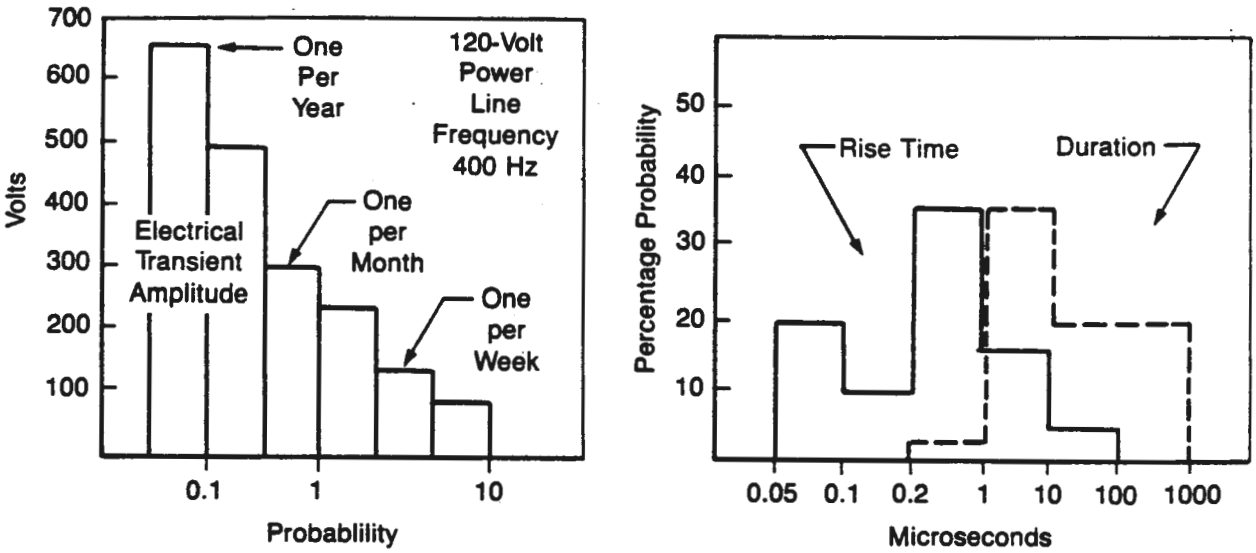



FIGURE 4.1-14. ELECTRICAL TRANSIENTS

One of the most evident electrical discharges, of course, is lightning. Large currents flow on the wing structure, fuselage, landing gear, or empennage leaving induced voltages in unprotected circuits. These induced voltages have resonant behavior often established by the characteristic of the size of the aircraft and length of wiring. Lightning protection is engineered to divert and contain voltages and currents in metallic structures to avoid damage to electrical wiring and parts. The transient voltage levels induced in wiring are required to be less than 600V. Induced voltages are usually less than 200V for the most severe strikes.

So you see, HF-VHF radio frequencies, 400-Hz power, and the various transients are electromagnetic interference or noise forms that are freely distributed and transferred into microprocessors to affect aircraft performance unless constrained and controlled by carefully integrated wiring and avionic and structure design.

## 4.2. Aircraft Protection

### 4.2.1. Shielding and Ground Reference

Aircraft and circuit protection is formed in a layered design: 1st layer, structural shielding, liners, trays, overbraid; 2nd layer, circuit shield; 3rd, balanced, isolated circuit; 4th, voltage, current limiting; and 5th, software.

Double shielding (two layers of shielding) fends off lightning and radio frequencies. For critical circuits having exposed wiring, double shielding is necessary.

Structure, skin, and panels of the aircraft form the first major level of shielding, and it is the system level barrier. Fuselage shielding is extended into open sections or unshielded areas, such as leading and trailing edge or landing gear, with a cableway, overbraid, or foil (figure 4.2-1). Shielding and wiring for radio frequencies, electrostatic charge, and lightning, safety fault returns and ground reference all call for a highly conductive material. Liners can be installed in nonmetallic electronic bay sections. Where spar, skin, or panel shielding is not inherent in structure, then foil, metal spray, or metal mesh (figure 4.2-1) can be designed into or onto nonconductive parts. Leading and trailing edges, engine bays, wheel wells, bulkheads, tail sections: all need evaluation to delineate the shielding and ground planes. Nonconductors, composites, and graphite-epoxy are not a ground or current return.

The second level or layer of shielding is individual circuit shielding continued (with back-shells) to the avionics enclosure shield. The second level offers protection against external threats and internal noise in wiring bundles too. The second level is connected, "grounded," to the first level.

### 4.2.2. Apertures and Electrical Bonding

Conductive panels, foil, or paint form and continue the shielding enclosure on skin panels and trays and, unless electrically bonded, will develop harmful voltages or electrostatic charge centers. (See excellent reports in bibliography by L. O. Hoeft on shielding and aperture losses.) Often good continuity is provided by fasteners. External items are electrically bonded and grounded to provide static discharge paths. Conductive paint is applied to external nonconductive surfaces.

Everything is searched out to check resistance.

- Materials: titanium, steel, stainless steel, aluminum alloys, fiber glass, graphite-epoxy, Kevlar, phosphor bronze, composites.
- Parts: instrument panels, keyboards, switching panels, seats, frames, window films or mesh, quick access doors, skin panels, cowls, fairings, trays, overbraid, backshells, plumbing, brackets, covers, control surfaces.
- Liners: foil, mesh, plating, depositions.
- Finishes: alodine, anodize, iridite, organic applications, copper plating, tin, silver, nickel.
- Sealants: paint, film, special substances.

Bonding is detail work, and often research and development is needed on new materials and techniques. Materials and procedures are recorded in a bonding and corrosion prevention document.

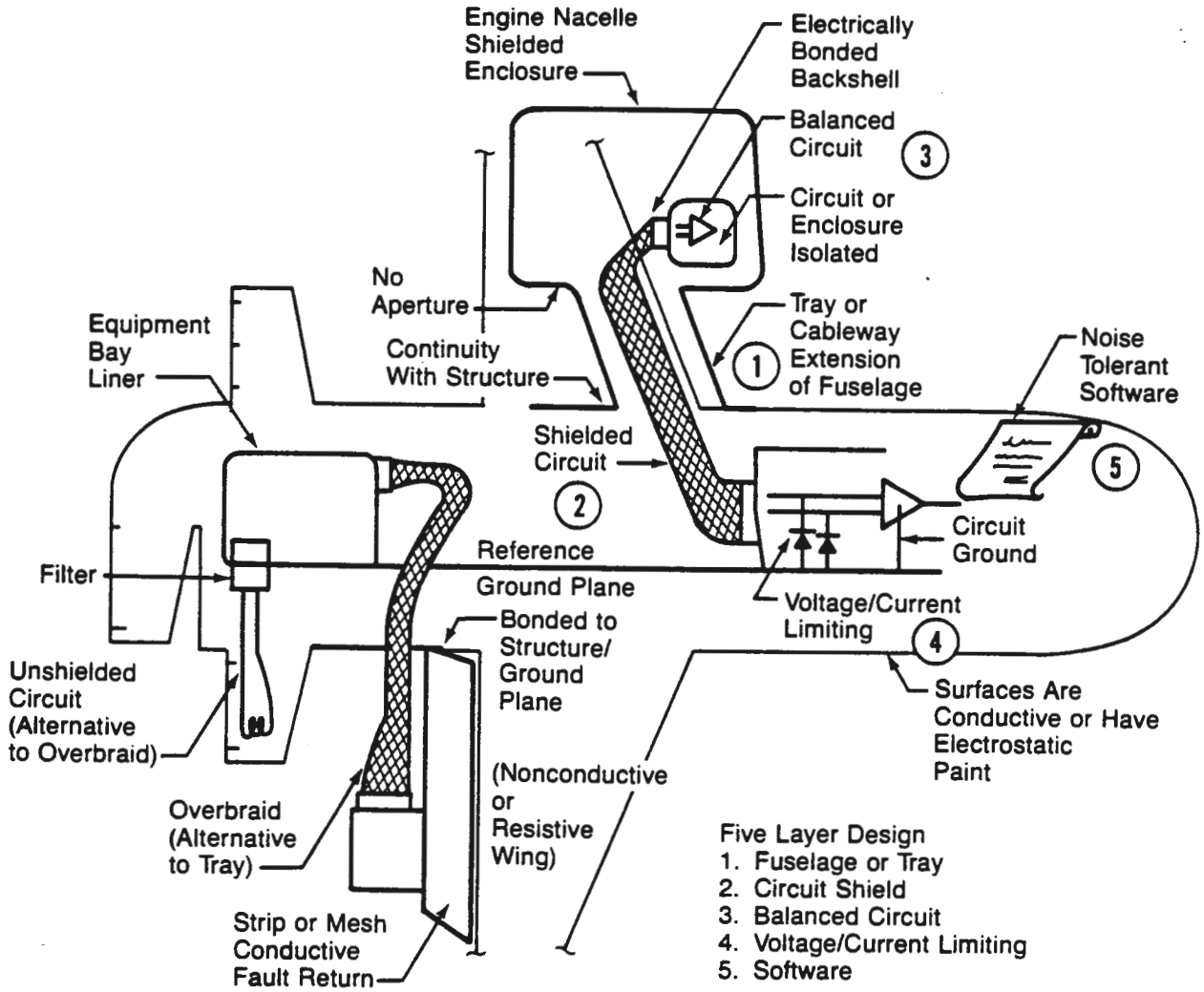



FIGURE 4.2-1. LAYERED DESIGN

4.2.3. External Wiring Interface Circuits

Long wiring runs extending out to the wing and to the engines are unintentional antennas that collect noise. Data, control, and sensor lines, outside of the shielding of the fuselage, demand protection against radio frequencies impinging on engine struts or mounts, leading and trailing edges, and landing gear.

Engine instruments, pressure, temperature, speed, thrust control, air data, fire, flight control computers (circuits for actuators and control surfaces), proximity switches, position indicators, temperatures, and braking circuits are candidates for analysis and protection. Circuits are categorized by criticality. Line replaceable units may number from 50 to 100 units and digital buses,

200 to 300, but the buses will reduce to 30 or so different types with only 5 or 10 types being external to the fuselage. Future systems may have less than 10 digital buses. Discrete circuits number 30 or 40 different types, with around 10 external. They can be thoroughly analyzed.

Good rules for external circuits (there are exceptions):

Grounding:

- Circuits isolated from structure at exposed end.
- EMI tests applied to returns/grounds of units not on a ground plane.
- Primary to secondary power isolated.

Bonding:

- Case bonded to ground plane (when circuits isolated from case).
- Case isolated from ground plane (when circuits grounded to case).
- Backshell/connector bonded to case.

Shielding:

- Double shielding installed on transmitter lines.
- Internal shield grounded internally.
- External shield grounded externally (to backshell).

Input/output:

- Interface circuits balanced; clock and data signals routed together.
- Interface circuits isolated: transformer, LEDs, fiber optics.
- High resistance (greater than 10 kΩ) designed into circuits.
- All circuits filtered.
- No shared power wire returns.

Wiring/packaging:

- Return wire twisted with signal wire.
- No wires installed across an aperture (apertures bonded).
- Line drivers/receivers packaged close to connector.
- Connectors on equipment case placed in one local area.

4.2.4. Circuit Protection

Protection depends upon electronics design, packaging and, especially for external wiring, the voltage stress conditions: stress on insulation, thin films, integrated circuits, and trace spacing, Solenoids and motors with heavy insulation and no electronics usually do not require protection. Thin film and integrated circuits do. (See comprehensive reports in bibliography by R. L. Carney, R. A. McConnell, and D. L. Sommer for excellent treatment of protection devices.)

Every interface circuit needs analysis and voltage/current limiting. Naturally, the transient or radio frequency threat must be known; first define the open circuit voltage; second the surge or characteristic impedance of the wiring and the input impedance of the input capacitor, resistor, or diode; then determine short circuit current; and finally the transient time and energy, or radio frequency power. The following are variable, but important, voltage withstand requirements or test levels that apply to insulations, parts, and electronics and are usable benchmarks.

No protection usually required:

- 1500V rms, 2100V peak, 400-Hz signal, impressed for one minute; this is the voltage withstand specification for insulation (solenoids, motors).
- 3000V or greater, one microsecond transient withstand for insulation, varies with humidity, configuration, altitude, time.
- 1500V one microsecond transient, discrete resistor (molded part).
- 1500V fifteen mil circuit card spacing (at sea level).

Protection required:

- 200V or less, thin film, transient withstand voltage.
- 200 to 800V receivers protected by integrated circuit diodes.
- 100V transistors.
- 30V or less, operational amplifier receiver.
- 360V or less, fifteen mil circuit card trace spacing, at 100,000 ft.
- 360V or less, corona initiation at 100,000 ft, varies with shape.

The five layers of protection: (1) structural shielding, overbraid, cableway; (2) circuit shields; (3) balanced, isolated circuit; (4) voltage/current limiting; and (5) software combined with a stable ground reference plane help to guarantee compatibility.

## 4.3. Trades

It is critical to know program design requirements-and the environment. Dissect the electromagnetic topology of the aircraft including digital transmission, power system, and antenna fields. Round up new and off-the shelf equipment and bay locations. Search out aircraft structural and skin panel materials. Define the exposed critical circuits and equipment.

1st Major Trade: Location of equipment, categories, and tailoring of electromagnetic compatibility requirements of each unit. The wiring and the equipment is kept shielded under aluminum or graphite-epoxy and away from radio frequency fields (figure 4.3-1). Units of a subsystem collocated in a protected environment may have the design/test levels eased: lowered for susceptibility, raised for emission. Units not on the wing will not experience resonant conditions. Off-the-shelf equipment is often unchangeable at the input/output interface, and if located internally may save on shielding or externally mounted filters.

2nd Major Trade: Wire length and separation. Shortened wiring or elimination of wiring. Equipment location, equipment combination and wiring deletion are waiting for evaluation, for example, on-engine electronics supplied with on-engine power; major interfacing units physically located close together. Noise amplitude is proportional to length. Metallic wiring may be replaced with fiber optics to save tens to hundreds of pounds and 25% to 50% reduction of wires (figure 4.3-2) and major reduction of EMI.

3rd Major Trade: Digital, balanced, isolated circuits versus analog circuits and single-ended or discrete circuits. Delete or reduce open wiring. Reduction of wiring, shielding, transformers, power, weight through use of multiplexed digital bus (figure 4.3-3).

4th Major Trade: The five levels of layered protection. If structure shielding is not available, the following alternatives may be investigated. Trade cableways, overbraid, and exposed wiring with filter/voltage/current limiting. For one or two wires on noncritical circuit, filter may trade off better than overbraid or cableway.

A recent large program relied on the fuselage for the major portion of circuit shielding. Wheel wells, engine bays, outer wing sections, and radome areas were shielded. Quick access is often necessary. Overbraid (20% of the bundles) was installed for critical flight control circuits to "extend" the airframe. It was indicated in that design that the use of metal overbraid on all bundles might be 10 times heavier than reliance on the fuselage. Also, complexity of the topology limited the use of overbraid, and in certain areas a combination of protection was needed. For circuits inside the fuselage, filter pins were used; for outside circuits, voltage limiters. Discrete filters for each circuit were not pursued because of excessive volume and weight.

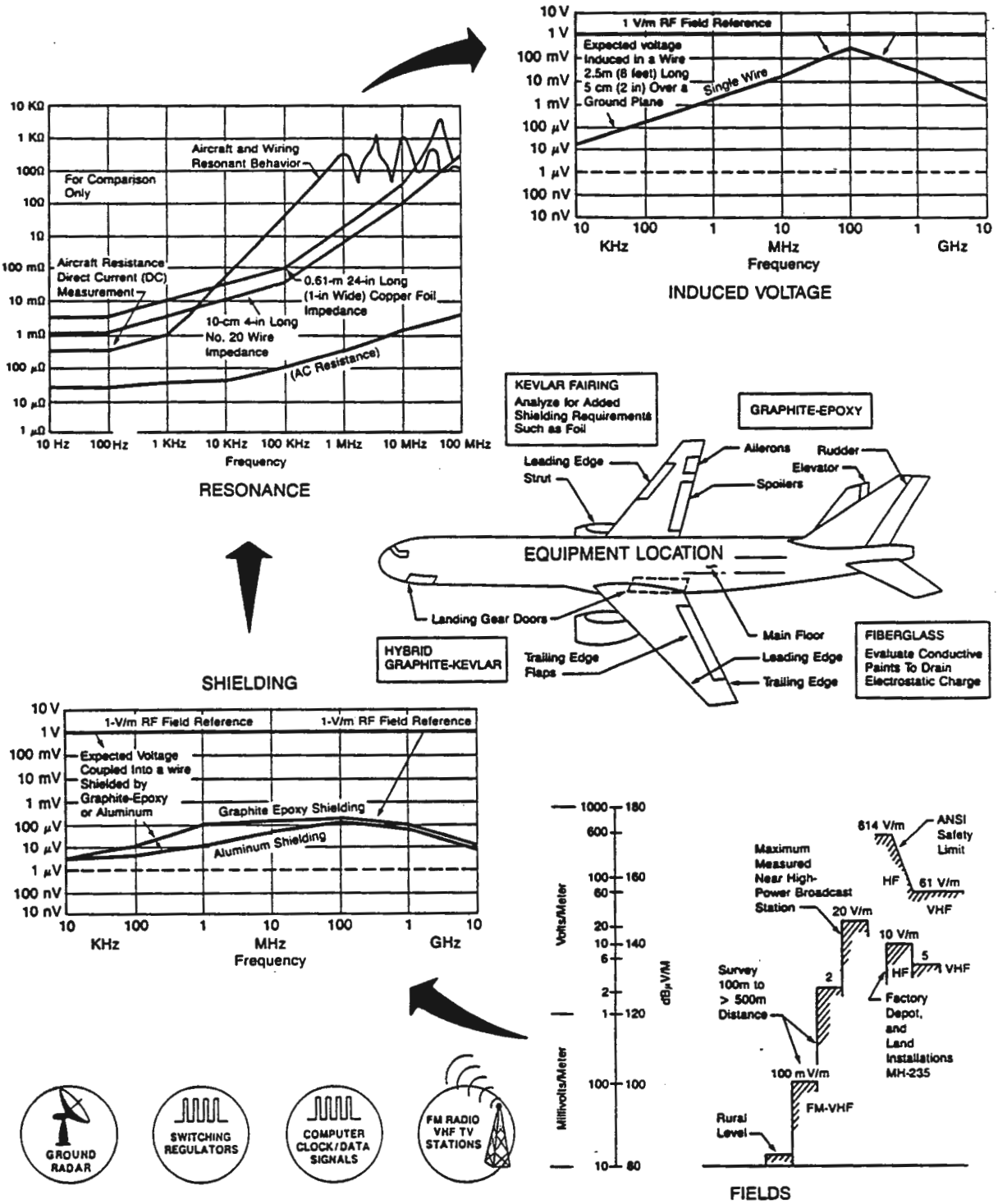



FIGURE 4.3-1. TRADE-EQUIPMENT LOCATION

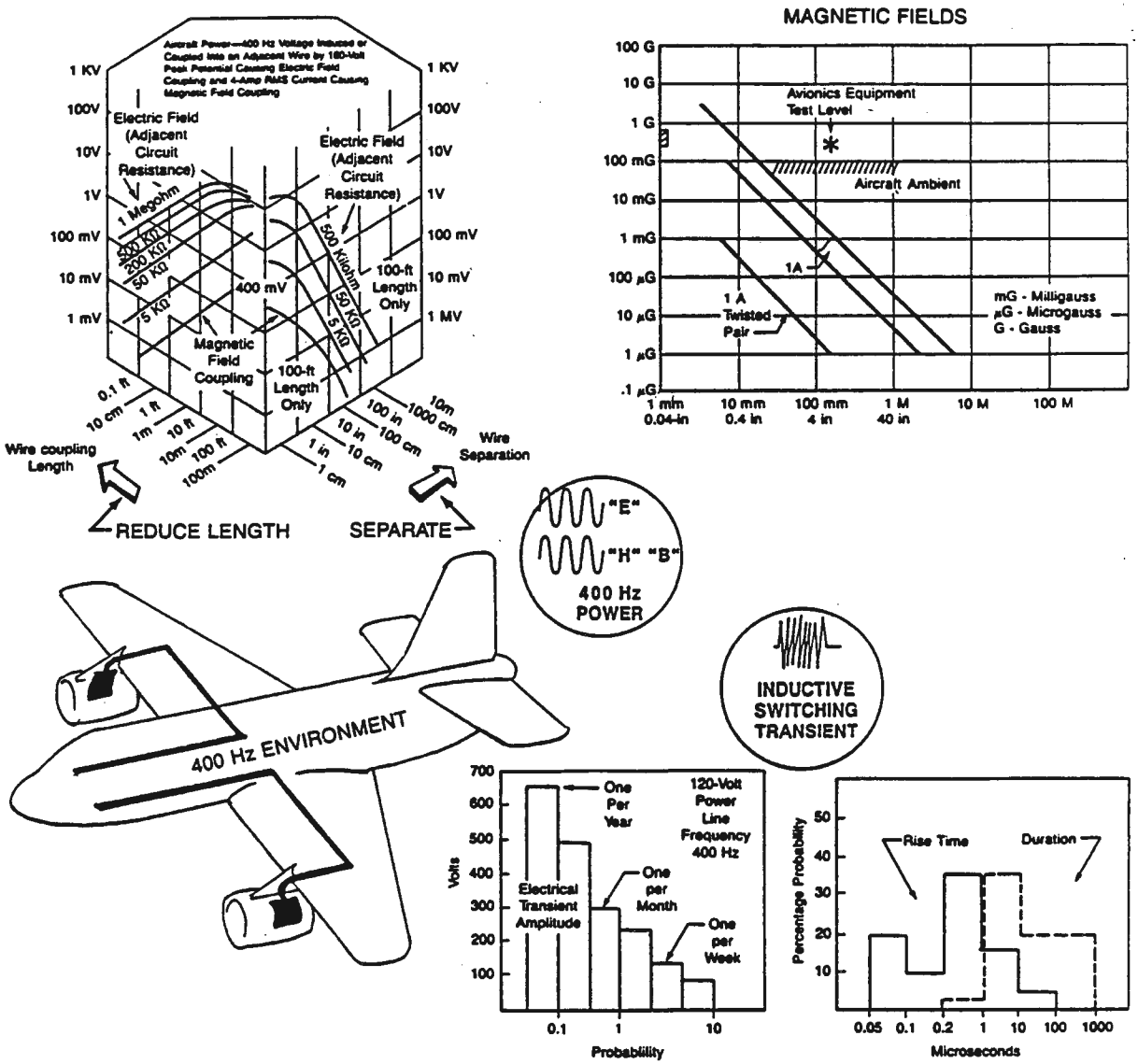



FIGURE 4.3-2. TRADE-WIRE LENGTH/SEPARATION

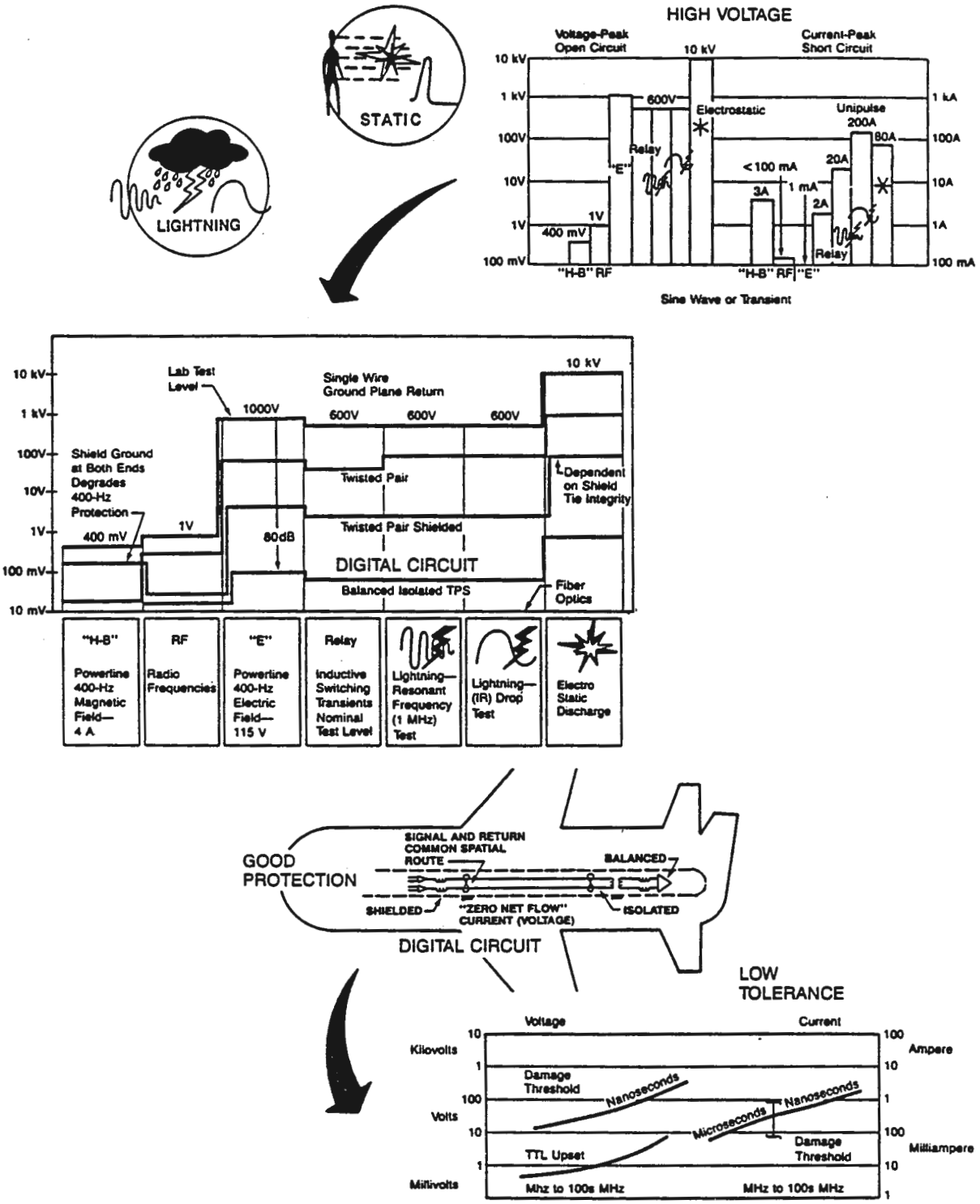



FIGURE 4.3-3. TRADE-DIGITAL CIRCUITS

If it is at all possible to install a cableway, tray or liner, it may be better in the long run. Consider these facts in favor of a tray:

Design:

- Proven designs/models and in-service experience.
- Easily formed, integrated with structure.
- Fewer electronics and "sneak paths".
- Freedom from reliability studies.

Protection:

- Line replaceable units installed with known protection.
- Freedom from frail electronics.
- Excludes high voltages and arcs.
- Rugged and durable.

Test:

- Lower test levels on electronics.
- Freedom from complex verifications.

Life cycle cost:

- Reduced engineering, part control, troubleshooting, maintenance.
- Easy in-service monitoring.
- Long life.

Computerized design procedures will speed the capability to make trades.

## 5. ACTIVITIES AND DOCUMENTS

### 5.1. Specifications/Documents

Some have postulated that most of the electromagnetic compatibility designs and trades at the airframe manufacturer's level must be quickly accomplished before go-ahead or at least at the very beginning of the program, that is: all of the major specifications must be set up - the equipment located and categorized - the environment documented - procurements contracted.

This may not, however, be as extensive an effort as it sounds because most specifications for environments and existing technologies are already available from other programs and need only to be reworked, shaped, and adapted.

Specifications set guidelines-standards-requirements. They are the foundation for any program, and a path that provides a reference and continuity. They help bypass unproductive squabbles and unnecessary changes. They sidestep duplication: "Reinventing the wheel?" They even help tie into the next program.

From the top documents, which outline the intent and scope, down to the lower tier designs, look for these specifications:

Industry:

- FAA Code of Federal Regulations, Aeronautics and Space #14, Parts 1-59; Part 25 Airworthiness Standards: Transport category Airplanes, reference 25.1309, 25.1353, and 25.1431; Part 23 Rotorcraft, 23.1309; Part 27 General Aviation, 27.1309.

- RTCA/DO-160B "Environmental Conditions and Test Procedures for Airborne Equipment," July 20, 1984, sections 1 to 3, and 15 to 22.

- (A generic industry system specification here would be appropriate.)

Program:

- "Program System Specification": Simple one or two paragraph delineation of top EMC references, generic system requirements, or scope.

- "System Electromagnetic Compatibility Design Requirements": Organized sections of system and aircraft program requirements for external environment, equipment categories, interface circuit policy, grounding, bonding, shielding, wiring, isolation, electrostatics, aircraft shield policy, special analyses, applicable EMC document list, and policy for critical equipment.

- "Equipment Electromagnetic Environmental Requirements Specification": Explicit equipment level design and test requirements including test setup and conditions-patterned after the industry standard DO-160.

- All Procurement Specifications: Reference in the design section to the "Equipment Electromagnetic Environmental Requirements Specification?" List of deviations. Requirements for power, dielectrics, circuit interfaces, grounding, bonding, shielding. Table of tests. Statement of Work: Developmental analysis and breadboard test report, analysis of off-theshelf equipment, technical exchange meetings, qualification test plan and report.

- Design Handbook/Guidelines/Design Notes: Interpretation and conversion of program design and test requirements into protection techniques.

- System Test Plan: Test policy and outline. Subsystem tests. Aircraft tests: ground test, environment test, electrical bonding measurements, power switching test, flight test.

Related specifications:

- "Power and Electrical Requirements Specification": Aircraft and external ground power quality, returns and grounding design.

- "Wiring and Shielding Design and Manufacturing Procedure": Wire separation, shield construction.

- "Electrical Bonding and Corrosion Prevention": Materials, compatibility, fasteners, surface preparation, joining and sealing.

5.2. Tasks/Activities

At the beginning of the program, some participants think immediately of the analytical tasks and modeling. Some think of staffing, organization, and people. Others think of schedules. The EMC engineer must pursue the design; he thinks of what the new aircraft is: new materials, new equipment, old equipment, size, layout, wiring runs. Defining the aircraft for electromagnetic compatibility is a monumental effort. The effort cannot be delayed. It must be done quickly. Some details will not be available. There are roadblocks, but these are the engineering tasks.

- Identify in a table or spreadsheet all equipment: new and off the shelf; in-house and vendor; supplying agency; purchasing specification number. List the equipment versus applicable EMC requirements (including outmoded EMC limits), deviations, waivers, and approvals.

- Assess aircraft and equipment architecture: flight deck, electronic bays, externally mounted units, engine units, power and signal circuitry. Identify each equipment and installation and focus on grounding, bonding, shielding, materials and wiring.

- Set up electromagnetic compatibility design categories/groups.

- Track down procurement specifications: remove or add limits and EMC interfaces; establish the developmental, engineering model, qualification, and acceptance tests; communicate with Procurement, Project, and the subcontractor.

- Define topology, field patterns, power of transmitters, and dig out the receiver thresholds.

- Extract power system type, power quality, ground returns and ground points.

- Chase down aircraft advanced technologies and materials. Unearth the differences from past programs.

- Dissect the new environments, external and internal, that are different from DO-160.

- Document new test equipment needs.

With the aircraft definition in hand, the electromagnetic compatibility engineer can take each of the EMC program documents and cut and fit them together: no overlap, no deficiencies, just acceptable coverage. He tailors the requirements based on: (1) past experience, (2) new environments and technologies (3) equipment location, (4) critical circuits, (5) weight, volume, and (6) test. Important: aircraft specifications are finalized more easily before formal design release has been invoked.

Analysis cannot wait. Analysis and models feed on physical dimensions: fuselage-wing-landing gear, length, width, height; and electrical parameters: resistance, capacitance, inductance. Computer models are bringing new capabilities for quick evaluation of multiple design alternatives, and keyboard editing speeds the rapid reevaluation and turnaround when changes are proposed.

The apertures, seams, wire routing, overbraid, isolation, and the shielding effectiveness of aluminum, titanium, graphite-epoxy, and steel must be known along with interface protective designs that must be pulled out of the schematics: filter pins, discrete filters, balanced circuits, transformer isolators, fiber optics. Getting an early handle on these items expedites the many wire coupling and radiation analyses of common noise sources, for instance: power frequencies and ripple; pulse width modulated power; transients from solenoids, valves, motors; and clock oscillator harmonics.

Just a word about responsibility. The electromagnetic compatibility engineer is usually responsible for either a system and product or, conversely, for a technology or even a combination of both. Becoming knowledgeable in a technical area is often the best. For example, for technology:

- Computers/digital circuits software.

- Radio frequency transmitters, receivers, antennas.

- Power quality, generators, switching regulators.

- Analog instrumentation, transducers, sensors, fiber optics.
- Grounding/bonding/shielding/wiring/packaging.
- Dielectrics, corona, and materials.

And for example, product or system:

- Environmental control and cargo (power).
- Flight control/management (digital/software).
- Communication/navigation (radio frequencies).
- Power and lighting and fuel.
- Air data, flight instruments (video, analog, CRTs, displays).

The definition and specification of electromagnetic compatibility on an airplane program is a sporty task. It must be initiated early. Most airplane programs need at least three engineers. Future programs with advanced avionics and flight control systems may require four or five.

# 6. EQUIPMENT SPECIFICATION

## 6.1. New and Existing

The countdown is on until the specifications on every last unit of new and existing equipment are found, cut apart, organized, and set straight. Equipment "procurement specifications" are the bottom line. They define design and test requirements and waivers or deviations. If a unit meets its tailored, allocated electromagnetic compatibility specifications, then electromagnetic compatibility on the aircraft is almost always assured. Deliberate on and wring out every last detail:

- Tailored requirements for each unit (per RTCA/DO-160B).
- Each bonding/grounding/shielding interface (between the supplier and airframe manufacturer).
- Power quality specific requirements.
- Dielectric voltage withstand levels.
- Circuit interface wiring design.
- Detailed unit tests: developmental, engineering model, qualification, acceptance.

The specifications on new and existing units give exact requirements, item by item, on electrical and physical parameters and tolerances.

Existing equipment has already proven itself with its "in-service history," and treatment is different from new equipment. Existing equipment is addressed with attention to conformance to past specifications and any new program requirements. On a new program, old requirements may not be adequate: transients may need redefinition and reapplication depending on unit location; electric field protection may need to be increased; radio frequency susceptibility tests may need strengthening; and power bus momentary interruptions may be a problem. Development tests cannot be overemphasized for new and even existing units. The policy is "make it upset" in order to find margins. Development tests are important and are a major element of successful programs. EMI protection cannot be competently added after the finalized design.

The packaging engineer and circuit designer at the supplier have to know the overall design policy for new and old equipment. They must also know what their interface design is going to be. Key design requirements may be dispersed in various documents but should be either in a system specification or in the equipment procurement specification. Policies or requirements vary from program to program, product to product; these are some standards:

Grounding:

- Designed, controlled, equipotential ground plane system.
- Optional digital circuit ground to case.
- No analog (audio) circuit ground to case without approval.
- Primary to secondary power isolation (transformer electrostatic shield).
- Power single point ground system; "star" architecture.
- No power current in nonmetallic structure.
- EMI requirements apply to returns/grounds of units not on ground plane.

Bonding:

- Case bonded directly to ground plane or,
- Connector pin for case ground wire.
- Backshell/connector bonded to case.

Shielding:

- Internal shield grounded internally.
- External shield grounded externally (to backshell).
- Shields grounded at both ends (except analog).
- No circuit current on shield (except coax).
- Double shield on transmitter lines.

Input/output circuits:

- Interface circuits balanced; clock and data signals routed together.
- Interface circuits isolated; transformer, LED, fiber optics.
- Increased circuit power ratings at outputs.
- Return wires run twisted with the signal wires.
- Connector pin for circuit ground.
- Connector pin for case ground or fault wire.
- Voltage/current limit all interface circuits.

- No shared power returns.

Packaging:

- Separation/isolation of power from signal.
- Power, signal, and ground returns on separate pins.
- Line drivers/receivers close to the connector.
- High-frequency circuits close to the connector.
- Low-frequency circuits at back of the box.
- Connectors on equipment case placed in one local area.

6.2. Aircraft Equipment Categories.

Category definition needs immediate attention. Avionics must not be over or under designed. Equipment categories represent the modification and tailoring of DO-160 requirements to units of equipment having widely variant properties or environment:

- Computers, power generators, transducers, motors.
- Units with extra long interface wiring.
- Units placed on or off the aircraft ground plane.
- Equipment exposed or not exposed to high-energy radio frequency.

Here are some typical categories:

Category 1A:

Energy storage devices (having no electronics): inductors, valves, motors, solenoids, and relays switched continuously or automatically. The items in this category are designed and tested to conducted emission and radiated emission requirements only. They do not require susceptibility tests.

Category 1B:

Energy storage devices operating on an intermittent basis, less than once every three minutes. The conducted emission and radiated emission requirements are raised (relaxed) 20 dB.

Category 1C:

Energy storage devices having short duration transients and operating two times or less per flight. Emission and susceptibility requirements are waived.

Category 2A:

Electrical/electronic equipment: avionics, power equipment, any unit having electronics located within the fuselage and protected by the fuselage metallic structure. All of the standard equipment electromagnetic compatibility requirements, equivalent to DO-160, apply: conducted emission, radiated emission, conducted susceptibility, radiated susceptibility.

Category 2B:

Electrical/electronic equipment within the fuselage, but having long wiring runs, over 100 ft. All standard requirements apply along with a raised (possibly 3 dB) 400-Hz electric/magnetic field and radio frequency field tests.

Category 2C:

Transmitters/receivers: All standard requirements, and with antenna terminal conducted emission and susceptibility tests added.

Category 3A:

Electrical/electronic equipment outside the fuselage and well shielded, but exposed to higher lightning induced transient activity. All standard requirements plus lightning induced transient requirements.

Category 3B:

Electrical/electronic equipment unshielded under nonconductive material or on external mountings. All standard requirements apply and lightning plus increased radio frequency susceptibility test levels (possibly 100 V/m).

Category 3C:

Electrical/electronic equipment not on a ground plane. All standard requirements apply and susceptibility and emission requirements are also applied to the equipment circuit ground, power ground, and case ground interface wires.

Category 4:

Support equipment associated with the aircraft. Apply radiated emission and conducted emission requirements on power that connects to aircraft.

Category 5:

Flight-test equipment: Apply radiated emission requirements and radiated susceptibility requirements for external equipment. Flight equipment is isolated from aircraft circuits.

## 6.3. Suppliers

Early in the program the EMC engineer hastens to know the supplier (vendor or subcontractor). It is important to record and establish a file of equipment identification, electrical characteristics and the tailored requirements:

- Unit: name, acronym, model, unique identifications.
- Company: name, location, engineers, organization.
- Affiliated paper: procurement specification, interface control drawing, statement of work, interface schematics.
- Unit history: new, existing, modified, previous program usage, in-service experience, equipment similarities, deviations, waivers.
- Schedule: technical meetings, prototype/engineering models, test dates, qualification test procedure.
- Grounds/bonds/shields: internal ground, interface bonding.
- Test: development, engineering model, qualification, acceptance.
- Subsystem: project engineer, staff engineer, contracts engineer.

At program startup, there may be specific candidate proposals and subcontractor control plans or procedures to evaluate. These are important attributes to look for:

- Treatment and awareness of specification requirements.
- Unit in-service use on previous programs.
- Past programs experience and success.
- Evaluation and analysis capabilities.
- Implementation of design techniques.
- In-house or provisions for testing facilities.

Of course, the size of the program will dictate the extent of the subcontractor effort, but this might be a typical statement of work:

> "The subcontractor shall conduct an early investigation and developmental EMC analysis and test on the breadboard or prototype hardware for new or modified equipment."
>
> "A developmental test report and evaluation shall be forwarded to the prime contractor containing information on noise measurements of interface circuits and proposed wiring, protection, and emission control techniques."

"The subcontractor shall obtain available EMC data and in-service history on existing, off-the-shelf equipment; determine compliance with the program requirements; and, where requirements are not met, recommend options for equipment modifications, deviations, or waivers."

"The subcontractor shall conduct timely technical exchange meetings and present grounding, shielding, hardware, and software protection. A qualification test plan and report shall be submitted for comment and approval."

The statement of work for the subcontractor may be more or less detailed depending on complexity and size of the equipment or subsystem.

Subcontractor equipment design changes are made throughout the entire program. An EMC engineer cannot monitor or assess all engineering changes on all equipment, but a change involving electrical properties may require some reevaluation or retest. For example, anytime a change deals with new coupling characteristics, that is: wiring harness rerouting, new input/output circuits, different circuit board layout, extensive software modification that changes timing sequence or adds new loops, then an analysis or susceptibility retest is appropriate. The vendor maintains complete responsibility and accountability for design and test verification of his unit or subsystem to fully utilize his own facilities, experience, test equipment and software capabilities.

# 7. VERIFICATION AND VALIDATION

## 7.1. Key EMC Designs

Verification: Equipment qualification test at a supplier to prove that a design meets specification. The qualification test of equipment forms the very keystone of verification.

Verification is also accomplished in the airframe manufacturer's laboratories to prove performance of key EMC designs and properties, such as:

- Grounding of digital or analog circuits.
- Bonding resistance of aircraft part, wire, or joint.
- Shielding properties of new materials.
- Leakage of joints and apertures.
- Emissions of pulse width modulated power.
- Wiring design and coupling conditions.
- Aspects of aircraft mounted filters and limiters.
- Susceptibility thresholds of interface circuits.

Verification tests quickly lead to the ultimate proper operation on the aircraft. Measurement reveals thresholds and noise upset margins. Verification lays a groundwork before validation.

Validation: Demonstration of and confidence in proper equipment function along with acceptable control of noise during aircraft operation in its intended environment.

Demonstration - A two part measurement and test:

- Operation of an aircraft (or subsystem) through its normal modes while operating equipment and devices.
- Introduction of jeopardizing noise environments to stress equipment: all the while monitoring functioning flight deck instruments and aircraft circuit operation internally and externally: internally with built in test circuits/equipment (BITE) and externally with meters, transient recorders, oscilloscopes, spectrum analyzers, and digital bus analyzers.

An airplane program must be structured on a foundation of solid verification and validation (see section 9).

## 7.2. Validation Plans

First, hasten to collect, document, and rely on existing qualification data, existing validated technologies, and validation by similarity.

Otherwise, validation tests can be run at the subsystem level in a flight avionics laboratory or on the production aircraft.

When considering subsystem versus aircraft test, assess the following deficiencies and benefits in cost, effectiveness, flexibility, and timeliness:

TABLE 7.2-1. SUBSYSTEM VERSUS AIRCRAFT TEST

| COST: | CAPITAL EQUIPMENT | ATTENDANT PERSONNEL | EQUIPMENT REWORK | TEST EQUIPMENT | LEARNING CURVE |
|---|---|---|---|---|---|
| Subsystem | Very Low | Low | Low | Low | Low |
| Airplane | High | High | High | High | High |
| **ENVIRONMENT:** | **EQUIPMENT COLOCATION** | **RF SIM-ULATION** | **GROUNDS/ SHIELDS** | **RESONANT CONDITIONS** | |
| Subsystem | Poor | Good | Poor | Different | |
| Airplane | Actual | Good | Actual | Actual | |
| **DESIGN:** | **FINAL HARDWARE** | **FINAL SOFTWARE** | **TEST SOFTWARE** | **FINAL WIRING** | **ENGINEER AVAILABLE** |
| Subsystem | Good | Good | Excel | Fair | Good |
| Airplane | Excel | Excel | Excel | Actual | Fair |
| **PROCEDURE:** | **APPROVAL SIGNATURES** | **FORMAL SEQUENCE** | **DECISIONS/ DIAGNOSTICS** | **CHANGE CONTROL** | **PROGRAM TIMING** |
| Subsystem | Few | No | Best | Good | Fair |
| Airplane | Many | Yes | Fair | Good | Late |

Aircraft offers fidelity, but demands a high price with inflexibility in a dedicated airplane outfitted with test equipment. The avionics laboratory is an active testing and simulation facility and testing can often be done speedily on a "noninterference" basis when test time is slack. Circuit access panels already exist. Testing can be easy, informal, flexible and promote diagnostics and real-time knowledge of noise behavior. A range of tests can be run at the subsystem level with selected tests on the aircraft.

## 7.3. Validation Procedure

The subsystem or aircraft configuration, setup, test equipment, and modes are identified and recorded paragraph by paragraph for each separate mode. The procedure then records step-by-step operation of equipment.

CONFIGURATION: Model, make, serial numbers, outstanding engineering changes or deviations, wiring or customer configuration, software, test software, test date, location.

TEST SETUP: Equipment bay lights, test stands, ground planes, shield ties, ground wires, unique simulations, monitors.

TEST EQUIPMENT: List test equipment and installation for the following systems by paragraph section:

- Flight control/flight management: computer breakout boxes, digital bus analyzers, oscilloscopes.
- Communication/navigation: interphone electronics breakout boxes, receiver breakout boxes, interphone headsets, RF voltmeters, spectrum analyzer.
- Power: power supply breakout boxes, chart recorders, transient analyzers, peak reading (transient) voltmeters.
- Flight indicating/recording: flight instrument breakout boxes, digital bus analyzers.
- Engine: monitors, indicators.
- Environment: magnetic field measuring instruments, spectrum analyzer and current probe (wiring) and antennas (radiated emission), signal generators.

PERSONNEL: Knowledgeable subsystem or equipment engineers, experienced EMC engineers and technicians. Monitor and record all initial settings, indicator light status, measurement instrumentation, BITE readouts. On functioning displays, observe and record status of flight instruments, CRT panels on flight deck, left, right, and center control panels, power panels. During step-by-step procedure, record upsets, circuit breaker disconnects, state changes/events, autopilot disconnect, warning signals/flags, annunciations, CRT distortion on all subsystems.

SOFTWARE: Identify software installation and initiation for each mode. Automated test software routines exercise processing functions and interfaces and monitor the data, status, error counters, or fault logging in real-time or on printout.

MODES OR PRESETS: Establish modes and settings for all systems:

- Environmental control: pressurization, temperature, air control.
- Flight control/management: mode control panel, autopilot engage at "left or right command"; test software installation, flight management computer; control display unit, flight path program, takeoff, cruise, landing, autoland; engine control operation.

- Communication/navigation: ADF, HF, VHF, ATC transmit/receive frequency modes and settings, IRS navigation modes, selector panel interphone left and right settings.

- Power: circuit breakers engaged/disengaged.

- Fuel and hydraulics: pumps, valves.

- Flight instruments and recorders: flight instrument, caution/advisory readouts, recorders.

- Lighting: light settings, dimmers, strobes.

- Engine: electronic engine control settings, indicators.

- Degraded modes: loss of redundant unit, low battery.

FUNCTIONAL SWITCHING TEST PROCEDURE: Establish a step-by-step procedure paragraph for each mode and possibly for each major subsystem. For chosen mode perform "functional switching test?" That is: operate, switch on/off, cycle, or exercise all units:

- Environmental control: pressurization, temperature, fans.

- Flight control/flight management: motors, actuators, flaps, slats.

- Communication/navigation: HF-VHF transmit.

- Power: all circuit breakers.

- Fuel and hydraulics: pumps, valves, solenoids.

- Flight indicating/recording: flight instruments, CRTs, recorders

- Lighting: cabin, flight deck, strobe, and landing lights.

- Engine: electronic controls, indicators, actuators, indicators.

A subsystem or airplane EMC validation test procedure demonstrates the absence of malfunction or undesirable noise and authenticates that the repeatability, accuracy, and reality of operation in normal modes of an airplane baseline production configuration meets the Federal Aviation Regulations and the Aircraft System Specification.

ENVIRONMENTAL STRESS TEST PROCEDURE: Establish separate paragraph for each mode or subsystem as above and operate equipment while introducing transients or radio frequency energy. The policy is "make it upset" not "show success?" If noise is transient, vary the pulse repetition rates relative to computer processing control cycles. Equipment should operate within performance specification and also demonstrate:

- Maintenance of proper internal configuration.

- Successful automatic restart.
- Continued operation.
- No permanent faults.
- No adverse control surface motion.
- Automatic transfer to secondary systems.

NOISE MEASUREMENT: Measure and characterize the signature of noise on the digital data bus, communication circuits or cables, interphone circuits, powerline bus, analog instrument circuits, and in the aircraft ambient to assist in validating operation and establishing a data base for diagnostics, not only for the existing program, but for future programs. Establish a separate paragraph for each mode. The following are noise types or thresholds:

- Clock oscillator harmonics or HF-VHF transmitter signals on circuits and cables.
- ADF-HF-VHF threshold sensitivity; passenger address circuits, voice recorder, crew interphones noise and thresholds.
- Powerline switching transients at circuit breakers and equipment terminals, switching regulator harmonics on buses and on cables.
- 400-Hz electric and magnetic field strengths, HF-VHF field strengths.

BONDING RESISTANCE TEST: Measure resistance of structure joints, skin panels, electrostatic paints, liners, foils, doors, flight deck panels, strut fairings and wing.

Testing and test assessment rests so heavily on a product's history and the intent of the product's function and future use, that it is, of course, an individual or program responsibility to define test concepts, requirements, and procedures. This validation test information, therefore, represents possible guides or techniques, but must be considered as not being appropriate for a specific test.

## 7.4. Program Design Reviews

During a program there is no quicker way to help improve the validity and timeliness of the design and specifications than by periodic reviews to establish agreement on perceived requirements and outcome. The following items are for consideration and might be selected and presented in a program preliminary design review or critical design review to help lay groundwork for verification, validation and certification:

- Program documentation, organization.
- Aircraft system and RTCA/DO-160 design/test requirements.

- Environment assessment, equipment EMC categories.
- Subsystem, equipment architecture/topology.
- Grounding system plan.
- Bonding: radio frequency/static/safety.
- Shielding: aircraft and wiring
- Wiring design, critical circuits.
- New technologies analysis/models.
- Equipment/subsystem/aircraft verification/validation plan.

The prompt review of key documentation and requirements helps to start and keep the program on a successful path.

## APPENDIX A - TEST AND TEST LIMITS

Early EMC history has seen the radio receiver as the centerpiece of electromagnetic field testing. It was necessary to find the source of noise entering aircraft audio circuits and radios. The keys today, in measuring emission and susceptibility parameters, are the oscilloscope and spectrum analyzer; the need for noise control spread to instrumentation and automatic pilot circuits.

We are now seeing the need for digital logic analyzers, bus controllers, and automatic, interactive testing capability to measure performance in fly-bywire, negative stability, aircraft systems.

Spectrum analyzers and storage oscilloscopes are proven and powerful troubleshooting tools in the measurement of noise. The scope displays the analog time domain waveform. The spectrum analyzer lays out the frequency components of that waveform. The scope shows peak amplitude. The spectrum analyzer reveals amplitudes versus frequency. With a scope, the pulse repetition rate is measurable. And with the spectrum analyzer, harmonics are caught in any frequency band. But, although spectrum analyzers and digitizing oscilloscopes are useful in finding and characterizing noise, it is ineffective to apply them to the task of detection of errors or noise margins in computers. The EMC community now needs these units to be paired with high-speed data bus analyzers and logic analyzers in order to provide even simple measurements of computer processor operations, such as: timing sequence, state activity, and bus status.

Recently, the processing functions of an avionic computer were upset by noise causing the unit to suspend operation, to "lock up?" It then required recycling of power to restart operation. Personnel from the electromagnetic compatibility group were asked to help diagnose the unpredictable upsets. An oscilloscope (analog waveforms) and the spectrum analyzer (frequency components) displayed noise occurring on the wire returns, grounds, and logic power supply lines; the noise levels were high enough to interfere with clock and data signals; wavering clock timing pulses destabilized the processing sequences; noise and unstable signals appeared everywhere on the circuit boards traces.

Many days were spent on this problem, but with "analog" instrumentation a solution could not be found until logic analyzers and bus controllers were brought in. Without the capability to control data entry formats, observe and evaluate output activity, timing sequences, and to correlate state changes with noise events, timely solutions to complex problems become impossible.

Future validation testing on automated aircraft systems will require laboratory personnel to monitor simultaneously occurring events, to visually correlate timing, logic state changes, and to automatically record data, decoded and converted, in real-time under a number of aircraft modes. Digital interface bus

analyzers, logic analyzers, and interactive computer controllers will offer fast solutions and bring about professional insights to noise problems on new digital avionics.

Laboratory personnel are called upon to execute a variety of tasks:

- Circuit research.
- Test equipment construction.
- Diagnostics and troubleshooting.
- Evaluation of avionics performance.
- Avionics and airplane qualification testing.

And laboratory personnel diagnose problems in a variety of aircraft units: computers, power controllers, transmitter/receivers amplifiers, motors/generators, analog sensors, all of which encompass a wide range of characteristics. Functions of the aircraft, in a flight context, span the systems of environmental control, flight control, flight management, fuel, communication, navigation, power, and engine controls, but to the EMC engineer and laboratory technician these functions bring to mind susceptibilities and emissions:

- Power: 400-Hz, dc buses, motors, relays, switching events.
- Dc-to-dc switching regulators, pulse width modulation controllers.
- Radio frequency receiver thresholds and transmitter antennas fields.
- Data transmission and clock oscillators.
- Sensitive analog (audio) sensors and circuits.

Data, information, and histories already exist on environmental levels and are documented in the RTCA/DO-160B and MIL STD 461 specification (see table A-1).

But, for noise effects on data, transmission and timing sequences relative to conditions of circuit stability and upset margins, the EMC community today does not have an adequate data base.

Speeds of future systems will increase, voltage levels will rise, and sensor and receiver thresholds will be lower and more sensitive. Emerging flat panel displays, microprocessor controls, voice controlled systems, dc power systems, and pulse width modulated, electric actuators will be available soon for full authority flight controls and will be beyond the reach of engineers in the EMC community for test and analysis.

In the shield room, the controlled environment and standard electrical references (for instance, voltage, current, impedance, frequency meters, and ground planes) offer established laboratory conditions for analysis and tradeoffs of digital flight control designs. But, most of the time during

today's EMI tests, current probe factors, antenna factors, line losses, attenuation factors, and bandwidth conversions are now essentially hand calculated and joined with raw data almost on a point-by-point basis in an anachronistic, time-consuming process. These computations are a hindrance, but the more important loss is inability to assess trends and compare, in real-time, circuit operational changes in a controlled and repeatable manner under various noise levels.

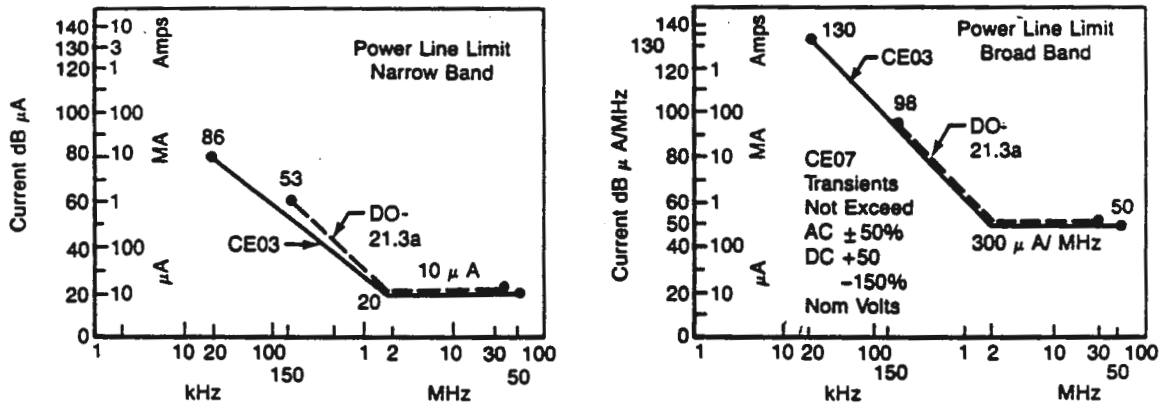
Controlling, probing, comparing, and recording status and data transmissions (being digital in nature) against the itinerant noise occurrences (having analog waveforms) stands or falls on the test and simulation capability.

Effectiveness resides in the technician's skill and his experience coupled with the tools and equipment with which that skill and experience is implemented. New, modern equipment having automatic microprocessor controls and automatic data readout is becoming available to provide interactive test decisions. Equipment and tools of the "analog" 1960s and 1970s cannot carry the load or be compatible with the flight controls of the "full-authority," "fly-by-wire" 1990s.

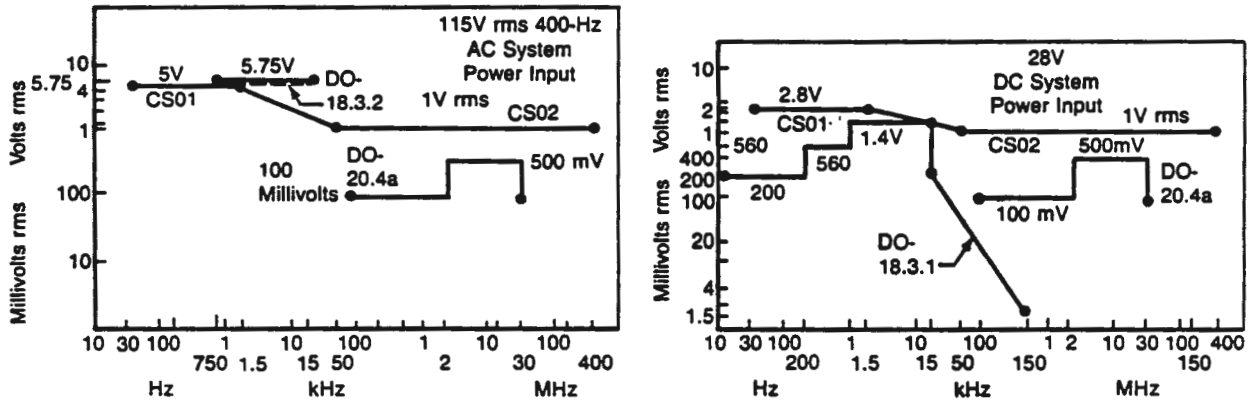
TABLE A-1. DO-160B AND MIL STD 461B. TEST CROSS REFERENCE
(see figure A-1)

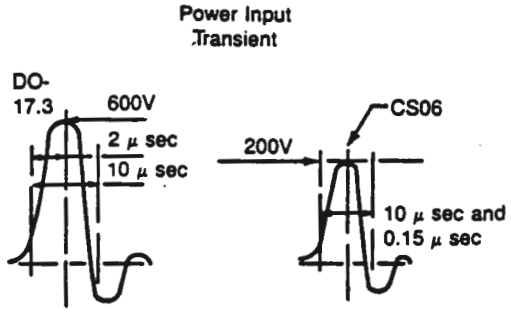
| TEST | 160B | 461B |
|---|---|---|
| POWER BUS: | | |
| Conducted Emission | 21.3a | CE03 |
| Switching Transients CE | None | CE07 |
| Conducted Susceptibility (CS): | | |
| Audio Frequency CS | 18.3 | CS01 |
| Radio Frequency CS | 20.4a | CS02 |
| Power Line Spike | 17.3 | CS06 |
| Bus Momentary Interruption | 16.5.1.4 | None |
| EQUIPMENT AND SIGNAL CABLE: | | |
| Conducted Emission Cable | 21.3b | None |
| Induced into equipment and cable: | | |
| Magnetic "H" Field Equipment | 19.3.1 | RS02 |
| Magnetic "H" Field Cable | 19.3.2 | RS02 |
| Transient Spike: 200V | None | RS02 |
| Electric "E" Field Cable | 19.3.3 | None |
| Inductive Switching Transient | 19.3.4 | None |
| Radio Frequency CS Cable | 20.4b | None |
| EQUIPMENT & INTERCONNECTING CABLE: | | |
| Radiated Emission | 21.4 | RE02 |
| Radiated Susceptibility | 20.5 | RS03 |

POWER LINE CONDUCTED EMISSION




CONDUCTED SUSCEPTIBILITY

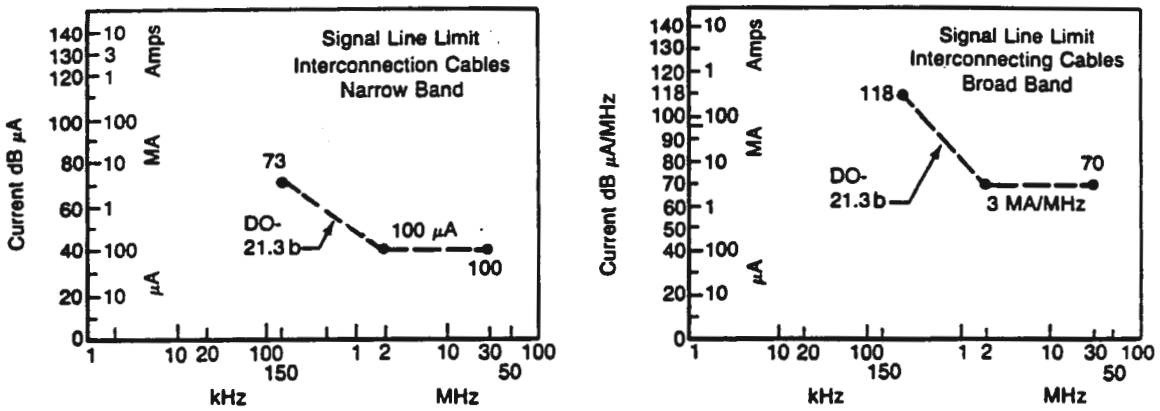


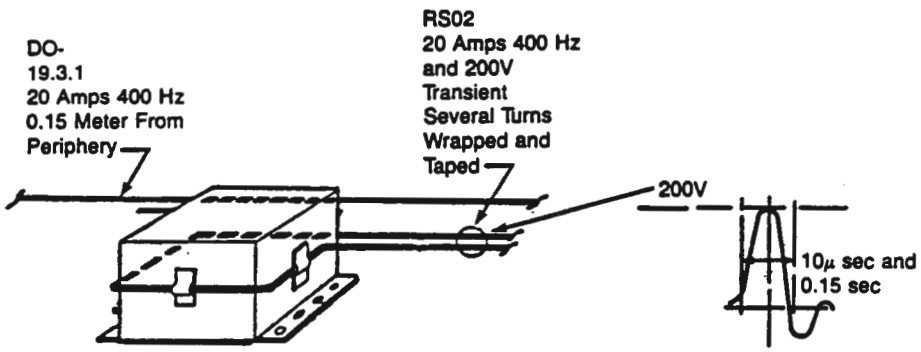

Note: The "DO-" indicates the RTCA/DO-160B paragraph.

FIGURE A-1. EMI TEST LIMITS (1 of 4)

SIGNAL INTERCONNECTING CABLES CONDUCTED EMISSION




INDUCED FIELDS—EQUIPMENT-CABLE




SIGNAL INTERCONNECTING CABLE
MAGNETIC ("H") FIELD

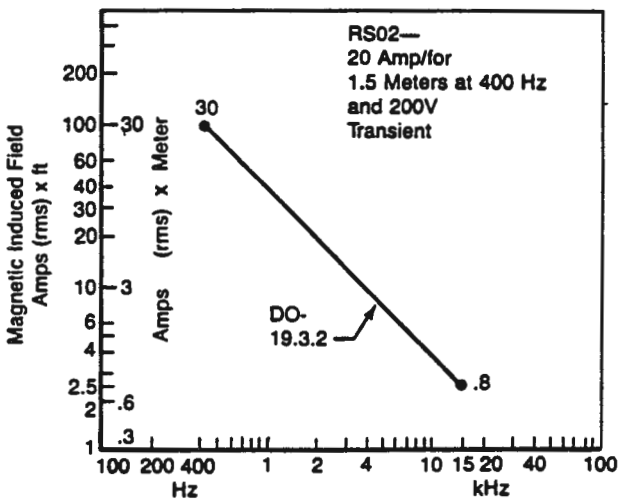
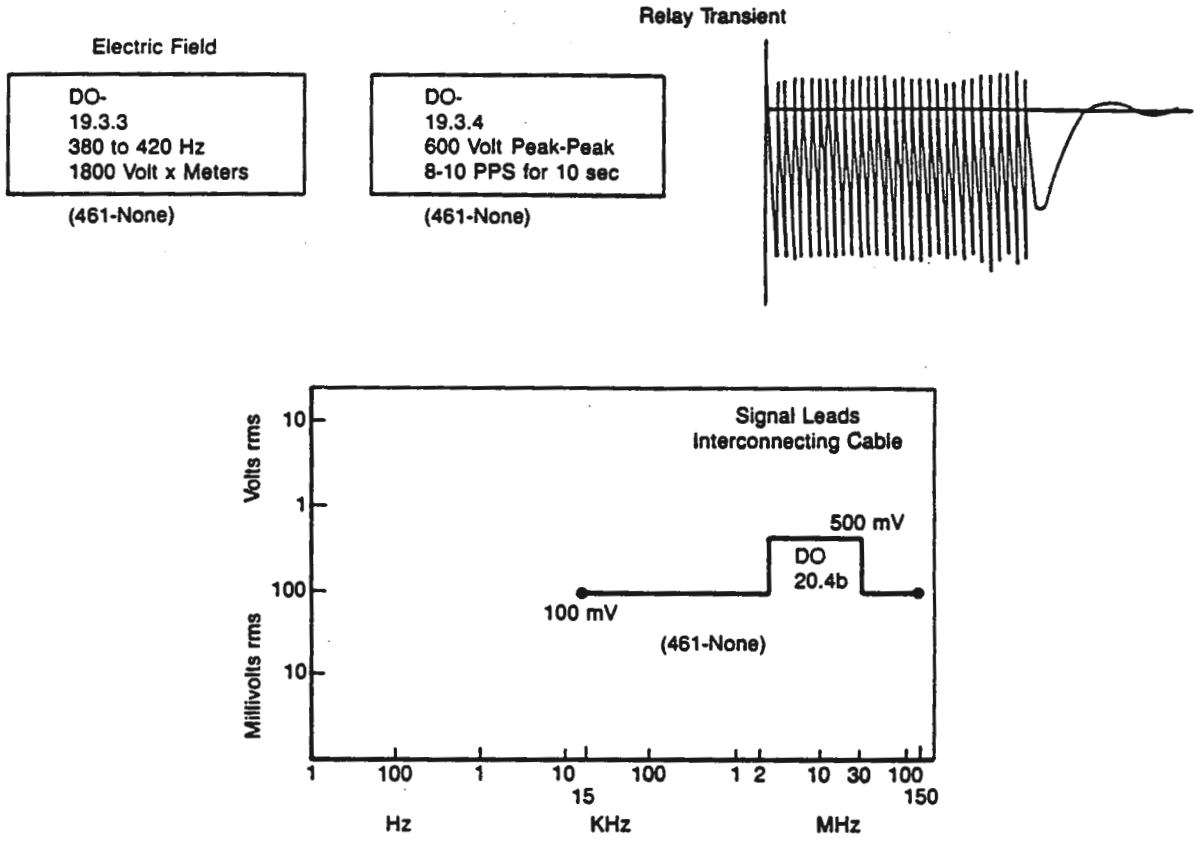



FIGURE A-1. EMI TEST LIMITS (2 of 4)

INDUCED FIELD—EQUIPMENT-CABLE (CONTINUED)




RADIATED EMISSION

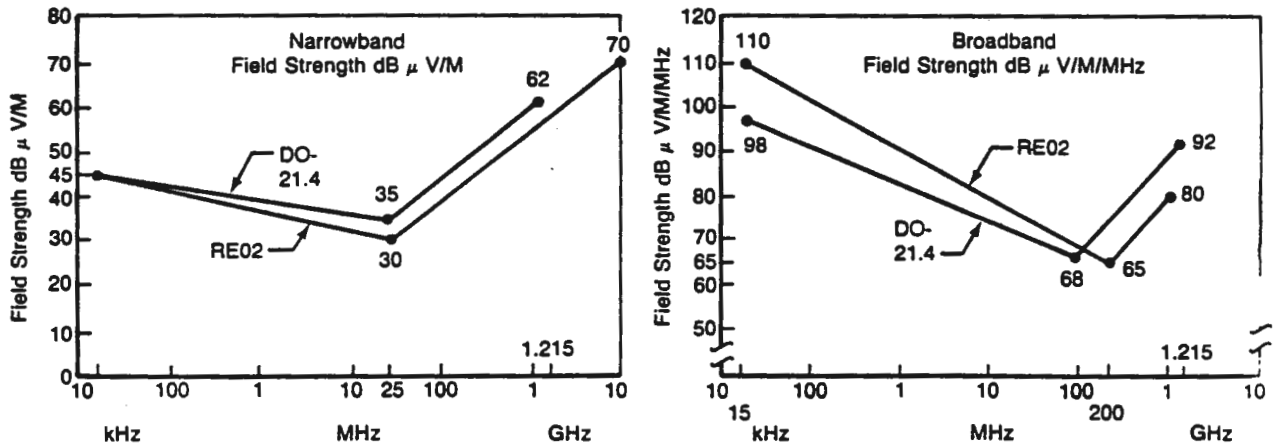



FIGURE A-1. EMI TEST LIMITS (3 of 4)

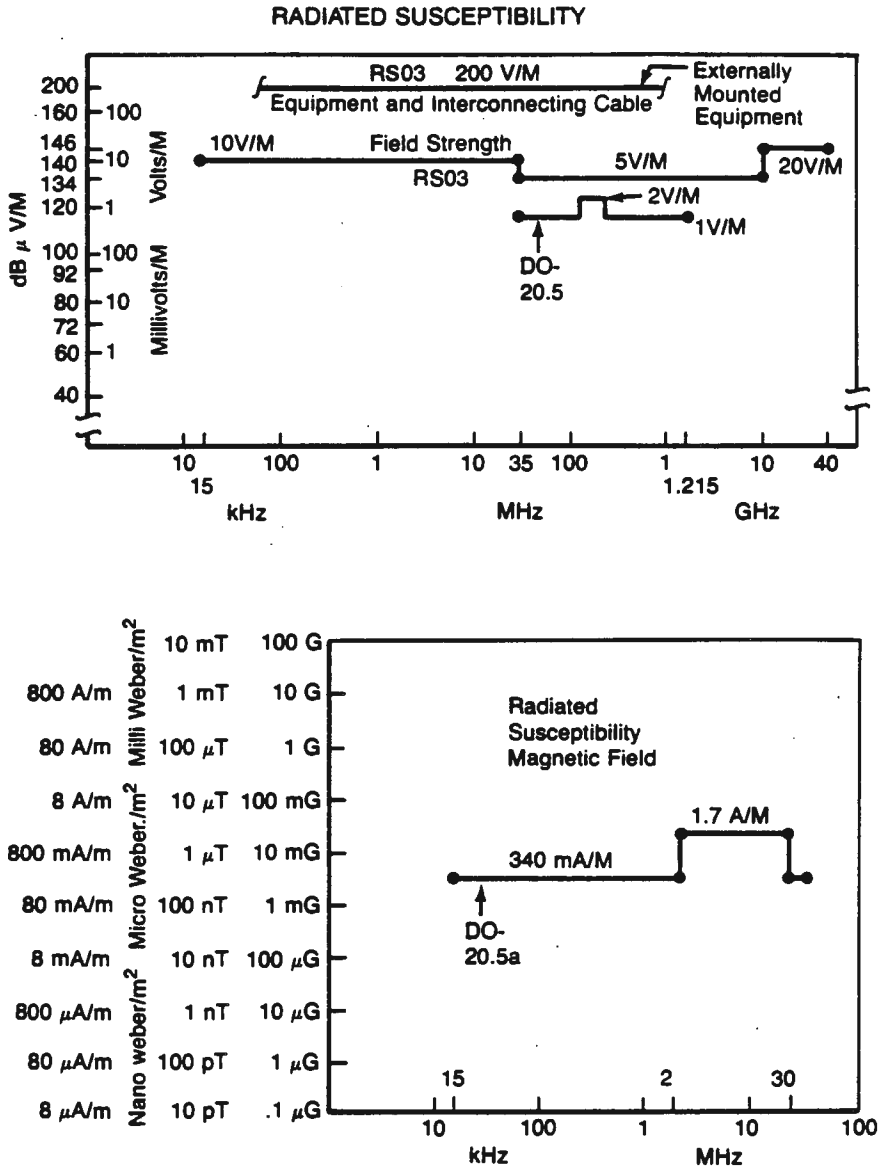



FIGURE A-1. EMI TEST LIMITS (4 of 4)

## BIBLIOGRAPHY


AC 25.1309-1, Advisory Circular-Systems Design Analysis, Federal Aviation Administration, September 7, 1982.

AFSC DH 1-4, Air Force EMC Design Handbook, January 10, 1972.

Ahmad, S., "Shielding and Aging Effects With Flexible Coaxial Cable," 1985 IEEE EMC Symposium.

Annanpalo, Jaakko, "Common-Mode Interference Rejection in Electrically Short Twisted Pairs," EMC Technology, September 1986.

ANSI C95.1-1982, "American National Standard Safety Levels With Respect to Human Exposure to Radio Frequency Electromagnetic Fields, 300 Kilohertz to 100 Gigahertz."

ARINC 429-8, "Digital Information Transfer System (DITS)," October 31, 1978.

Bachman, L., "An Assessment of Shipboard Power Line Transients," 1981 IEEE EMC Symposium.

Baker, D. M., D-18306, Magnetic and Electrostatic Wire Coupling in the Audio Frequency Range, The Boeing Company, July 1956.

"Belden-Wiring and Shielding Data," Belden Electronic Wire and Cable, Co., Richmond, Indiana.

Belisle, K. M., "EMI Design Techniques for De-coupling and Isolation of Microcircuits, 1983" IEEE EMC Symposium.

Biegon, R., "EMC Characteristics of RS-232 Cable Assemblies," Connection Technology, January 1986.

Birken, J. A., NAVAIR AIR-518-7, Notebook on Electromagnetic Properties of Composite Materials Below 1 GHz, September 1981.

Bly, R. P., "The Inside and the Outside Are Not the Same - Experimental Investigations of Ground and Shield Topology," 1982 IEEE EMC Symposium.

Boaen, V., "Designing Logic Circuits for High Noise Immunity," IEEE Spectrum, January 1973.

Bodnar, G. D., "Shielding Effectiveness Measurements on Conductive Plastics," 1979 IEEE EMC Symposium.

Brettle, J., "Electrical Bonding in Aircraft," 1979 IEEE EMC Symposium.

Brooks, P., Physical Design and Electronic Packaging Part 1, EDN, September 5, 1973.

Carney, R. L., D180-27423-49, Part IV "Design Guides for Air Vehicles," February 1987.

Chesworth, E. T., "Electromagnetic Interference Control in Structures and Buildings," EMC Technology, January-February 1986.

Clayton, R. E., "Transmission Line Electromagnetic Compatibility," 1975 IEEE EMC Symposium.

Code of Federal Regulations (CFR) 14, Part 25 Airworthiness Standards, "Transport Category Airplanes, Sub Part F Equipment:"

25.1301, Function and Installation
25.1309, Equipment, Systems, and Installations
25.1351, General
25.1353, Electrical Equipment and Installations
25.1431, Electronic Equipment

Cowdell, R. B., "Simple Equations Compute Radiated Emissions," 1983 IEEE EMC Symposium.

Crawford, M. L., "Comparing EM Susceptibility Measurement Results Between Reverberation and Anechoic Chambers," 1985 IEEE EMC Symposium.

Creason, R. R., "Measurement of the Magnetic Field From Long Two-Wire Lines at Low Frequencies," 1983 IEEE EMC Symposium.

DeMario, W. F., "New World for Aerospace Composites," Aerospace America, October 1985.

Dillon, W. L., "Antenna to Antenna Analysis of the E-4A Advanced Airborne Command Post," 1974 IEEE EMC Symposium.

Ditton, V. R., "Coupling to Aerospace Cables at Microwave Frequencies," 1975 IEEE EMC Symposium.

Dixon, D. S., "Low Frequency Radiated Magnetic Field Emissions: Rationale for a MIL STD 461B, RE-01 Limit Change," 1984 IEEE EMC Symposium.

DOD-HDBK-263, Electrostatic Discharge Control Handbook for Protection of Electrical and Electronic Parts, Assemblies, and Equipment (Excluding Electrically Initiated Explosive Devices),"May 2, 1980.

D6-16018, Electric Bonding and Grounding Design Requirements, The Boeing Company.

D6-16050-2, Electromagnetic Interference Control Requirements, The Boeing Company.

Don White Consultants, Inc., DM-S81, "Syllabus EMC - Design and Measurement for Control of EMI," March 9, 1981.

Dubil, E. F., Electrostatic Discharge Special Supplement Douglas Service, Vol. 42, October 1985.

Final Environmental Impact Statement for Seattle City Light, Highline 230 KV Transmission Project.

ECAC Study, A320 Electromagnetic Environment (A320 EME), by IIT Research Institute, (Contract F19628-85-C-0071 prepared for Federal Aviation Administration).

Force, R. D., "Integrated Application of Active Controls (IAAC) Technology to an Advanced Subsonic Transport Project," ACT/Control/Guidance System Study-Volume I, NASA CR-165963, December 1982.

_____., Report D180-20186-4, Investigation of Effects of Electromagnetic Energy on Advanced Composite Aircraft Structures and Their Associated Avionic/Electrical Equipment, September 1977.

Gibbons, R., "Performance Analysis Helps Designers Fine Tune Software," Electronic Design, October 18, 1984.

Glancy, D., "Preventing EMI in ATE Systems," Test and Measurement World, January 1987.

Gormady, J., "Random Susceptibility of an IC 7400 TTL Nand Gate," 1983 IEEE EMC Symposium.

Hafer, J. W., "The Effects of Shield Grounding Techniques for Isolation to Electromagnetic Waves," 1981 IEEE EMC Symposium.

Hariya, E., "Instruments for Measuring the Electromagnetic Shielding Effectiveness," 1984 IEEE EMC Symposium.

Hart, A. R., LSI Design Considerations for ESD Protection Structures Related to Process and Layout Variations, Hewlett Packard Company.

Heirman, D. N., "Education and Training of the Industrial Regulatory Compliance Test Team," 1981 IEEE EMC Symposium.

Hitt, E. F., DOT/FAA/CT-82-115, Validation of Digital System Avionics and Flight Control Applications, December 1982.

Hjellen, G. A., "A Thermal Damaged Model for Bi-Polar Semi-Conductors," IEEE EMC Symposium.

Hoeft, L. O., "A Simple Theory for Predicting the Electromagnetic Performance of Enclosures," 1985 IEEE EMC Symposium.

_____., "Current Division and Shielded Conductive Cables," 1983 IEEE EMC Symposium.

_____., "How Big a Hole is Allowable in a Shield," 1986 IEEE EMC Symposium

_____., "Measured Magnetic Field Reduction of Copper Sprayed Wood Panels," 1985 IEEE EMC Symposium.

Jarrett, R., "Software Fault Tolerance Staves Off the Errors that Besiege Microprocessor Systems," Electronic Design, August 9, 1984.

IPC-D-317, "Design Standard for Electronic Packaging Utilizing High Speed Techniques," August 1985.

ITEM Interference Technology Engineers' Master, Robar Industries, Inc., R & B Enterprises Division.

Kashyap, S., "Feed Cable Resonance in a TEM Cell," 1985 IEEE EMC Symposium.

Kendall, C. M., DOT/FAA/CT-83/49, Aircraft Generated Electromagnetic Interference on Future Electronic Systems, December 1983.

_____., "Data Processing Grounding A Need For Circuit Isolation," 1982 IEEE EMC Symposium.

_____., "EMC Control in Mainframe Computing Systems," 1985 IEEE EMC Symposium.

_____., "Microfiltering of Input/Output Cables," 1978 IEEE EMC Symposium.

Ketterer, J. R., "The Navy F/A-18A Hornet Electromagnetic Compatibility Program," 1981 IEEE EMC Symposium.

Koeritz, K. W., "A Systems and Environmental EMC Control Program for the Airtrans Automated Ground Transportation System," 1974 IEEE EMC Symposium.

Larsen, W. E., IEEE Transactions on EMC, "A Modified Ebers-Moll Transistor Model for RF Interference Analysis," November 4, 1979.

_____, Paper No. 84-2605-CP, 6th DASC, "An Overview of the Digital Avionics Assessment Activities Being Conducted by the FAA at NASA-AMES," December 1984.

_____, DOT/FAA/CT-84/9, The Effect of Aircraft Generated Electromagnetic Interference (EMI) on Future Avionic Systems A Compendium, April 1984.

Liao, S. Y., "Light Transmittance and RF Shielding Effectiveness of a Metallic Film Coating on a Plastic Substrate," 1977 IEEE EMC Symposium.

Lowe, K., "Noise-margin Analysis Automatically Lays Bare Hidden Logic Problems," Electronic Design, October 18, 1984.

Madle, P. J., Cable and Connector Shielding Attenuation and Transfer Impedance Measurements Using Quadraxial and Quintaxial Test Methods, 1975 IEEE EMC Symposium.

March, D. N., "Siting Considerations for Industrial Facilities That Generate Environmental Electromagnetic Noise," 1980 IEEE EMC Symposium.

Martin, A. R., "Shielding Effectiveness of Long Cables," 1979 IEEE EMC Symposium.

McAteer, O. J., "Shocking Blow to Military Electronics," Military Electronics/Counter-Measures, June 1979.

McBrayer, P., "RF Compatibility-Environment to Component Part," IEEE EMC Symposium.

McConnell, R. A., DOT/FAA/CT-87/19, Contract NAS2-12448, Avionics System Design for High Energy Fields.

MIL-HDBK-235-1A, Electromagnetic (Radiated) Environment Considerations for Design and Procurement of Electrical and Electronic Equipment, June 23, 1972.

MIL-HDBK-35 (USAF), Management and Design Guidance Electromagnetic Radiation Hardness for Air Launched Ordinance Systems, January 15, 1981.

MIL-STD 461B, Electromagnetic Emission and Susceptibility Requirements for the Control of EMI, April 1, 1980.

MIL-STD 1250 (MI), Corrosion, Prevention, and Deterioration Control in Electronic Components and Assemblies, March 31, 1967.

MIL-STD 1553B, Digital Databus System.

MIL C 5541A, "Chemical Films and Chemical Film Materials for Aluminum and Aluminum Alloys," March 31, 1964.

Morgan, G. E., "Examples of System Engineering in the EMP Hardening of Facilities and Aircraft," 1983 IEEE EMC Symposium.

Morgan, M. G., "Power Line Fields and Human Health," 1985 IEEE EMC Symposium.

Moser, J. R., IEEE Transactions on EMC, "Peripheral Cable Shield Termination: The System EMC Kernel," February 1986.

National Research Council, Aeronautics Technology Possibilities for 2000: Report of a Work Shop, Aeronautics and Space Engineering Board, Commission on Engineering and Technical Systems, 1984.

Olsen, R. G., "A Simple Model for Weakly Coupled Lossy Transmission Lines of Finite Length," 1984 IEEE EMC Symposium.

Ott, H. W., "Digital Circuit Grounding and Interconnection," 1981 IEEE EMC Symposium.

Palmgren, C. M., "Shielded Flat Cables for EMI and ESD Reduction," 1981 IEEE EMC Symposium.

Paul, C. R., "Affect of Pigtails on Coupling to Shielded Wires," 1979 IEEE EMC Symposium.

_____., "Coupling of Electromagnetic Fields to Transmission Lines," 1981 IEEE EMC Symposium.

_____., "Prediction of Cross Talk in Flat Pack, Coaxial Cables," 1984 IEEE EMC Symposium.

_____., "Printed Circuit Board Cross Talk," 1985 IEEE EMC Symposium.

_____., "Sensitivity of Multiconductor Cable Coupling to Parameter Variations," 1974 IEEE EMC Symposium.

Regan, J. J., Plastics Technology, "EMI Shielding: What You Need to Know And Why," January 1980.

Rhoades, W. T., "Achieving ESD Equipment Protection With Emission Controls," 1985 IEEE EMC Symposium.

_____., "Designing Commercial Equipment for Conducted Susceptibility," 1979 IEEE EMC Symposium Records.

_____., "Development of Power Main Transient Protection for Commercial Equipment," 1980 IEEE EMC Symposium.

RTCA/DO-160B, Environmental Conditions and Test Procedures for Airborne Equipment, July 20, 1984.

Schneider, L. M., "Noise Source Equivalent Circuit Model for Off-line Converters and Its Use in Input Filter Design," 1983 IEEE EMC Symposium.

"Shielding Against Electromagnetic Interference," Plastics Design Forum, March/April 1979.

Shimayama, T., "Measurement of the Suppression Characteristic of Filter Network," 1984 IEEE EMC Symposium.

Shores, M. W., "EMC Language in Perspective," 1981 IEEE EMC Symposium.

Small, J., Document (to be issued), Study of the Non-Damage Effects of Lightning on Avionics Systems, The Boeing Company.

Sommer, D. L., AFWAL-TR-81-2117, Protection of Advanced Electrical Power Systems from Atmospheric Electromagnetic Hazards, December 1981.

_____., D180-27423-17, Part 3, "Atmospheric Electricity Hazards Balanced Protection Schemes," September 1984.

Strawe, D., D180-18879-1 (AFWL-TR-75-141), Interaction of Advanced Composites With Electromagnetic Pulse (EMP) Environment, September 1975.

Tell, R. A., "Recent Results on Determining Population Exposure to VHF and UHF Broadcast Radiation in the United States," 1979 IEEE EMC Symposium.

Tenning, C. B., T6-2408, Inductive Switching Transients on the KC-135 Airplane, The Boeing Company, March 1964.

Thomas, D. E., "Measurements and Calculations of the Cross Talk Due to Capacitive Coupling Between Connector Pins," 1983 IEEE EMC Symposium.

Turner, T. E., Electrostatic Sensitivity of Various Input Protection Networks, Mostek Corp.

Vance, R. D., ITEM 1977, Magnetic Shielding.

Violette, M. F., EMC Technology, Vol. 5, Number 2, "EMI Control in the Design and Layout of Printed Circuit Boards," April 1986.

Weinstock, G. L., Electromagnetic Integration of Composite Structure in Aircraft, McDonnell Aircraft Company.

Weinstock, G. L., Intra-Vehicle Electromagnetic Compatibility Analysis, AFAL-TR--71-155, Part 1, July 1971.

White, D. R. J., "Building Attenuation and the Impact on Products Susceptibility," 1974 IEEE EMC Symposium.

_____., "EMI Control in the Design of Printed Circuit Boards," EMC Technology, January 1982.

_____., Taming EMI in Microprocessor Systems, IEEE Spectrum, December 1985.

Whittlesey, A. C., "Electric Welding Hazard to Spacecraft Electronics," 1981 IEEE EMC Symposium.

Woody, J. A., "Modeling Techniques for Discrete Passive Components to Include Parasitic Effects in EMC Analysis and Design," 1980 IEEE EMC Symposium.

Zajac, H., Study of Effects of Electrostatic Discharge on Solid State Devices, Tektronix, Inc.

Zenter, J. C., "Aircraft EMC Problems and Their Relationship to Subsystem EMI Requirements, ASD, EMC and Power Branch," WPAFB, Ohio, Proceedings IEEE, Vol. I, National Aerospace and Electronics Conference, May 17, 1983, Dayton, Ohio.

## GLOSSARY OF TERMS

ABSORPTION LOSS. Attenuation or retention of electromagnetic energy passing through a material, a shield. Absorption loss and reflection loss contribute to total shielding effectiveness (SE).

ANODIZE. A preparation by electrolytic process that deposits a protective oxide, insulating film on a metallic surface (aluminum). The oxide defeats electrical bonding. Alodine and iridite finishes on aluminum are conductive.

APERTURE. An opening, such as a nonconductive panel joint, slot or crack, allowing electromagnetic energy to pass through a shield.

AUDIO FREQUENCY (AF). The spectrum (20 to 20,000 Hz) of human hearing, often defined as extending from approximately 20 Hz to 50 kHz and sometimes to 150 kHz. Audio noise is nuisance hum, static, or tones from power line 400 Hz, switching regulator and digital clock harmonics, or HF,VHF transmitter frequencies.

BACKSHELL. Metal shell connecting circuit shields or overbraid to an electrical connector.

BALANCED CIRCUIT. A signal, acting line-to-line, between two conductors having symmetrical voltages identical and equal in relation to other circuits and to ground. "Differential mode" is line-to-line; "common mode" is line to ground.

BANDWIDTH (BW). Frequencies bounded by an upper and lower limit in a given band associated with electronic devices, filters, and receivers.

BOND, ELECTRICAL. Electrical connection at two metallic surfaces securely joined to assure good conductivity often 2.5-m maximum for electrical/electronic units and 1Ω for electrostatic dissipation or safety. A "faying surface" bond maintains contact between relatively large or long surfaces. Inherently bonded parts are permanently assembled and conductivity exists without special preparation: such as with welding, brazing.

BRAID, OVERBRAID. Fine metallic conductors woven to form a flexible conduit or cableway and installed around insulated wires to provide protection against electric fields and radio frequencies. Best when peripherally connected to backshells. A grounding strap/jumper may be made of braid.

CABLE OR HARNESS. A bundle of separate, insulated, electrical circuits, shielded or unshielded, usually long and flexible and having breakouts, terminations, overbraid, and mounting provisions completely assembled.

CABLEWAY. A solid metallic housing (liner, foil, coating) surrounding and shielding insulated electrical conductors. Also called conduit, tray, or

raceway. Crosswise or transverse openings or breaks in the metallic cableway cause noise voltages to be transferred to internal wire circuits.

COMMON MODE (CM) IMPEDANCE. Impedance or resistance shared by two or more circuits so that noise voltages/currents generated by one are impressed on the others.

COMMON MODE REJECTION. The ability of wiring or an electronic device to reject common mode (line-to-ground) signals and maintain fidelity of differential mode (line-to-line) signals.

COMMON MODE SIGNAL. Identical and equal signals on input conductors or at the terminals of a device relative to ground.

CONDUCTED EMISSION (CE) OR INTERFERENCE. Voltage/current noise signals entering or leaving a unit on interface conductors-emission is the general term, interference is undesired noise.

COUPLING. The transfer of energy between wires or components of a circuit electrostatically, electromagnetically, or directly.

CROSS COUPLING (CROSSTALK). Transfer of signals from one channel, circuit, or conductor to another as an undesired or nuisance signal or the resulting noise.

DAMAGE. The irreversible failure of a component.

DECIBEL (dB). Decibel expresses the ratio between two amounts of power, P1 and P2, at two separate points in a circuit. By definition, the number of dB = 10 log to the base 10 of (P1/P2). For special cases, when a standard power level P2 = 1 mW or 1 W or 1 kW, then the ratio is defined as "dBm," "dBw," or "dBKW." Because $P = V^2/R$ and also $I^2R$, decibels express voltage and current ratios. Ideally, the voltages and currents are measured at two points having identical impedances. By definition, dB = 20 log V1/V2 and dB = log I1/I2. For convenience, V2 or I2 are often chosen as 1 $\mu$V or 1 $\mu$A and the ratio is defined as dB above a $\mu$V or dB above a $\mu$A when graphing emission or susceptibility limits.

DIELECTRIC STRENGTH. Voltage withstand capability that an insulating material sustains before destructive arcing and current flow, usually expressed in volts per mil thickness. Dielectric withstand voltage is the voltage level at which insulation breakdown occurs.

DIFFERENTIAL MODE (DM) SIGNAL. The signal in a two-wire circuit measured from line-to-line.

DUAL GROUND. Equipment case ground/return through two independent circuit paths to structure implemented in flammable zones and water leakage areas-each path meeting electrical conductivity (resistance) requirements.

ELECTRIC FIELD. High-impedance, radiated voltage field, positive or negative, from a voltage source as contrasted to a low-impedance magnetic field from a current source.

ELECTROMAGNETIC COMPATIBILITY (EMC). Operation within performance specification in the intended electromagnetic interference environment.

ELECTROMAGNETIC INTERFERENCE (EMI). Conducted and radiated voltage/current noise signals, broadband (BB) or narrow band (NB), that degrade the specified performance of equipment.

ELECTROSTATIC CHARGE. Electric potential energy with a surrounding electric field, uniform or nonuniform, moving or at rest, on a material.

EMISSION. Voltage/current noise on a wire or in space. Broadband emission has uniform spectral energy over a wide frequency range and can be identified by the response of a measuring receiver not varying when tuned over several receiver "bandwidths." Or, energy present over a bandwidth greater than the resolution bandwidth where individual spectral components cannot be resolved. Broadband (BB) may be of two types: (1) impulse and coherent varies 20 dB per decade of bandwidth and (2) random or statistical, varies 10 dB per decade. A narrow band (NB) emission or signal, sometimes called continuous wave, occurs at a discrete frequency and does not vary with bandwidth.

FAULT CURRENT. The maximum current (magnitude and duration) flowing through a fault point-equal to the supply voltage divided by the dc resistance of power line leads, circuit breakers, and the current return in wire or structure.

FILTERING. Device or unit that passes or rejects a frequency band and designed to block noise from entering or leaving a circuit or unit.

GROUND. A generic term having multiple meanings and indicating a circuit return path or a voltage reference: not "zero" voltage reference. Four hundred millivolts of noise voltage is common on "quiet" grounds. There are several types of returns and references.

Return:

- Structure, for power, fault, and "discrete" circuits.
- A grid of wires, solid sheet, or foil.
- A wire from circuit load back to source or to case.
- Circuit card "ground plane," also a reference and shield.

Reference:

- Structure, for electronics, shields, power.
- A grid of wires, solid sheet, or foil.
- A wire from circuit to grounding block or case.
- A wire from circuit to structure.

- Shield tie.
- Earth.

IMMUNITY. Capability of a circuit or unit to operate within performance specification in a specified electromagnetic interference environment.

ISOLATION. Electrical separation and insulation of circuits from ground and other circuits or arrangement of parts to provide protection and prevention of uncontrolled electrical contact.

JUMPER/STRAP. A short wire, strip, strap, or braid conductor installed to make a safety ground connection, to dissipate electrostatic charge, or establish continuity around a break in a circuit.

LIMITING, VOLTAGE/CURRENT. Semiconductor components, diodes, Transorb, or filter designed to clip and shunt to ground an applied transient or steadystate voltage. Used to protect against noise frequencies, faults, lightning, and inductive switching transients.

MAGNETIC FIELD. A radiated, low-impedance field having lines of "flux" or magnetomotive force associated with an electrical current.

MALFUNCTION. Failure or degradation in performance that compromises flight safety.

NOISE. Conducted or radiated emission causing circuit upset, performance disorder, or undesired sound.

PRECIPITATION STATIC (P-static). Electrostatic discharge, corona, arcing, and streamering, steady state or impulsive, causing circuit upset, receiver noise or component damage.

RADIATED EMISSION (RE). Electromagnetic energy transmitted and propagated in space usually considered as audio frequency or radio frequency noise.

RADIO FREQUENCY (RF). Frequencies in the electromagnetic spectrum used for radio communications extending from kilohertz to gigahertz.

RADIO FREQUENCY INTERFERENCE (RFI). Electromagnetic interference in the radio frequency range.

SEALANT. An applied substance enclosing and protecting the integrity of a joint, fastener, or electrical bond from moisture, contaminants, oxidation, and acid or alkaline corrosion.

SHIELD. A conductive material, opaque to electromagnetic energy, for confining or repelling electromagnetic fields A structure, skin panel, case, cover, liner, foil, coating, braid, or cable-way that reduces electric and magnetic fields into or out of circuits or prevents accidental contact with hazardous voltages.

SHIELD EFFECTIVENESS (SE). The ability of a shield to reject electromagnetic fields. A measure of attenuation in field strength at a point in space caused by the insertion of a shield between the source and the point.

SIGNAL RETURN. A wire conductor between a load and the signal or driving source. Structure can be a signal and power return. Commonly, it is the low voltage side of the closed loop energy transfer circuit.

SINGLE-ENDED CIRCUIT. A circuit with source and load ends grounded to case and structure and using structure as return.

STRUCTURE. Basic members, supports, spars, stanchions, housing, skin panels, or coverings that may or may not provide conductive return paths and shields for electrical/electronic circuits.

SUSCEPTIBILITY. Upset behavior or characteristic response of an equipment when subjected to specified electromagnetic energy-identified with the point, threshold, or onset of operation outside of performance limits. Conducted Susceptibility (CS) applies to energy on interface conductors; Radiated Susceptibility (RS) to radiated fields.

THRESHOLD, NOISE. The lowest electromagnetic interference signal level that produces onset of susceptibility.

UPSET. Temporary interruption of performance that is self-correcting or reversible by manual or automatic process.

UNACCEPTABLE RESPONSE. Upset, degradation of performance, or failure, not designated a malfunction, but is detrimental or compromising to cost, schedule, comfort, or workload.

UNDESIRABLE RESPONSE. Change of performance and output, not designated a malfunction or safety hazard, that is evaluated as acceptable as is because of minimum nuisance effects and excessive cost burdens to correct.

VALIDATION. Demonstration and authentication that a final product operates in all modes and performs consistently and successfully under all actual operational and environmental conditions founded upon conformance to the applicable specifications.

VERIFICATION. Demonstration by similarity, previous in-service experience, analysis, measurement, or operation that the performance, characteristics, or parameters of equipment and parts demonstrate accuracy, show the quality of being repeatable, and meet or are acceptable under applicable specifications.

## ACRONYMS AND ABBREVIATIONS

| | |
|---|---|
| A/C | Aircraft |
| ACARS | ARC Communications Addressing and Reporting System |
| ACT | Active Controls Technology |
| ADC | Air Data Computer |
| AF | Audio Frequency |
| ADF | Automatic Direction Finder |
| AFCS | Automatic Flight Control System |
| ANSI | American National Standards Institute |
| APU | Auxiliary Power Unit |
| ARC | Aeronautical Radio, Inc. |
| ASDS | Airport Surface Detection System |
| ATCRBS | Air Traffic Control Radar Beacon System |
| BB | Broadband |
| BITE | Built-In Test Equipment |
| BW | Bandwidth |
| CAS | Criticality Advisory System |
| CDU | Control Display Unit |
| CE | Conducted Emission |
| CM | Common Mode |
| CRT | Cathode Ray Tube |
| CS | Conducted Susceptibility |
| DFDAU | Digital Flight Data Acquisition Unit |
| DFDR | Digital Flight Data Recorder |
| DITS | Digital Information Transfer System |
| DM | Differential Mode |
| DME | Distance Measuring Equipment |
| E | Electromagnetic Environmental Effects |
| EADI | Electronic Attitude Director Indicator |
| ECAC | Electromagnetic Compatibility Analysis Center |
| ECS | Environmental Control System |
| E/E | Electrical/Electronic |
| EEC | Electronic Engine Control |
| EED | Electro-Explosive Device |
| E-FIELD | Electric Field |
| EFIS | Electronic Flight Instrument System |
| EGT | Exhaust Gas Temperature |
| EHSI | Electronic Horizontal Situation Indicator |
| EICAS | Engine Indication and Crew Alerting System |
| EM | Electromagnetic |
| EMC | Electromagnetic Compatibility |
| EME | Electromagnetic Effects |
| EMI | Electromagnetic Interference |
| EMIC | Electromagnetic Interference/Compatibility |
| EMP | Electromagnetic Pulse |
| EPR | Engine Pressure Ratio |
| ESD | Electrostatic Discharge |

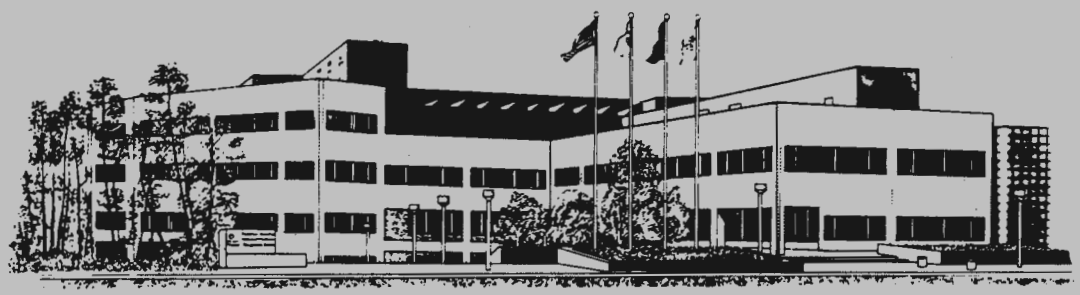
| | |
|---|---|
| ESE | Electric (field) Shield Effectiveness |
| FCC | Flight Control Computer |
| FDEP | Flight Data Entry Panel |
| FMC | Flight Management Computer |
| Gr/Ep | Graphite/Epoxy |
| GPS | Global-Positioning-System |
| GPWS | Ground Proximity Warning System |
| HF | High-Frequency |
| H-FIELD | Magnetic Field |
| IAAC | Integrated Application of Active Controls Technology (to an Advanced Subsonic Transport Project) |
| IDG | Integrated Drive Generator |
| ILS | Instrument Landing System |
| INS | Inertial Navigation System |
| IRS | Inertial Reference System |
| LCC | Life Cycle Cost |
| LOC | Localizer |
| LRRA | Low Range Radio Altimeter |
| LRU | Line Replaceable Unit |
| MCDP | Maintenance Control and Display Panel |
| MCP | Mode Control Panel |
| mil | One thousandths of an inch (0.001) |
| MLS | Microwave Landing System |
| MSE | Magnetic (Field) Shielding Effectiveness |
| NB | Narrow Band Signal |
| Hl | Fan Speed |
| N2 | Core Engine Speed |
| OMEGA | Very Low Frequency Navigation |
| PCU | Power Control Unit |
| PRF | Pulse Repetition Frequency |
| P-static | Precipitation Static |
| PWM | Pulse Width Modulation |
| RDMI | Radio Distance Magnetic Indicator |
| RE | Radiated Emission |
| RF | Radio Frequency |
| RFI | Radio Frequency Interference |
| RS | Radiated Susceptibility |
| RTCA | Radio Technical Commission for Aeronautics |
| S/A | Spectrum Analyzer |
| SE | Shielding Effectiveness |
| SHF | Super High-Frequency |
| TCAS | Traffic Alert and Collision Avoidance System |
| TLA | Thrust Lever Angle |
| TMC | Thrust Management Computer |
| TTL | Transistor-Transistor-Logic |
| UHF | Ultra High-Frequency |
| VHF | Very High-Frequency |
| VLF | Very Low-Frequency |
| VOR | VHF Omnidirectional Range |
| VORTA/VHF | Omnirange/Tactical Air Navigation |
| VSI | Vertical Speed Indicator |
| WRU | Weapons Replaceable Unit |

U.S. Department of Transportation
Federal Aviation Administration

DOT/FAA/CT-88/10

# HANDBOOK-VOLUME II
# DIGITAL SYSTEMS VALIDATION

## CHAPTER 12
## FAST RISE-TIME ELECTRICAL TRANSIENTS IN AIRCRAFT



PREPARED BY:

COMPUTER RESOURCE MANAGEMENT, INC.
950 HERNDON PARKWAY, SUITE 360
HERNDON, VIRGINIA 22070

PREPARED FOR:

FEDERAL AVIATION ADMINISTRATION
TECHNICAL CENTER
ATLANTIC CITY INTERNATIONAL AIRPORT, NEW JERSEY 08405

NOTICE

This document is disseminated under the sponsorship of the U.S. Department of Transportation in the interest of information exchange. The United States Government assumes no liability for the contents or use thereof.

The United States Government does not endorse products or manufacturers. Trade or manufacturers' names appear herein solely because they are considered essential to the objective of this report.

TABLE OF CONTENTS


| Section | | Page |
|---|---|---|
| 1. | INTRODUCTION | 12-1 |
| | 1.1. Topic History | 12-1 |
| | 1.2. Topic Overview | 12-1 |
| | 1.3. Function Criticality | 12-2 |
| 2. | TRANSIENT ANALYSIS: THE DOUBLE EXPONENTIAL PULSE | 12-5 |
| | 2.1. A Distinction between Lightning, NEMP, and ESD | 12-5 |
| | 2.2. Response of the Aircraft to Lightning and NEMP | 12-7 |
| 3. | ENERGY TRANSFER FROM THE PULSE TO THE SYSTEM | 12-11 |
| | 3.1. An Energy Transfer Example | 12-12 |
| 4. | FREQUENCY SPECTRUM OF THE SYSTEM RESPONSE | 12-13 |
| 5. | ELECTROSTATIC DISCHARGE | 12-19 |
| | 5.1. Energy Transfer in ESD | 12-19 |
| | 5.2. Description of the ESD Pulse | 12-21 |
| | 5.3. Direct Discharge ESD | 12-21 |
| | 5.4. Prediction of ESD Radiated Electric Fields | 12-25 |
| | 5.5. Prediction of Voltage Induced on a Susceptible Circuit | 12-33 |
| | 5.6. ESD Induced Voltages on Cables | 12-35 |
| | 5.7. Summary of ESD Prediction Technique | 12-35 |
| | 5.8. ESD Testing Recommendations | 12-36 |
| 6. | CONDUCTED FAST TRANSIENTS ON POWER LINES | 12-39 |
| 7. | RADIATED SUSCEPTIBILITY TO MAGNETIC INDUCTION FIELDS | 12-41 |
| 8. | MIL-STD-461C EMP TESTS | 12-45 |
| 9. | PROTECTIVE MEASURES | 12-47 |
| BIBLIOGRAPHY | | 12-49 |
| GLOSSARY | | 12-50 |
| ACRONYMS AND ABBREVIATIONS | | 12-51 |

## LIST OF ILLUSTRATIONS


| Figure | | Page |
|---|---|---|
| 2-1 | DOUBLE EXPONENTIAL PULSE | 12-6 |
| 2.2-1 | NEMP CURRENT | 12-9 |
| 4-1 | FREQUENCY SPECTRUM OF A 1 MHz DAMPED SINUSOID | 12-14 |
| 4-2 | FREQUENCY SPECTRUM OF A DAMPED SINUSOID WITH TWO RESONANCES | 12-17 |
| 5.2-1 | ESD VOLTAGE PREDICTION | 12-22 |
| 5.3-1 | HUMAN ESD MODELING | 12-23 |
| 5.3-2 | ESD TEST CIRCUIT | 12-24 |
| 5.3-3 | ESD DIRECT DISCHARGE EQUIPMENT CONFIGURATION | 12-31 |
| 5.5-1 | INDUCED VOLTAGE ON A LOOP FROM ESD | 12-34 |
| 5.6-1 | ESD TEST CONFIGURATION (CABLE INDUCED) | 12-36 |
| 6-1 | POWER LINE TRANSIENT TEST WAVESHAPES | 12-40 |
| 7-1 | RADIATED SUSCEPTIBILITY, MAGNETIC INDUCTION FIELD, CABLE TEST | 12-42 |
| 7-2 | RADIATED SUSCEPTIBILITY, MAGNETIC INDUCTION FIELD, CASE TEST | 12-43 |


## LIST OF TABLES


| Table | | Page |
|---|---|---|
| 5-1 | TYPICAL ELECTROSTATIC VOLTAGES | 12-20 |
| 5.3-1 | ESD PARTS BY PART TYPE | 12-26 |
| 5.3-2 | PART CONSTITUENTS SUSCEPTIBLE TO ESD | 12-29 |
| 5.4-1 | PREDICTION OF RADIATED EMISSIONS | 12-32 |
| 5.8-1 | ESD TESTS | 12-37 |

# 1. INTRODUCTION

## 1.1. Topic History

Fast rise-time transients have always had potentially serious effects upon aircraft structures, wiring, and systems. With the application of fly-by-wire control systems to engines and control surfaces, an understanding of the nature and effects of fast transients is now even more essential for certifi-cation authorities.

The measures, which can be taken to protect aircraft and their systems against fast transients, must also be understood for certification authorities to decide whether an aircraft is adequately protected.

Solid state devices and digital systems have improved reliability by orders of magnitude compared to the vacuum tubes, analog systems, and electromechanical devices which they replaced. Unfortunately, solid state devices and digital systems are far more susceptible to electromagnetic interference than were the older devices. Therefore, much care must be given early in the design phases of new systems to ensure against upset or damage from a wide variety of electro-magnetic fields generated both by the aircraft systems themselves and by sources external to the aircraft.

This chapter describes and analyses some of the fast rise-time transients to which modern aircraft are exposed. Estimates are made of the energy levels and field strengths which these transients are capable of generating. These energy levels and field strengths are compared to the typical upset and damage thres-holds which exist in aircraft systems and devices. Various methods of testing systems for susceptibility to fast transients are described. Finally, the measures which can be taken to protect systems and devices from these transients are described.

## 1.2. Topic Overview

Fast rise-time transients include the Nuclear Electromagnetic Pulse (NEMP), lightning, and Electrostatic Discharge (ESD). The primary power lines of aircraft are also subject to fast transient disturbances. These types of transients will also be described.

Historically, there has been a large difference in the levels of electromagnetic protection applied to civil and military aircraft. Civil aircraft are subject to the rather modest requirements of Radio Technical Commission for Aeronautics (RTCA) DO-160B. Military aircraft are subject to the rigorous requirements of MIL-STD-461C and MIL-STD-462.

Recently there has been widespread recognition that the electromagnetic environment is similar for many military and civil aircraft. (One notable

exception is the very severe environment to which Navy carrier aircraft are exposed.) Ground-based radar systems to which aircraft are exposed are much the same at military and civilian airports. Many airports are jointly used by both civil and military aircraft. Air traffic control routes and procedures expose civil and military aircraft to a similar electromagnetic environment. In the future, the Electromagnetic Compatibility (EMC) requirements for civil aircraft will be close to the requirements for military aircraft.

The potential for exposure of civil aircraft to the NEMP is a debatable point, but the civil air fleet constitutes a logistical reserve for the military. More importantly, the electrical characteristics of NEMP, lightning, and ESD are so similar that a description of all three and a comparison of their characteristics is useful as a tutorial method. Furthermore, the methods used to protect aircraft against all these transients have much in common.

In this chapter, military specifications and test methods are referenced. It is recognized that these specifications and methods do not apply directly to civil aircraft, but a thorough knowledge of the military approach is most important for understanding the evolving field of aircraft EMC.

The effects of lightning are addressed in chapter 13 of this handbook, and therefore will not be covered in depth in this chapter. The characteristics which the lightning transient has in common with other fast transients will, however, be explored.

## 1.3. Function Criticality

The fast transients described have the capability of upsetting or damaging a variety of systems which are critical to the continued safe flight and landing of the aircraft.

For junction type devices (which include transistors, integrated circuits, and diodes) it is usual to assign a burnout level of 100 microjoules. This energy level can easily be exceeded when an aircraft is subjected to NEMP, lightning, or ESD.

The consequences of circuit damage are obvious: the circuit function will be lost or at least impaired. Backup systems may have to be called into service.

Circuit upset, short of damage, may also occur. Transistor-Transistor Logic (TTL) circuits can be upset by spurious voltages of 0.4 volts. The electromagnetic fields generated within the aircraft by fast transients can generate voltages which readily exceed this value.

There are several possible consequences of circuit upset. An automatic computer controlled reset may occur. A manual reset by a flight crew member may be necessary. Or it may be necessary to completely power down a system and restart it.

The consequences of upset or damage depend, of course, on what the affected system is controlling and how long it takes to reset the system or bring backup controls into operation. Obviously, loss of engine or flight surface controls

may be extremely critical depending upon the phase of flight. Loss of navigation or communications systems is less critical. The presence of backup systems, such as on present day fly-by-wire engine controls, greatly affects the criticality.

## 2. TRANSIENT ANALYSIS: THE DOUBLE EXPONENTIAL PULSE

Lightning, NEMP, and ESD can all be described in terms of the double exponential pulse. The common characteristics of all three phenomena are that they display a fast rise-time followed by a much slower fall-time. Such a pulse is shown in figure 2-1.

For each type of pulse, the rise- and fall-times are different, and the maximum voltages or currents are different, but the pulse shapes are similar.

The NEMP has a rise-time of about 5 nanoseconds and a fall-time of perhaps 600 nanoseconds. Lightning has traditionally been assigned a rise-time of 1 to 2 microseconds and a fall-time of 50 to 100 microseconds. However, recent investigations have shown that a significant proportion of lightning discharges have rise-times in the range of 200 nanoseconds or less. The ESD pulse has a typical rise-time (10 percent to 90 percent) of 2 nanoseconds, and a fall-time to the 50 percent level of 200 nanoseconds.

The ESD pulse has an amplitude of from 1 to 30 kilovolts. The NEMP pulse has an amplitude of about 50 kilovolts, and the voltage induced by lightning on an aircraft structure can range up to about 60 kilovolts, although the peak voltage of the lightning strike itself may be much higher. The voltages induced on aircraft wiring by lightning and NEMP are much smaller, on the order of 600 to 1500 volts, and voltages of this magnitude are often specified in pin level tests of system cabling.

Mathematically, the voltage of the double exponential pulse can be represented as:

$$V = V_0 \left(e^{-at} - e^{-bt}\right) \qquad (1)$$

where the fast rise is governed by the "b" coefficient, and the slow fall is governed by the "a" coefficient.

The fast transient pulse itself can be compared to a hammer blow. The response of the aircraft structure, cabling, and systems can be compared to the ringing of a bell when struck by the hammer.

### 2.1. A Distinction between Lightning, NEMP, and ESD

The following analysis applies principally to the NEMP and to the indirect effects of lightning. Because the source impedance of an electrostatic discharge is a function of frequency and is also much higher than that of NEMP and lightning, the system response to ESD is somewhat different, and its analysis will be addressed separately.

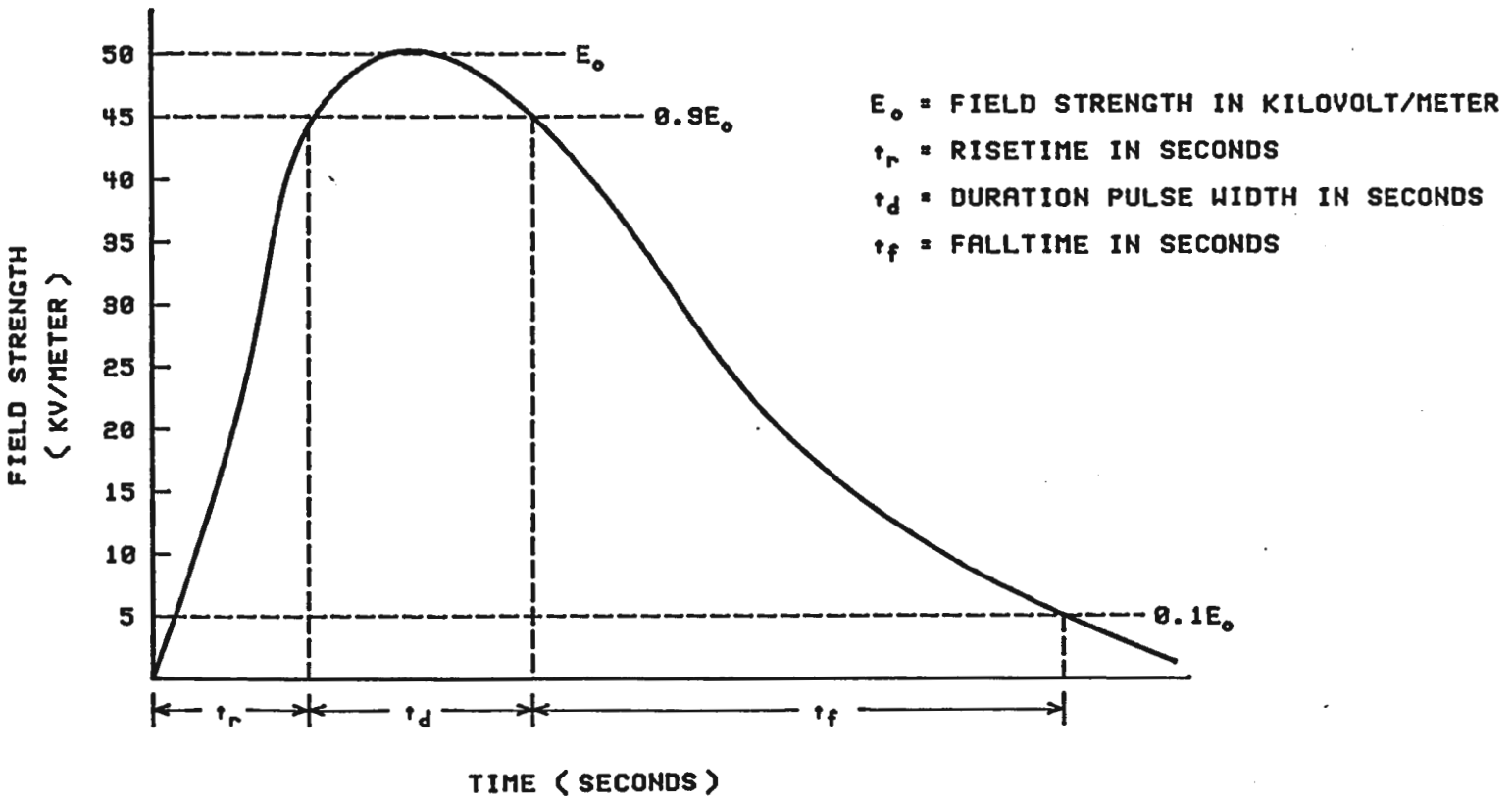



FIGURE 2-1. DOUBLE EXPONENTIAL PULSE

## 2.2. Response of the Aircraft to Lightning and NEMP

The size of the aircraft, the length of the cables within it, and the exact method of electrical termination of those cables determine the response of the aircraft to the double exponential pulse.

For example, in a large commercial transport, if a long cable in the wings or fuselage is terminated at one end with a low impedance, and at the other end in a high impedance, the cable can resonate in a quarter wavelength mode. The resulting resonant frequency might be on the order of 1 Megahertz (MHz).

When the double exponential pulse excites a resonant system in the aircraft, the response is an exponentially damped sinusoidal wave form. The frequency of this sinusoidal response is determined by the dimensions of the structure or cabling. The damping is determined by the energy storage and losses of the aircraft system, or quality factor, Q.

The resulting waveform can be represented mathematically as:

$$V = V_o \left(e^{-at} - e^{-bt}\right) \cos(2\pi f_o t) \qquad (2)$$

where $f_o$ is the resonant frequency assigned to the aircraft structure or cabling. The rise- and fall-times are not those of the excitation pulse, but are determined by the physical properties of the resonant system. "Physical properties of the resonant system" refer to the energy storage capabilities of the aircraft wiring (determined by the inductance and capacitance of the wiring) and to the resistive losses (determined by the loads connected to the cables).

The voltage $V_o$ in Equation 2 is a function of the exciting pulse amplitude, the aircraft fuselage shielding, and the cable shielding. $V_o$ is usually much smaller than the voltage amplitude of the double exponential pulse itself.

A simpler version of Equation 2 is often used to represent the aircraft system response:

$$V = 1.05\ V_o \left(e^{-at}\right) \sin(2\pi f_o t) \qquad (3)$$

In this representation the rise-time is governed by the choice of the system resonant frequency. The rise-time is on the order of the time required for the sine wave to rise from zero to its first peak, at one-quarter of one period of the resonant frequency. The "a" coefficient, which determines the rate of decay, is defined as $\pi\ f_o/Q$. The value of Q is usually between 15 and 25 for aircraft cabling systems.

This form of the equation is somewhat more easily analyzed than that of Equation 2 and will be used throughout this chapter. Although Equation 2 is a more exact

representation than is Equation 3, the differences between them in terms of maximum voltage, maximum current, frequency spectrum, and energy transfer are negligible.

It should be noted that the $V_o$ and "a" terms in Equation 1 and Equation 3 are not the same. The only connection is that the peak amplitude of the system response is proportional to the peak amplitude of the NEMP or lightning pulse.

In Equation 3 the coefficient 1.05 is a function of the Q of the system, and can vary from about 1.03 to 1.05 as the Q varies from 15 to 25 in typical resonant structures. The exact value of this coefficient is:

$$k = e^{\frac{\pi}{4Q}} \qquad (4)$$

This coefficient, multiplied by the maximum voltage induced on the cabling ($V_o$), yields an initial over-voltage such that when the falling exponential curve meets the rising sinusoidal curve, the resulting voltage is the maximum voltage of the transient response. The exact value of k is not particularly critical to the energy analysis, but the value assigned to Q is, since the energy of the system response is closely proportional to Q.

For NEMP analysis, the voltage $V_o$ is replaced with a current $I_o$. MIL-STD-461 CS10 and CS11 define $I_o$ as in figure 2.2-1. The current shown in figure 2.2-1 represents the current that could be induced on aircraft wiring when the aircraft is subjected to a NEMP or a NEMP simulator during testing. Figure 2.2-1 also shows the exponentially damped sinusoidal waveform.

Lightning and NEMP test requirements may specify either the voltage or the current of the test waveform. Through a knowledge of system impedances and with the use of Ohm's Law, voltages and currents can be rendered equivalent to one another.

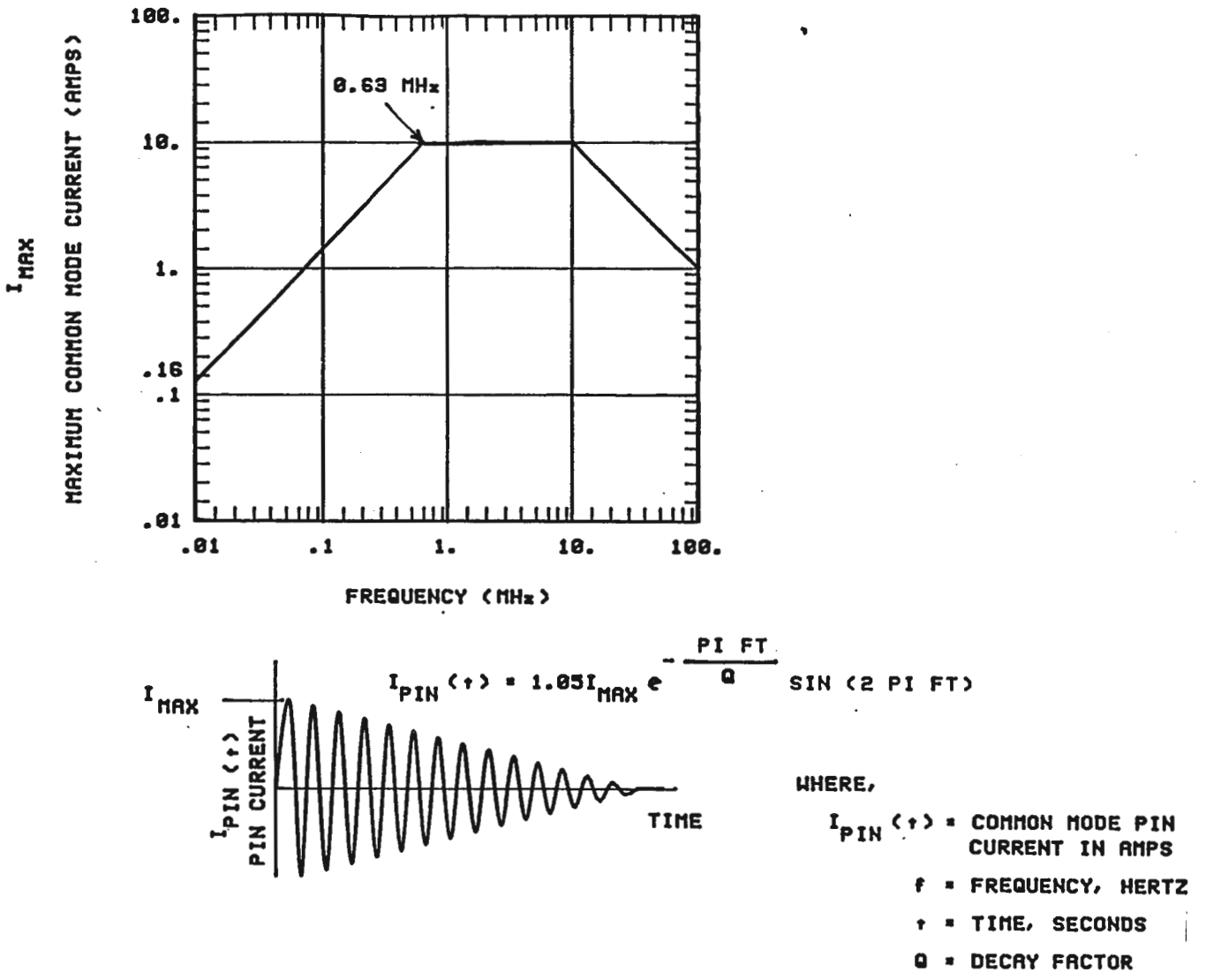



FIGURE 2.2-1. NEMP CURRENT

## 3. ENERGY TRANSFER FROM THE PULSE TO THE SYSTEM

When the double exponential pulse is applied to the system, and the system responds in its resonant fashion, energy is transferred from the pulse into the system. It is this energy which can cause upset or damage.

In a pin-level test the voltage of Equation 3 is applied directly to one of the conductors in a multiconductor cable. If the source resistance of the testing device and the load resistances connected to each end of the cable are known, the power developed in the system is then found by squaring Equation 3 and dividing the result by the sum of the source and load resistances.

The total power is:

$$P = \frac{V^2}{(R_S + R_L)} \tag{5}$$

where V is obtained from Equation 3. $R_S$ is the source resistance of the testing device and $R_L$ is the load resistance of the cable under test.

To find the energy in the system, it is necessary to integrate Equation 5 with respect to time. The total energy delivered to both source and load is:

$$E_t = \frac{k^2 V_0^2}{(R_S + R_L)} \int_0^\infty e^{-2at} \sin^2(w_0 t)dt \tag{6}$$

The total energy is divided between the source and load resistances as follows:

$$E_S = \frac{E_t R_S}{(R_S + R_L)} \tag{7}$$

$$E_L = \frac{E_t R_L}{(R_S + R_L)} \tag{8}$$

where $E_S$ and $E_L$ are the energy into the source and load resistances, respectively.

Integrating Equation 6 and allocating the energy to source and load as in Equations 7 and 8 produces the following equations:

$$E_S = \frac{k^2 V_o^2 R_S Q}{(R_S + R_L)^2 \left(1 + \frac{1}{4Q^2}\right)(4\pi f_o)} \quad (9)$$

$$E_L = \frac{k^2 V_o^2 R_L Q}{(R_S + R_L)^2 \left(1 + \frac{1}{4Q^2}\right)(4\pi f_o)} \quad (10)$$

### 3.1. An Energy Transfer Example

As an example of the energy that can be transferred from a NEMP or from the indirect effects of lightning, Equations 9 and 10 can be solved by assigning typical values to the variables:

$V_o$ = 600 volts
$Q$ = 20
$k$ = 1.04
$R_S$ = 20 ohms
$R_L$ = 20 ohms
$f_o$ = 1 MHz

The voltage (600 volts) is typical of the voltage induced on aircraft wiring by fast transients.

The Q value of 20 is representative of aircraft systems. The source and load impedances can have many values. The values chosen are low and will result in high energy transfer.

The resonant frequency of 1 MHz is suggestive of a very large aircraft. It also implies a large energy transfer.

When Equations 9 and 10 are solved using the assigned values, 7.74 millijoules of energy are delivered to the source and to the load. The burnout level for solid state devices is often taken to be 100 microjoules, where 1 joule is equal to 1 watt-second. The calculated value of energy is 77.4 times the burnout level, so obviously there is a problem. The system affected by the transient pulse must be protected in some way.

The energy distribution of the system response in the frequency domain will now be examined. This approach will lead immediately to a low-pass filter specification which will limit the amount of transferred energy to below the burnout level.

## 4. FREQUENCY SPECTRUM OF THE SYSTEM RESPONSE

The transient response of the system as a function of time has been depicted, as shown in the lower part of figure 2.2-1. There is also a frequency spectrum representation from which the energy distribution as a function of frequency may be found.

Mathematically, the Fourier transform is used to derive the voltage as a function of frequency from its time domain representation. An explanation of the Fourier transform is beyond the scope of this chapter. However, the frequency spectrum derived through the Fourier transform is shown in figure 4-1. The spectrum is that of a system having a resonance at 1 MHz and a Q of 20. The amplitude scale at the left is in decibels (dB), with the zero dB level corresponding to the voltage in the system at zero frequency. The voltage at resonance is Q times the voltage at zero frequency. From this consideration it may be observed that the voltage peak at resonance is 20 Log Q dB above the zero frequency reference level. The bandwidth between the points 3 dB below the peak response also yields the value of Q through the relation:

$$Q = \frac{f_o}{\text{BandWidth(3dB)}} \tag{11}$$

At about two times the resonant frequency the slope of the voltage characteristic begins to approach a 40 dB per decade slope. As a consequence, the energy contribution at frequencies more than about 4 times higher than the resonance is negligible.

If the voltage versus frequency characteristic of figure 4-1 is squared, divided by the sum of the source and load resistances, and integrated with respect to frequency, the energy as a function of frequency can be found. If the integration limits are zero and infinity, the total energy over the entire frequency spectrum is found and is the same as that previously determined in the time domain by Equations 9 and 10. If the integration limits are zero and 500 kilohertz (kHz), for example, then the energy between those limits can be determined.

A program in Electromagnetic Computer aided design (EMCad$^{tm}$) usable on IBM compatible PCs has been written to perform this integration and has generated some useful guidelines for the energy distribution:

- One-half of the energy lies between the half-power (3 dB) points of the resonance.
- Approximately one-fourth of the energy lies below the lower half-power (3 dB) point of the resonance.

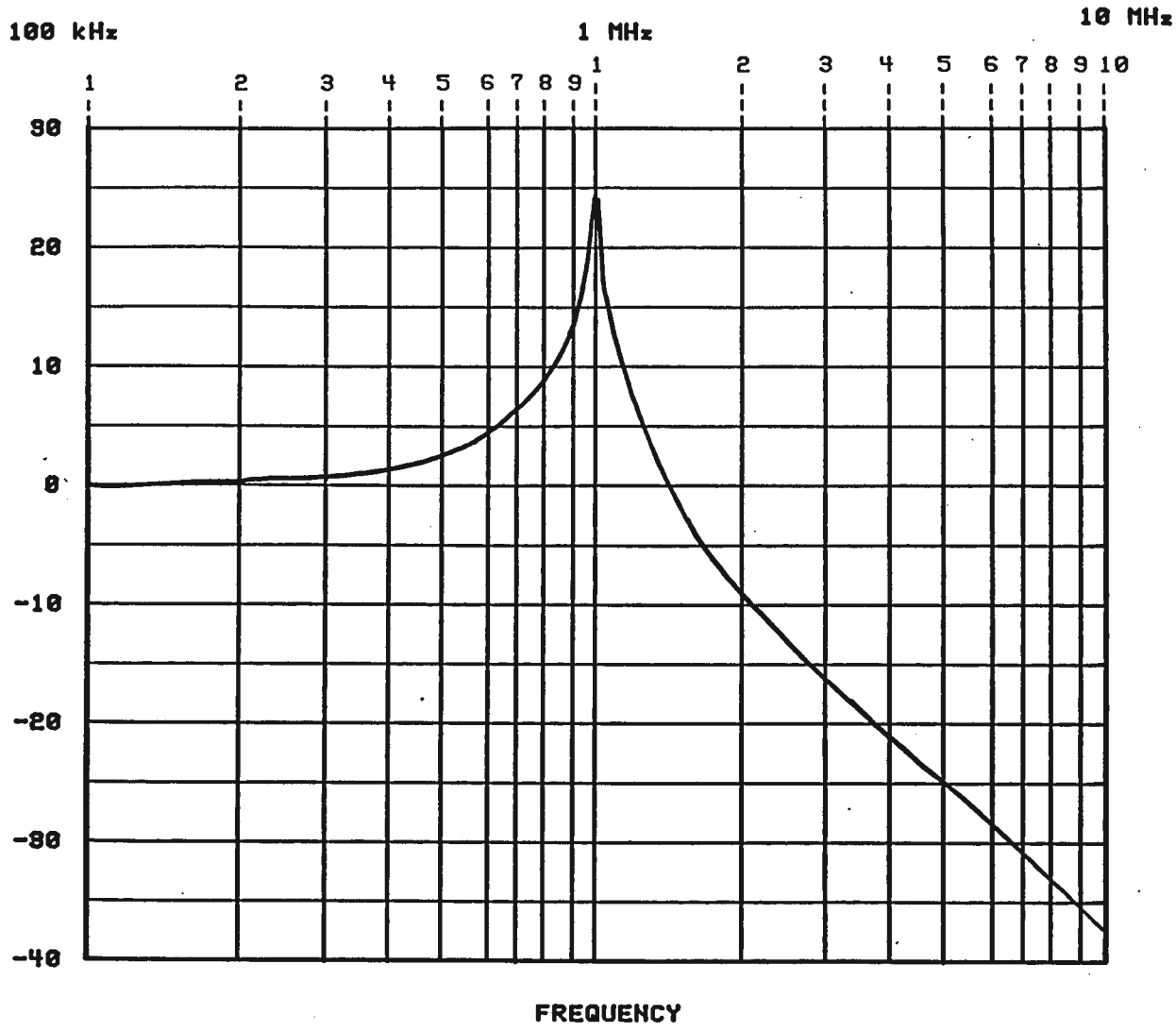



FIGURE 4-1. FREQUENCY SPECTRUM OF A 1 MHz DAMPED SINUSOID

- Approximately one-fourth of the energy lies above the upper half-power (3 dB) point of the resonance.

- Approximately 2 percent of the energy lies below 10 half-power bandwidths below the resonant frequency.

Using these guidelines, or (for more precision) the EMCad[tm] program, the frequency at which the burnout level is reached can be determined. From the examples in Equations 9 and 10, the energy delivered to the load is 7.74 millijoules. The half-power bandwidth of the 1 MHz resonance is 50 kHz (from Equation 11). The 10 half-power bandwidths below resonance then yield a frequency of 500 kHz, at which 2 percent of 7.74 millijoules will have accumulated. That is, over the frequency range from Direct Current (DC) to 500 kHz, the transient pulse will deliver 155 microjoules into the load impedance. For a more precise answer, the computer program shows that 100 microjoules will be delivered between DC and 368 kHz.

An approximate, though accurate, formula has been developed for the energy as a function of frequency. By setting the frequency equal to zero in the Fourier transform, the voltage at zero frequency can be found:

$$V(0) = \frac{V_0 \, k}{2\pi f_0 \left( 1 + \frac{1}{4Q^2} \right)} \tag{12}$$

Assuming that the system losses (represented by the "a" in $e^{-at}$ of Equation 6) are zero, then Q is infinite. For infinite Q, the voltage as a function of frequency is given by:

$$V = \frac{V(0) \, f_0^2}{\left( f_0^2 - f^2 \right)} \tag{13}$$

This expression is extremely accurate for Q's as low as 15 up to frequencies within a few half-power bandwidths of the resonant frequency.

The total energy as a function of frequency is given by:

$$E_t = \frac{2 \, V(0)^2 \, f_0^4}{(R_S + R_L)} \int_0^{f_1} \frac{df}{\left( f_0^2 - f^2 \right)^2} \tag{14}$$

The factor of 2 is required because the energy integral should be evaluated between limits of $-f_1$ and $+f_1$ rather than 0 and $f_1$. The integral of the time

function, Equation 6, is still integrated starting from time (t) = 0 and is always zero for time before t = 0.

After integrating the expression above, the energy as a function of frequency is given by:

$$E(f) = \frac{V(0)^2 f_0^2}{(R_S + R_L)} \left[ \frac{f_1}{(f_0^2 - f_1^2)} + \frac{1}{2f_0} \text{Log}_e \frac{f_0 + f_1}{f_0 - f_1} \right] \quad (15)$$

The energy calculated here divides resistively between source and load as shown in Equations 7 and 8.

If Equation 15 is evaluated at $f_1$ and $f_2$ instead of 0 and $f_1$, the energy in any specified portion of the spectrum can be found.

Criteria for a filter design are now established. To protect the circuit with the values given earlier from energy levels above 100 microjoules, the circuit must have low-pass filtering which cuts off all energy above 368 kHz.

A NEMP or lightning specification sometimes identifies several frequencies of resonance in an aircraft, such as 1, 5, 10, and 45 MHz. Each frequency corresponds to some structural or cable dimension which resonates at a wavelength usually related to that dimension in a quarter or half-wave mode. An aircraft typically displays a multitude of resonances in the 1 to 50 MHz range.

Computer analysis has shown that if the aircraft resonances are separated in frequency by just a few half-power bandwidths, each resonance and its energy contribution may be treated as independent. A separate analysis may be made for each resonant frequency, and the energy levels from each may be added to find the total energy transfer. In a multiple-resonant system the energy contribution of each resonance scales inversely with frequency, so in many cases only the two lowest resonances make a significant energy contribution.

Multiple resonances can readily occur if a cable is resonant at 1 MHz in a quarter wave-length mode. The cable will then also be resonant at odd multiples of 1 MHz, such as 3, 5, and 7 MHz. In the numerical example worked above, if a 3 MHz resonance is assumed in addition to the 1 MHz resonance, the total energy will increase from 7.74 millijoules to 10.33 millijoules. Resonances at even higher frequencies will make only a small additional energy contribution. The voltage spectrum of a system with resonances at 1 and 3 MHz is shown in figure 4-2.

A low-pass filter, which takes care of limiting the energy transfer at the lowest frequency resonances, will also eliminate the energy transfer at the higher frequency resonances, if the filter retains its attenuation characteristics at the higher frequencies. For this reason the choice of filters or filter components, which have good high frequency characteristics, is important.

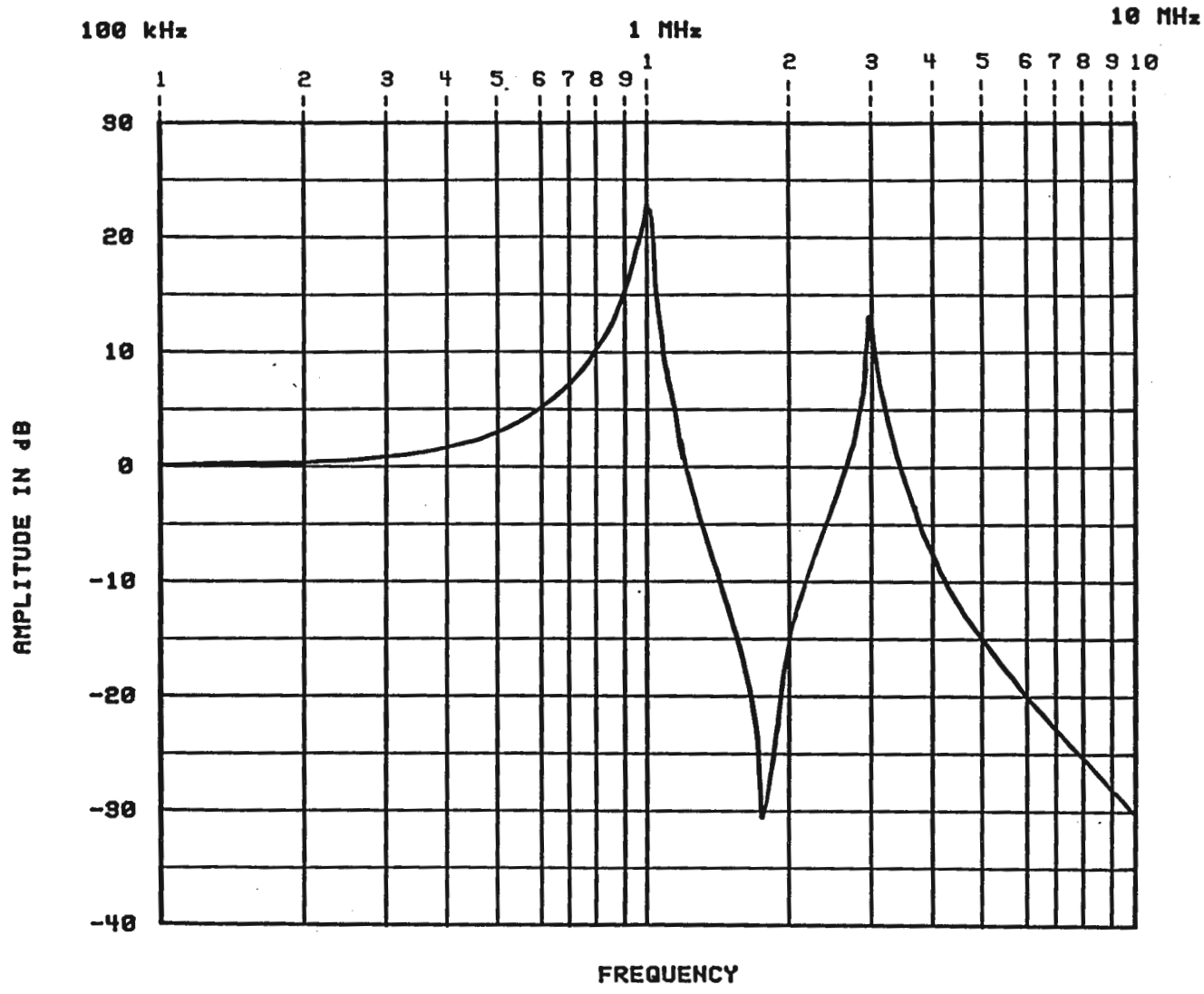



FIGURE 4-2. FREQUENCY SPECTRUM OF A DAMPED SINUSOID WITH TWO RESONANCES

If a lightning or NEMP requirement specifies only one resonant frequency, good practice dictates that a safety factor of 2 should be applied to the cut-off frequency determined by the energy analysis to eliminate the multiple resonance condition.

Since hundreds or thousands of individual circuits may have to be analyzed, a computerized approach to an evaluation of the system is essential. Spread sheets identifying conductors, load resistances, and energy levels are useful in this type of analysis. Computerized filter design programs are also helpful. The usual filter is a low-pass type, and usually only a series inductance and a shunt capacitance are required. Sometimes the necessary characteristics can be provided by a filter pin connector, in which the filter elements are incorporated into the body of the connector.

# 5. ELECTROSTATIC DISCHARGE

ESD is the familiar phenomenon experienced in stroking the back of a cat or touching a metal door knob after crossing a carpeted room. Some typical methods of generating electrostatic voltages are shown in table 5-1.

The accumulation of charge on the human body, or upon objects, can generate voltages up to 30 kilovolts. When the accumulated charge is eventually redistributed via a discharge arc, the fast rise-time pulse that results can have adverse effects on susceptible electronics equipment, ranging from upset to damage.

Aircraft are well suited for the generation of ESD. Relative humidity is often low, so that accumulated charge cannot slowly and harmlessly leak off without an arc. The decrease of atmospheric pressure with altitude somewhat decreases the insulating properties of air, and the motion of the aircraft through the air can cause an accumulation of charge. Finally, the movements of passengers and crew can cause charge accumulation in the same way as on the ground. Discharges from passengers or crew to metallic objects in the cabin can generate a field which may couple to cables routed behind decorative panels.

## 5.1. Energy Transfer in ESD

Although the waveform of the transient pulse for ESD is similar in shape to that of lightning and NEMP, the frequency spectrum of the energy transferred to the victim circuit from an ESD pulse is rather different. The source impedance of lightning and NEMP test devices, and the source impedance of the physical phenomena are usually quite low and mainly resistive. The ESD source, as shown in table 5-1, is characterized as a resistance in series with a capacitance. The impedance of this source is quite high at low frequencies and decreases with increasing frequency as the impedance of the capacitor decreases. As a result, the spectral content of an ESD has less energy at low frequencies and more at high frequencies than lightning and NEMP. Also, much higher voltages are required in ESD to generate the same total energy as for lightning and NEMP. However, the voltages available from ESD generating mechanisms are in fact very high, and the total transferred energy can be comparable to that of lightning and NEMP.

The energy of an ESD pulse may be delivered directly to a sensitive conductor or component, but more often the discharge is from the human body to a grounded object. Most of the energy is dissipated harmlessly to ground, but the flow of current to ground generates an electric field. In the analysis of ESD effects, it is useful to consider the effect of this electric field on surrounding objects in addition to the energy transfer considerations.

TABLE 5-1. TYPICAL ELECTROSTATIC VOLTAGES

| Means of Static Generation | Electrostatic Voltages | |
|---|---|---|
| | 10 to 20 Percent Relative Humidity | 65 to 90 Percent Relative Humidity |
| Walking across carpet | 35,000 | 1,500 |
| Walking over vinyl floor | 12,000 | 250 |
| Worker at bench | 6,000 | 100 |
| Vinyl envelopes for work instructions | 7,000 | 600 |
| Common poly bag picked up from bench | 20,000 | 1,200 |
| Work chair padded with polyurethane foam | 18,000 | 1,500 |

## 5.2. Description of the ESD Pulse

The typical ESD pulse is a double exponential and has been described earlier in this chapter. For easier analysis the pulse can be approximated by a triangular wave form as shown in figure 5.2-1. The frequency domain characterization of the pulse is also shown in figure 5.2-1. Significant amounts of energy appear well into the VHF region. The ESD pulse is inherently broadband, and susceptible equipment must be protected against it from very low frequencies to several hundred MHz.

This broadband pulse is capable of exciting resonances in nearby structures. The resulting response is a damped sinusoid, as shown earlier in figure 2.2-1.

## 5.3. Direct Discharge ESD

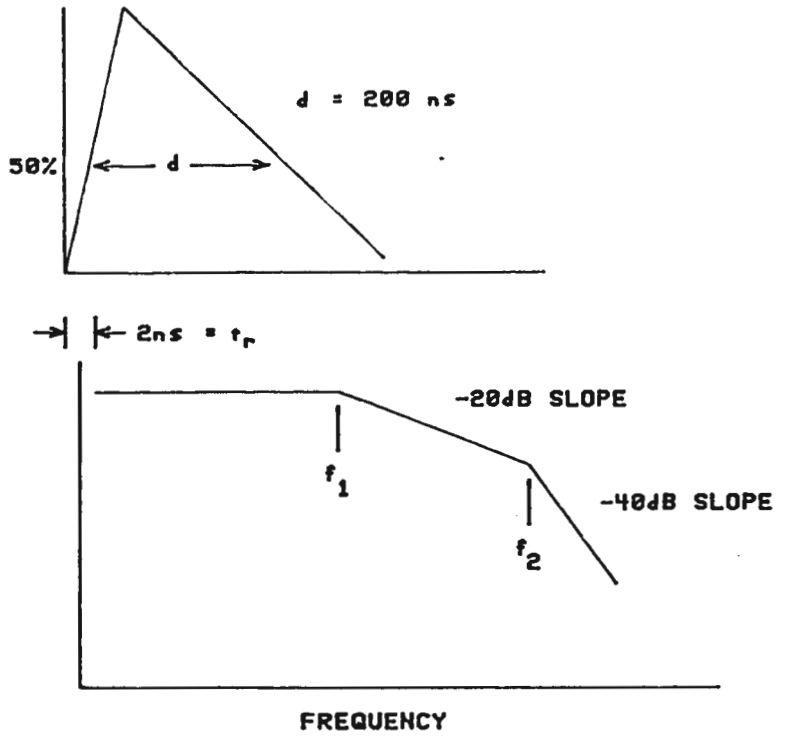
An arc from the hand to a susceptible device is called direct discharge. The human body, for ESD purposes, can be modeled as a simple series RC circuit (shown in figure 5.3-1).

For the simulation of an ESD, an ESD simulator is used (shown in figure 5.3-2). The energy stored in the human body or in the capacitor of the ESD simulator is given by $1/2\ CV^2$, where C is the capacitance and V is the voltage. This energy is in joules or watt-seconds. For a 100 pf capacitor charged to 12 kV, the total energy delivered to the series resistor and the device under test is 7.2 millijoules. The total energy divides between the series resistance and the resistance of the test device. The energy delivered to the test device can be compared to its energy specification, if given, to see if there is danger of damage. The energy division occurs as shown in Equations 7 and 8.

The time required for the voltage to decay to 1/e (37 percent) of its initial value is given by $t = RC$. For a series resistance of 1500 ohms and a 100 pf capacitor, t is 0.15 microseconds.

In many cases upset rather than damage is the result of this kind of discharge. This condition is best checked during actual operation of the system under test. Repeated discharges will sometimes cause upset when a single discharge does not. A single discharge may occur in the time between data bit transmissions, whereas a repeated discharge has a good chance of occurring at the time of a data bit transmission and can cause a loss or alteration of data.

Many solid state devices are very susceptible to permanent damage from ESD incidental to handling. Careful training of installation and maintenance personnel, and adherence to handling procedures is required to avoid damage. Complimentary Metal-Oxide Semiconducter (CMOS) integrated circuits are known to be particularly subject to damage by mishandling, both as isolated components and when installed on circuit boards. For the proper handling procedures for ESD sensitive components, much useful information can be obtained from ESD Control in the Manufacturing Environment, published by the Department of Defense (DOD) Reliability Analysis Center. Military Handbook DOD-HDBK-263, Electrostatic Discharge Control Handbook for Protection of Electrical and Electronic Parts, Assemblies and Equipment is also of use.




$$f_2 = \frac{1}{PI\ t_r}$$

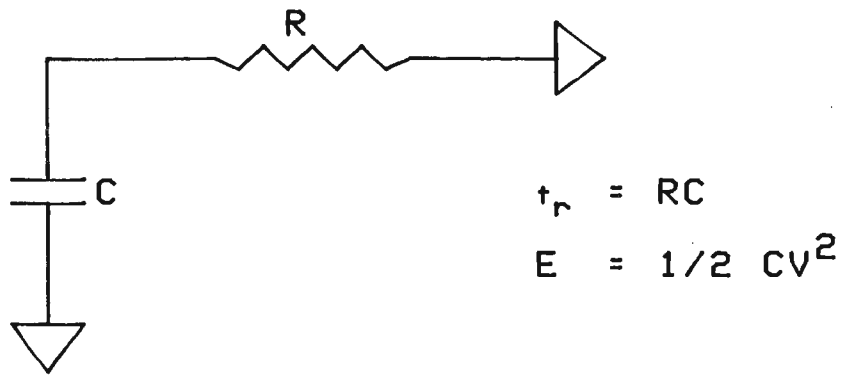
$$f_1 = \frac{1}{PI\ (d + t_r)}$$

$$C_n = \frac{2\ A\ (d + t_r)}{T}$$

LET $f_o$ = 1 SEC. T = 1
A = 12KV

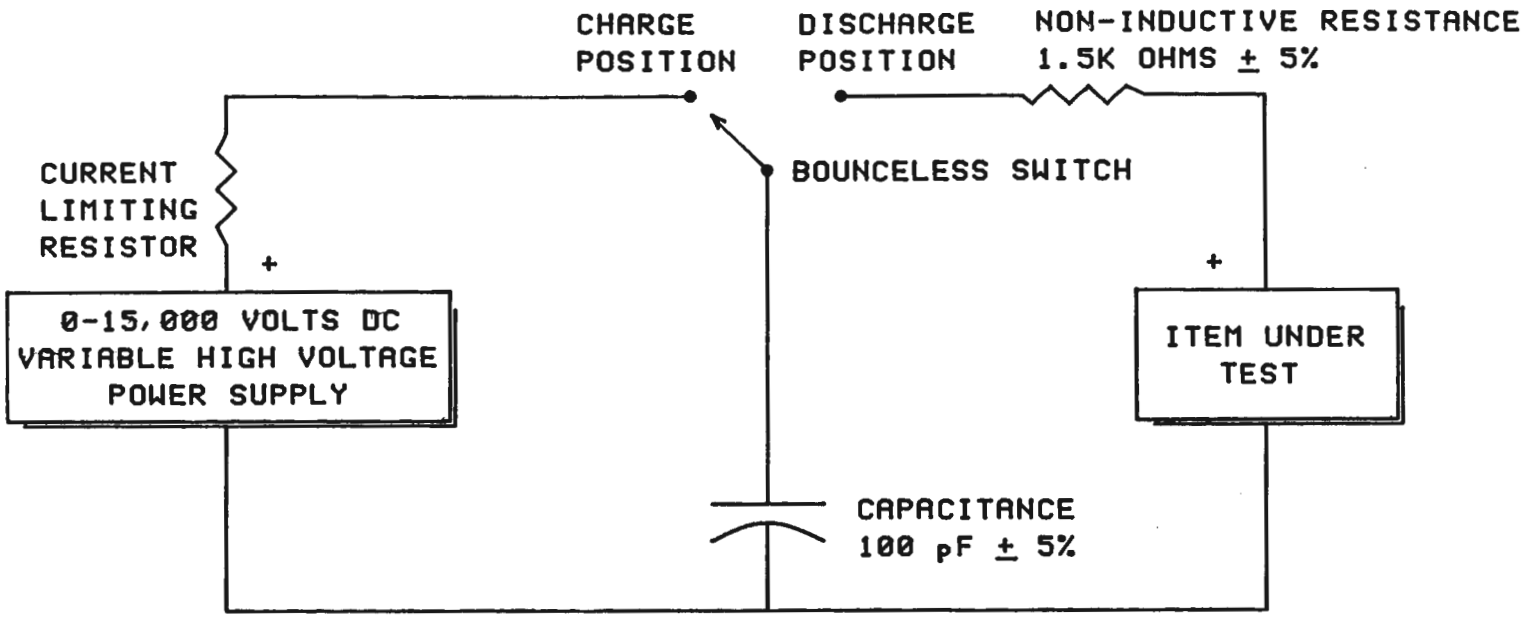
| FREQUENCY | Cn mV | Cn dBuV | Cn dBuV/MHz |
|---|---|---|---|
| 1 Hz | 4.8 | 73.7 | 193.7 |
| $f_1$ = 1.58MHz | 4.8 | 73.7 | 193.7 |
| $f_2$ = 159MHz | 4.8 | 33.7 | 153.7 |

FIGURE 5.2-1. ESD VOLTAGE PREDICTION




| PARAMETER | LOW | HIGH | TYPICAL |
|---|---|---|---|
| R | 5 | 100 K OHMS | 1-4 K OHMS |
| C | 25pf | .003 uf | 100-150pf |
| V | 1 kv | 30 kv | 8-12 kv |
| $t_r$ | 125 P SEC | 300 U SEC | 1-6 N SEC |
| E | 100 uJ | 1.3 J | 3-10 mJ |

FIGURE 5.3-1. HUMAN ESD MODELING




NOTE: TEST VOLTAGES ARE MEASURED ACROSS THE CAPACITANCE. THE CAPACITOR SHALL BE DISCHARGED THROUGH THE SERIES RESISTOR INTO THE ITEM UNDER TEST BY MAINTAINING THE BOUNCELESS SWITCH TO THE DISCHARGE POSITION FOR A TIME NO SHORTER THAN REQUIRED TO DECAY THE CAPACITOR VOLTAGE TO LESS THAN 1 PERCENT OF THE TEST VOLTAGE OR 5 SECONDS, WHICHEVER IS LESS. POWER SUPPLY VOLTAGE SHALL BE WITHIN A TOLERANCE OF ± 5 PERCENT OF TEST VOLTAGE.

FIGURE 5.3-2. ESD TEST CIRCUIT

Table 5.3-1 lists a wide variety of electronic devices, grouped by their sensitivity to the voltage of an ESD. Table 5.3-2 shows the failure mechanisms that occur in various devices from ESD exposure.

Testing for direct discharge effects is performed as shown in figure 5.3-3. In the laboratory, the Alternating Current (AC) power would be provided as shown. On board an aircraft, it may be convenient to power the ESD simulator from batteries. Systems are best checked by operating them and watching for upset or damage when the ESD pulse is applied.

The effects of ESD on a digital system can range from missing data, to a condition requiring reset of the system, to damage requiring the replacement of parts.

## 5.4. Prediction of ESD Radiated Electric Fields

In the following example, a prediction is made of the effects of an ESD from the human hand to a metallic object in the cockpit of a commercial transport. A cabin attendant comes forward, touches a metallic object located one meter away from a susceptible item of equipment, and causes a discharge to occur. This analysis starts with the triangular pulse approximation of figure 5.2-1, and uses the chart in table 5.4-1 to guide the calculations.

In table 5.4-1, the amplitude of the pulse is 12 kV, which is quite typical of the ESD associated with the human body. The Fourier coefficient, which gives the amplitude of the pulse in the frequency domain, is $C_n$ measured in dB microvolts per MHz. Note in the legend at the bottom of figure 5.2-1 that the amplitude as a function of frequency is constant from essentially DC to 1.58 MHz and then decreases at a rate of 20 dB per decade to 159 MHz. Above 159 MHz the amplitude decreases at a rate of 40 dB per decade. The electric field strength that is generated at a distance of one meter from this 12 kV discharge can now be calculated.

Referring to the frequency column of table 5.4-1, 1.58 MHz was entered as the frequency at which the radiated field of a 12 kV discharge was first calculated. Enter in column 1, 193.7 dB$\mu$V/MHz, the Fourier coefficient obtained in figure 5.2-1. In column 2 enter -34 dB, which is a standard transfer function value to convert from a voltage on a conductor to the electric field generated one meter away. In column 3 enter a figure in dB (for the antenna factor) to determine how efficient the source is as a radiator of energy. This calculation compares the level of radiation from a short antenna to that of a quarter wave-length antenna at the same frequency. In this case, the entire human body is the major part of the radiating circuit and can be determined to be 2 meters. Using Note 3 in table 5.4-1, the antenna factor is calculated as 13.8 dB. Finally, in columns 4, 5, and 6 enter dB values for the number of conductors carrying the signal for the distance from the discharge to the susceptible nearby equipment and for the height above ground of the radiating object. Values of zero dB have been entered in each of these columns, indicating one conductor, a distance of 1 meter, and a height above ground of 0.5 meters.

TABLE 5.3-1. ESD PARTS BY PART TYPE

| CLASS 1: SENSITIVITY RANGE 0 TO $\leq$ 1000VOLTS |
|---|
| • Metal Oxide Semiconductor (MOS) devices including C, D, N, P, V, and other MOS technology without protective circuitry, or protective circuitry having Class 1 sensitivity |
| • Surface Acoustic Wave (SAW) devices |
| • Operational Amplifiers (OP AMP) with unprotected MOS capacitors |
| • Junction Field Effect Transistors (JFETs) (Ref.: Similarity to MIL-STD-701: Junction field effect transistors and junction field effect transistors, dual unitized) |
| • Silicon Controlled Rectifiers (SCRs) with Io $<$0.175 amperes at 100$^{0}$ Celsius ($^{0}$ C) ambient temperature (Ref.: Similarity to MIL-STD-701: Thyristors (silicon controlled rectifiers)) |
| • Precision Voltage Regulator Microcircuits: Line or Load Voltage Regulation $<$0.5 percent |
| • Microwave and Ultra-High Frequency Semiconductors and Microcircuits: Frequency $>$1 gigahertz |
| • Thin Film Resistors (Type RN) with tolerance of $\leq$0.1 percent; power $>$ 0.05 watt |
| • Thin Film Resistors (Type RN) with tolerance of $>$ 0.1 percent; power $\leq$0.05 watt |
| • Large Scale Integrated (LSI) Microcircuits including microprocessors and memories without protective circuitry, or protective circuitry having Class 1 sensitivity (Note: LSI devices usually have two or three layers of circuitry with metallization crossovers and small geometry active elements) |
| • Hybrids Utilizing Class 1 parts |

TABLE 5.3-1. ESD PARTS BY PART TYPE (Continued)

| CLASS 2: SENSITIVITY RANGE > 1000 TO ≤ 4000 VOLTS |
|---|
| • MOS devices or devices containing MOS constituents including C, D, N, P, V, or other MOS technology with protective circuitry having Class 2 sensitivity |
| • Schottky diodes (Ref.: Similarity to MIL-STD-701: Silicon switching diodes (listed in order of increasing tr)) |
| • Precision Resistor Networks (Type RZ) |
| • High Speed Emitter Coupled Logic (ECL) Microcircuits with propagation delay ≤1 nanosecond |
| • Transistor-Transistor Logic (TTL) Microcircuits (Schottky, low power, high speed, and standard) |
| • Operational Amplifiers (OP AMP) with MOS capacitors with protective circuitry having Class 2 sensitivity |
| • LSI with input protection having Class 2 sensitivity |
| • Hybrids utilizing Class 2 parts |

TABLE 5.3-1. ESD PARTS BY PART TYPE (Continued)

| CLASS 3: SENSITIVITY RANGE >4000 TO ≤15,000 VOLTS |
|---|
| • Lower Power Chopper Resistors (Ref.: Similarity to MIL-STD-701: Silicon Low Power Chopper Transistors) |
| • Resistor Chips |
| • Small Signal Diodes with power ≤1watt excluding Zeners (Ref.: Similarity to MIL-STD-701: Silicon Switching Diodes (listed in order of increasing tr)) |
| • General Purpose Silicon Rectifier Diodes and Fast Recovery Diodes (Ref.: Similarity to MIL-STD-701: Silicon Axial Lead Power Rectifiers, Silicon Power Diodes (listed in order of maximum DC output current), Fast Recovery Diodes (listed in order of tr)) |
| • Low Power Silicon Transistors with power ≤5watts at 25° C (Ref.: Similarity to MIL-STD-701: Silicon Switching Diodes (listed in order of increasing tr), Thyristors (bi-directional triodes), Silicon PNP Low-Power Transistors (Pc ≤5 watts $T_A$ = 25° ), Silicon RF Transistors) |
| • All other Microcircuits not included in Class 1 or Class 2 |
| • Piezoelectric Crystals |
| • Hybrids utilizing Class 3 parts |

TABLE 5.3-2. PART CONSTITUENTS SUSCEPTIBLE TO ESD

| Part Constituent | Part Type | Failure Mechanism | Failure Indicator |
|---|---|---|---|
| Film Resistors | Hybrid ICs:<br>Thick Film Resistors<br>Thin Film Resistors<br><br>Monolithic IC-Thin Film Resistors<br><br>Encapsulated Film Resistors | Dielectric breakdown voltage dependent-creation of new current paths<br><br>Joule heating-energy dependent-destruction of minute current paths | Resistance shift |
| Metalization Strips | Hybrid ICs<br><br>Monolithic ICs<br><br>Multiple Finger Overlay Transistors | Joule heating-energy dependent metalization burnout | Open |
| Field Effect Structures and Nonconductive Lids | LSI and Memory ICs employing nonconductive quartz or ceramic package lids especially ultraviolet EPROMS | Surface inversion or gate threshold voltage shifts from ions deposited on surface from ESD | Operational degradation |
| Piezoelectric Crystals | Crystal Oscillators | Crystal fracture from mechanical forces when excessive voltage is applied. | Operational degradation |
| Closely Spaced Electrodes | Surface Acoustic Wave Devices<br><br>Thin metal unpassivated, unprotected semiconductors and microcircuits | Arc Discharge melting and fusing of electrode metal | Operational degradation |

TABLE 5.3-2. PART CONSTITUENTS SUSCEPTIBLE TO ESD (Continued)

| Part Constituent | Part Type | Failure Mechanism | Failure Indicator |
|---|---|---|---|
| MOS Structures | MOS FET (Discretes)<br><br>MOS ICs<br><br>Semiconductors with metalization crossovers<br>Digital ICs (Bipolar and MOS)<br>Linear ICs (Bipolar and MOS)<br><br>MOS Capacitors<br>Hybrids | Dielectric breakdown from excess voltage and subsequent high current | Short (high leakage) |
| Semiconductor Junctions | Diodes (PN, PIN,<br><br>Schottky)<br><br>Transistors, Bipolar Junction Field Effect Transistors<br><br>Thyristors<br><br>Bipolar ICs, Digital and Linear<br><br>Input Protection Circuits on:<br>Discrete MOS FETs<br>MOS ICs | Microdiffusion from micro-plasma secondary breakdown from excess energy or heat<br><br>Current filament growth by silicon and aluminum diffusion (electromigration) | Operational degradation |

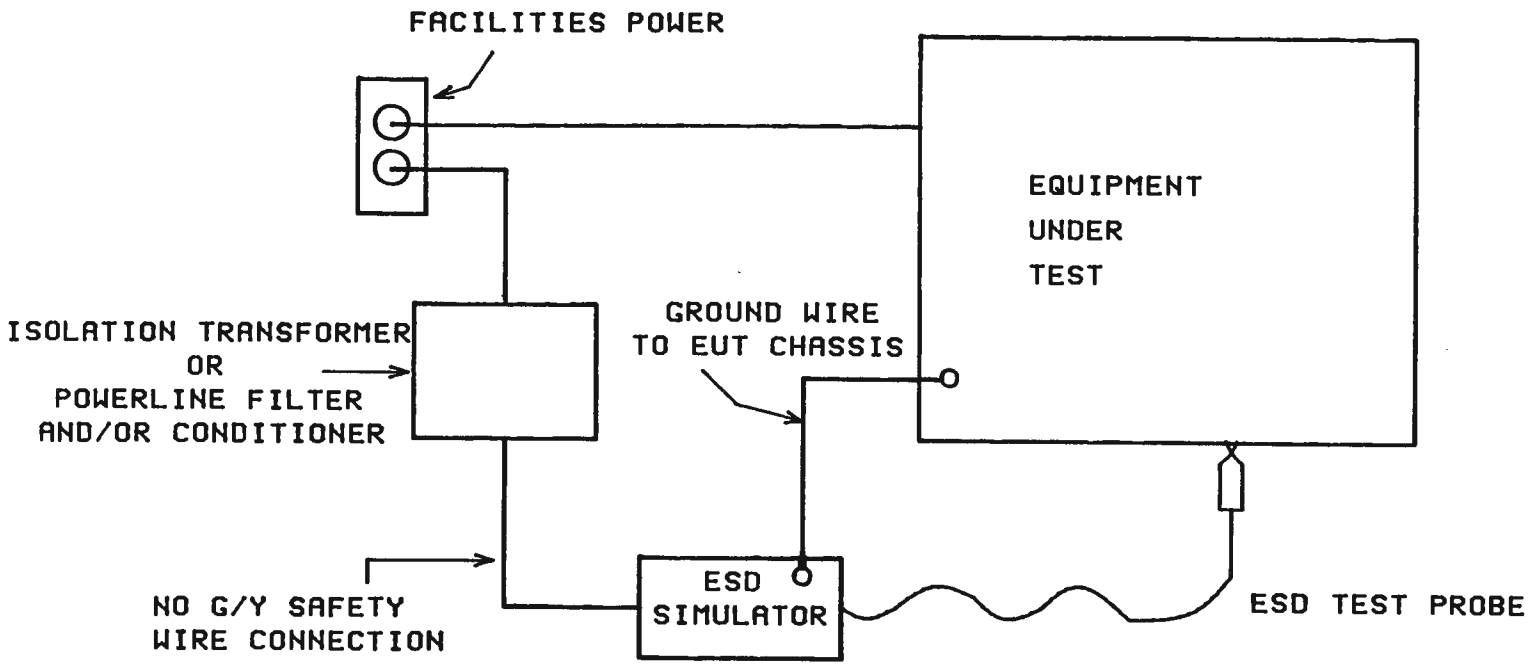



FIGURE 5.3-3. ESD DIRECT DISCHARGE EQUIPMENT CONFIGURATION

TABLE 5.4-1. PREDICTION OF RADIATED EMISSIONS

| FREQ | 1. Cn dBuV/MHz | 2. $T_F$ | 3. ANT. F. | 4. N | 5. D | 6. HEIGHT | 7. RESULT |
|---|---|---|---|---|---|---|---|
| 1.58 MHz | 193.7 | -34 | -13.8 | 0 | 0 | 0 | 145.9 |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |

The dB values in columns 1 through 6 are now added to obtain the result in column 7, 145.9 dBμV/meter/MHz. This value can be converted to a radiated field strength:

$$E = 10^{(145.9/20)}\ \mu V/meter/MHz$$

or

$$E = 19.7\ V/meter/MHz$$

To arrive at the magnitude of the field threatening a broadband device, the frequency domain plot of figure 5.2-1 can be integrated over the frequency range of interest. Piecewise integration from DC to 1.58 MHz yields a value of 31.2 volts per meter. This value is the field value that would be seen by a device which cuts off sharply at 1.58 MHz. Continuing the integration from 1.58 MHz to 159 MHz gives an additional field value of 1,550 volts per meter as the contribution from this portion of the spectrum. The total field is the sum of the two piecewise integrations and is then 31.2 + 1,550 = 1,581 volts per meter per 159 MHz. This is the field which would be seen by a broadband device having response from very low frequencies up to 159 MHz. Above 159 MHz, the field strength decreases at a rate of 40 dB per decade, and the contribution from this part of the spectrum is negligible.

5.5. Prediction of Voltage Induced on a Susceptible Circuit

The next part of the problem is to calculate how much voltage might be induced on a susceptible circuit by the field calculated above. For the susceptible circuit a printed circuit board will be used into which a loop area has been inadvertently designed. The loop acts as an antenna which responds to the field calculated above. Assumed are loop dimensions of 4 inches by 6 inches, which could be created by a trace around the periphery of the board. The induced voltage as a function of field strength is given by:

$$V(volts) = \frac{2\pi}{\lambda} E\ N\ A \cos\theta \qquad (16)$$

E is the field strength; N is the number of turns in the receiving loop; A is the area of the loop in square meters; lambda is the wavelength; and theta is the angle of the loop with respect to the radiating field. (Theta is usually taken to be zero for a worst case estimate.)

An effective way of combining the broadband field generated by the pulse with the response of the loop from DC to 159 MHz is to plot the field and the response in dB on semilog paper, as in figure 5.5-1. No values are shown above 159 MHz because the induced voltage above that point is negligible. The generated E field is shown in the uppermost curve and has frequency distribution as in figure 5.2-1. The loop response with an applied field of 1 volt per meter is shown in the lower curve and shows a linear response with frequency with a

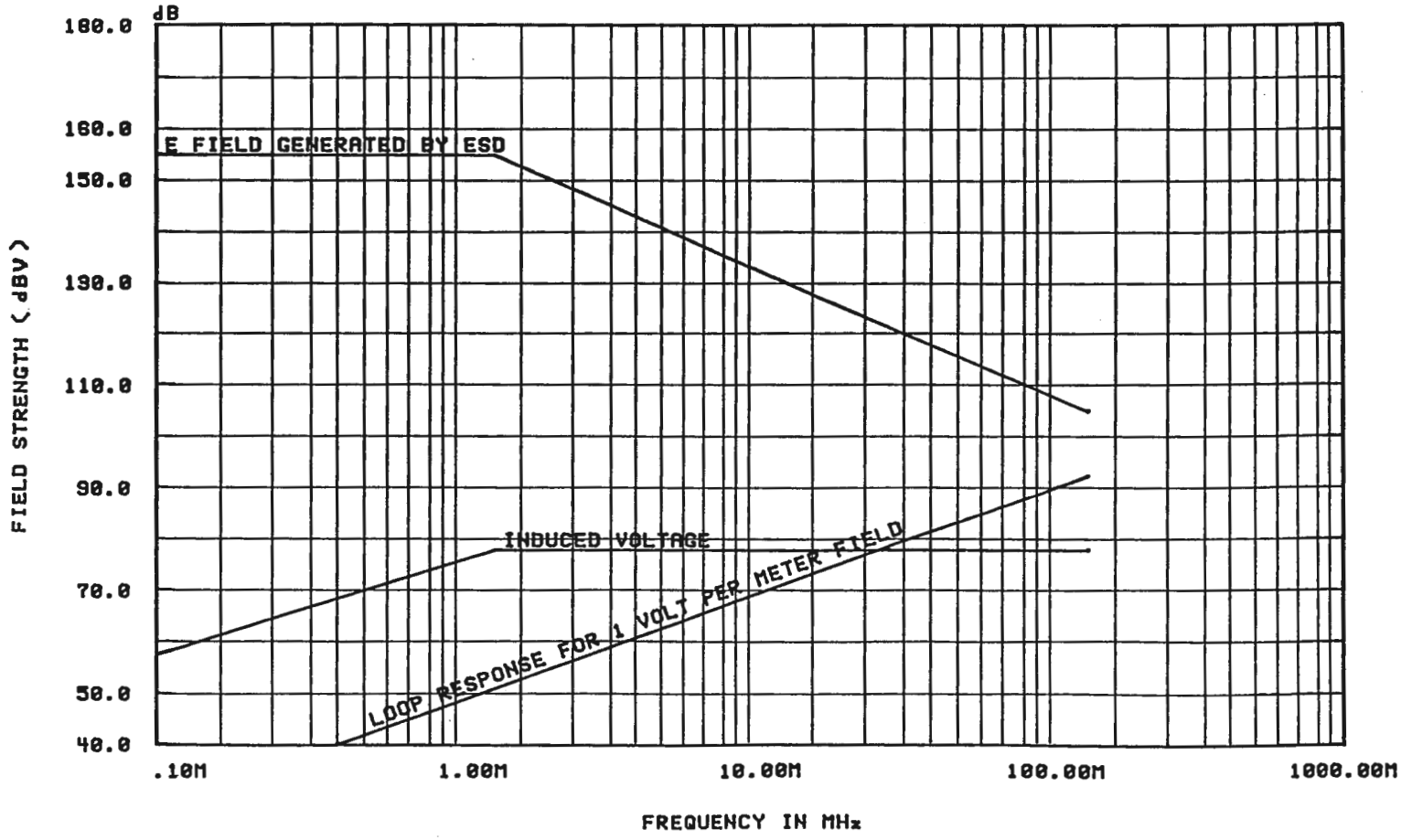



FIGURE 5.5-1. INDUCED VOLTAGE ON A LOOP FROM ESD

slope of 20 db per decade. The curve for the induced voltage in the loop is found by combining the applied field curve and the response curve. From 1.58 MHz up, the curve shows a steady value of 7.9 millivolts per MHz.

When the curve in figure 5.5-1 is integrated over the bandwidth to 159 MHz, the total voltage to which a broadband device would be exposed is about 1.25 volts, which is in the upset range for TTL devices.

In the analysis it was assumed that the receiving loop was unshielded. Shielding of the susceptible circuit will normally provide enough attenuation to reduce the induced voltage to a non-threatening level. However, an increase in the original 12 kV discharge voltage, a failure of shielding, a resonance condition in the susceptible circuit, or a decrease in the distance from the discharge to the susceptible equipment, can increase the induced voltage to the upset or damage levels.

Testing for radiated induced ESD is performed on installed systems using an ESD simulator while operating the systems under test and looking for upset conditions or damage conditions.

## 5.6. ESD Induced Voltages on Cables

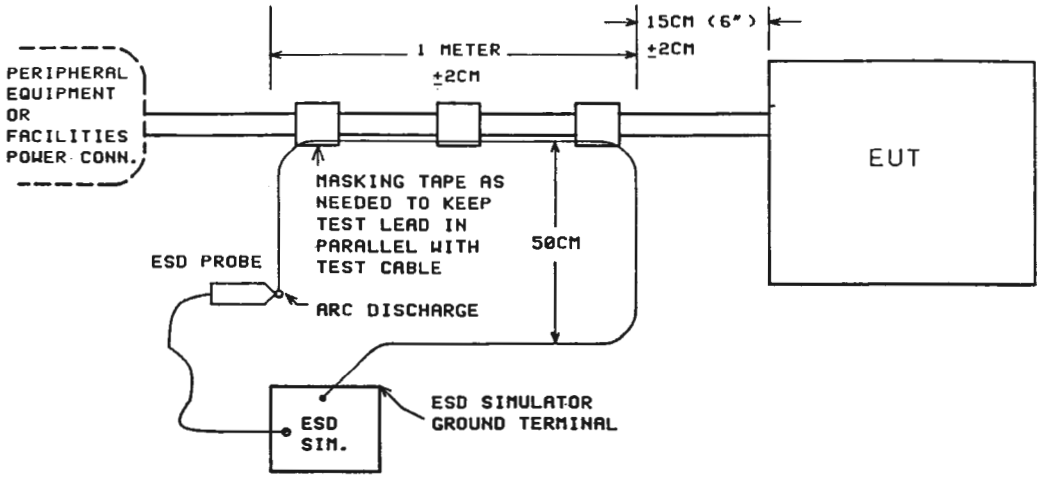
When a discharge occurs to a cabinet, which is connected to a shielded cable, some of the discharge current flows to ground via the cable shield. The noise of the discharge can be coupled to the cable center conductor through the transfer impedance of the cable, or by direct coupling to an exposed center conductor (in the case of improperly terminated shields), or directly through the shield itself. Figure 5.6-1 shows a test arrangement used for the ESD testing of cables. In an installed system, the ESD simulator voltage would be applied near the connectors at each end of a cable and to as much of the shielding between as is accessible.

In tests by the 3M Company, voltages larger than 500 volts could be induced on a center conductor by ESD if one end of the shield was unterminated. When the shield was terminated with a pigtail connection, the induced voltage was reduced to 16 volts. When the proper 360° backshell termination was made, a further reduction to just over a volt occurred. These results point out both the threatening nature of ESD and the necessity of proper cable shield termination.

## 5.7. Summary of ESD Prediction Technique

The procedure just given falls into four parts. First, the field at a distance of one meter from the discharge is calculated. Later, if desired, the field strength can be adjusted for distances other than one meter. A starting point for this field calculation is the information contained in figure 5.2-1.

Next, the susceptibility of the threatened circuit is calculated. Then the field and the susceptibility curves are combined to produce a resultant curve of induced voltage versus frequency. Graphical combining of the curves is most readily performed on a semilog scale as shown in figure 5.5-1.




NOTES:
1. TEST LEAD PLUS LOOP SHOULD BE 1 METER AWAY FROM ANY CONDUCTING SURFACE INCLUDING CEMENT FLOOR. A WOOD TABLE IS AN IDEAL TEST PLATFORM.
2. WHEN TEST CABLE IS LESS THAN 1.15 METERS IN LENGTH, THEN TEST PARALLEL SHALL BE SHORTER AND STARTED A MINIMUM OF 5CM (2") AWAY FROM EUT CONNECTOR.

FIGURE 5.6-1. ESD TEST CONFIGURATION (CABLE INDUCED)

The voltage induced in the susceptible circuit is found by converting from dB microvolts per meter per MHz to volts per meter per MHz.

Finally, the average value of the induced voltage over the frequency range is found by integrating over the entire spectrum of interest.

After appropriate adjustments are made for shielding or other attenuation factors, the induced voltage can be compared to the upset and damage levels of the threatened circuit or system.

## 5.8. ESD Testing Recommendations

It is recommended that ESD tests be performed because of the flight critical nature of new systems and the high susceptibility of broadband logic devices to ESD phenomena. The specific ESD test recommendations for flight critical systems follows.

Equipment, systems, and installed systems should be ESD tested according to table 5.8-1.

A soft failure is one which causes an alteration of data or missing data. A hard failure is one which requires a reset of equipment. Damage requires repair or replacement of a system, subsystem, or component.

All tests should start at the 2 kV level and should be advanced in 1 kV increments until a failure occurs (or the specified level is reached without failure).

TABLE 5.8-1. ESD TESTS

| Failure Mode | ESD Test Level | Simulator | Circuit Values |
|---|---|---|---|
| Soft | 8 kV | 150 pf | 1200 ohms |
| Hard | 12 kV | 150 pf | 1200 ohms |
| Damage | 25 kV | 150 pf | 1200 ohms |

## 6. CONDUCTED FAST TRANSIENTS ON POWER LINES

When an aircraft is on the ground and being serviced, it is possible that fast transients will be introduced on the primary power lines. The connection and disconnection of Auxiliary Power Units (APU) and switch-over from the APU to aircraft power are likely sources of these transients. In flight the operation of other on-board equipment may cause similar disturbances.

Typical power line transients are described in two comprehensive military standards. MIL-STD-461C defines pulse shapes, amplitudes, and time characteristics. MIL-STD-462 gives detailed test procedures. The applicable test is Conducted Susceptibility, CS06. A review of this specification is recommended.

Two types of transients are described. One is a heavily damped spike which starts out with a positive polarity, goes negative, and then rapidly decays to zero. The second transient is a trapezoidal pulse. Tests are conducted with pulse times of both 0.15 microseconds and 10 microseconds. The pulse amplitude is specified as 200 volts. These transients are shown in figure 6-1.

Although the military specification states that these pulses are to be applied to equipment and subsystem AC and DC power lines, such testing may be inappropriate in some cases. For example, low voltage DC power supplies may be unable to tolerate 200 volt spikes applied to their input leads. In recognition of these difficulties, MIL-STD-462, Notice 2, states "A detailed test procedure shall be included in the test plan to provide a practical measurement on the item to be tested, as the specified procedure often cannot be accomplished."

The CS06 tests are most appropriate for 28 Volt, Direct Current and 115 Volt, Alternating Current, 400 Hz primary power lines.

RTCA/DO-160B describes rather similar power line transients and test procedures. A thorough review of Section 17 of that document is recommended.

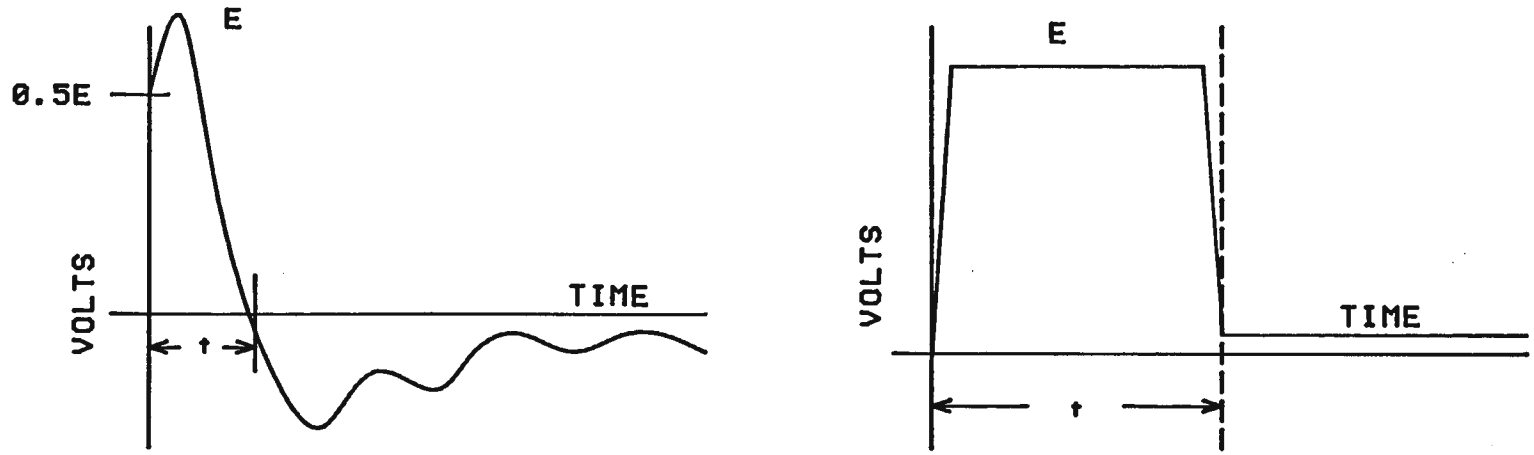
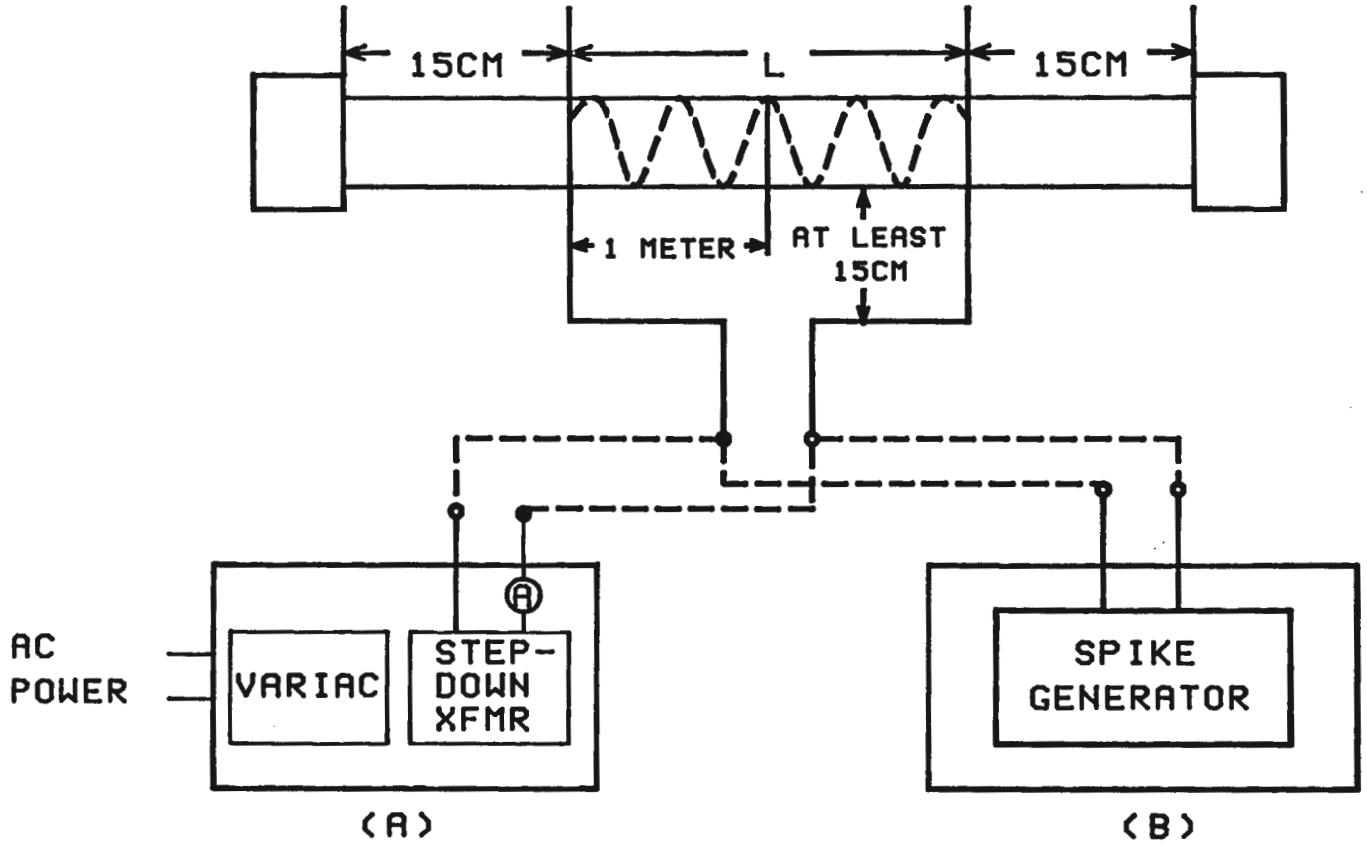



FIGURE 6-1. POWER LINE TRANSIENT TEST WAVESHAPES

## 7. RADIATED SUSCEPTIBILITY TO MAGNETIC INDUCTION FIELDS

The same transient spikes described in the previous section are also applied, under MIL-STD-461C, RS02, as magnetic induction fields to cables and equipment. The transient spikes must not cause "malfunction, degradation or deviation from specified indications..."

For these tests, a wire is wrapped around either cables or equipment, as shown in figures 7-1 and 7-2. The specified test spikes are applied from the spike generator. For complete details, see MIL-STD-461C and MIL-STD-462.




NOTE:

(1) L SHALL BE THE LENGTH OF THE CABLE IN THE ACTUAL INSTALLATION OR 1.5 METERS, WHICHEVER IS LESS.

FIGURE 7-1. RADIATED SUSCEPTIBILITY, MAGNETIC INDUCTION FIELD, CABLE TEST

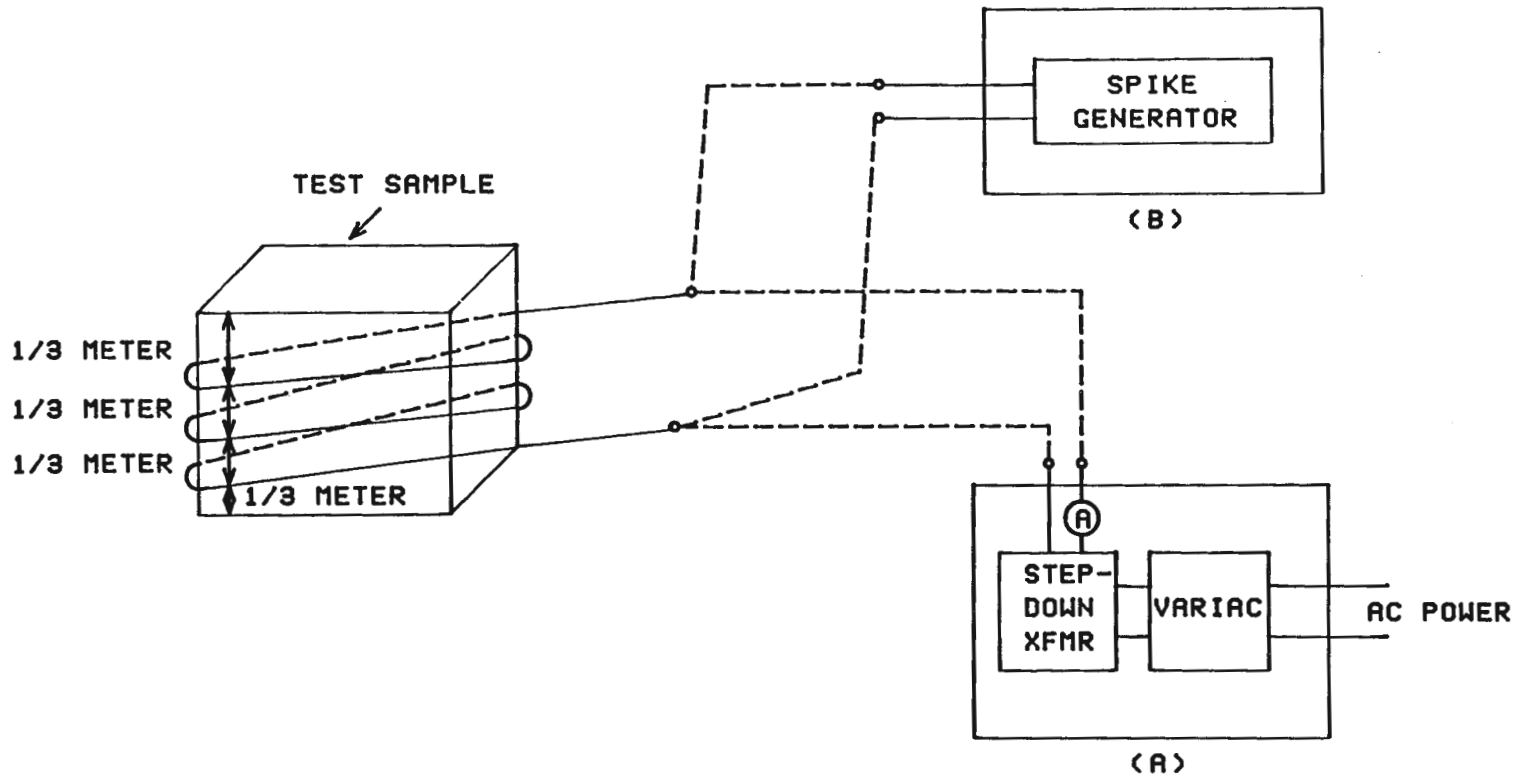



FIGURE 7-2. RADIATED SUSCEPTIBILITY, MAGNETIC INDUCTION FIELD, CASE TEST

## 8. MIL-STD-461C EMP TESTS

MIL-STD-461C defines Electromagnetic Pulse (EMP) pin level test limits under Conducted Susceptibility CS-10 and EMP cable test limits under Conducted Susceptibility CS-11. MIL-STD-462 gives detailed test procedures. A review of these references is recommended for a thorough understanding of these tests.

Briefly, a pin level test involves the direct connection of a test signal generator to a pin of a cable connector, followed by application of the test signal, and observation of the equipment under test for malfunction or degradation.

The cable test, Bulk Cable Injection (BCI), calls for inducing a specified current on a cable bundle or shield and again observing the equipment under test.

## 9. PROTECTIVE MEASURES

This section briefly describes various ways in which equipment and systems can be protected from the effects of fast transients.

Shielding is the most fundamental way of protecting both equipment and cables. All multiconductor cables, which carry signals that could be affected by transients, should be shielded. Other circuits, which can be regarded as generators of transient energy, should also be shielded to reduce their radiated emissions.

Particularly critical data can be transmitted on shielded twisted pairs. The twisting provides a measure of shielding effect by itself, and a single or even double shield provides further protection.

Coaxial lines with a single center conductor are frequently used. The braid-over-foil type is most effective. The foil provides nearly the effect of a solid outer shield, and the braid provides both physical protection and additional shielding. The type of fold is critical to the shielding effectiveness: a longitudinal "z-fold" or cigarette type fold is best. Spirally wrapped foil shields are less effective. The most effective coaxial cable shield is a solid metal outer conductor, but because of its lack of flexibility, it is ordinarily only used in short lengths within the avionic systems.

The method of shield termination is critical to maximizing the effectiveness of the shielding. All shields should be terminated with a $360^{o}$ connection of the shield to the back shell of the cable connector. Experiments have shown a 30 dB or more degradation of shielding when a "pigtail" connection as little as 1 inch in length is used instead of the $360^{o}$ connection.

Equipment case shielding is another line of defense against fast transients. Metal boxes provide effective shielding except against low frequency magnetic fields; however, holes, slots, seams, and apertures of any kind can allow entry of energy. In the case of ESD, some of the energy is in the 10 to 1000 MHz region and even fairly small apertures can degrade the shielding effectiveness.

A one-inch diameter hole has an attenuation of 40 dB at 60 MHz. This value scales downward (less attenuation) with increasing frequency at a rate of 20 dB per decade. Seams and slots can actually act as antennas. If the length of a slot or seam is one-half wavelength for an interfering signal, the slot or seam can resonate (like a magnetic dipole) and transfer signals with no attenuation at the frequency of resonance.

Filtering is another form of protection from fast transients. The energy in fast transients is primarily at higher frequencies. Low-pass filters which pass low frequencies and reject high frequencies can be effectively used on power lines and data lines to limit both radiation and reception of transient energy.

The cutoff frequency of the filter must be high enough to allow passage of the desired power or data frequency and low enough to prevent the flow of the transient energy. An example was given earlier in this chapter of the selection of a cutoff frequency in a low-pass filter. Filters can be designed in the form of discrete components. Filters may also be designed into the cable connectors. For many applications, a series inductance and a shunt capacitor provide adequate attenuation.

Ferrite beads placed over the end of a wire or cable can also provide a type of filtering. The common mode (line to ground) impedance is raised by the bead. For initially low impedance circuits, 6 to 10 dB of common mode attenuation results at frequencies from 10 to 100 MHz.

Various solid state protection devices exist which generally shunt transient energy to ground, while interrupting the normal flow of power or data for only brief periods. Variously known as transzorbs, surge suppressors, and varisters, these devices can all be considered as switches which are automatically operated by excessive voltage. They are connected in shunt-to-ground, and normally present a high impedance to the power or data line. When operated by the presence of the transient, they change state, go to a very low impedance, and short-circuit the line to ground. Designs are available in both unidirectional and bidirectional versions.

ESD protection requires attention to equipment case shielding. Control knobs and shafts can be grounded to the equipment case. This grounding will prevent entry of transient energy inside the case and will force the energy to flow on the outside to ground.

Attention should also be given to materials selection both in the aircraft structure and the equipment. Some materials, particularly plastics, are more apt to promote charge separation and accumulation, leading to ESD.

High voltage DC power supplies (found in video display units) are sources of charge and should be completely shielded to prevent charge transfer to nearby objects.

Keyboards on data entry units are often points of entry for ESD. Membrane- type keyboards are particularly vulnerable, but shielding can be incorporated between the keyboard and the sensitive components beneath.

# BIBLIOGRAPHY


Air Force Systems Command Design Handbook, AFSC DH1-4, Electromagnetic Compatibility, Wright Patterson Air Force Base (AFB).

Electrical Overstress/Electrostatic Discharge Symposium Proceedings, EOS-1, EOS-2, EOS-3, EOS-4, and EOS-5. Reliability Analysis Center, Rome Air Development Center, Griffiss AFB, NY 13441.

ESD Control in the Manufacturing Environment, Department of Defense

IEEE Transaction on Electromagnetic Compatibility: Many references available since 1958.

IEEE International Symposium on Electromagnetic Compatibility: Many annual references available since 1969.

Mardiguian, Michael, Electrostatic Discharge: Understand, Simulate and Fix ESD Problems, Interface Control Technologies, Library of Congress Number 85-80686, ISBN 0-932263-27-5.

McConnell, Roger A., DOT/FAA/CT-87/19, Avionics System Design for High Energy Fields.

MIL-B-82547, Electromagnetic Discharge Control Program for Protection of Electrical and Electronic Parts, Assemblies, and Equipment, (Excluding Electrically Initiated Explosive Devices).

MIL-STD-461-C, Electromagnetic Emission and Susceptibility Requirements for the Control of Electromagnetic Interference.

MIL-STD-462, Electromagnetic Interference Characteristics, Measurements of.

Military Handbook DOD-Hdbk-263, Electrostatic Discharge Control Handbook for the Protection of Electrical and Electronic Parts, Assemblies and Equipment.

National Communications System, NCS TIB 85-10, Volume 1, Electromagnetic Pulse/Transient Threat Testing of Protection Devices for Amateur/Military Affiliate Radio System Equipment.

Radio Technical Commission for Aeronautics, RTCA/DO-160B, Environmental Conditions and Test Procedures for Airborne Equipment.

R&B Enterprises, EMP Testing Handbook, R&B Enterprises, EMP Division, 20 Clipper Road, West Conshohocken, Pa. 19428.

# GLOSSARY

BROADBAND. A frequency spectrum which is wide compared to the bandwidth of the device used to detect it.

DECIBEL. A dimensionless unit for representing the ratio of values of power or voltage.

dBμV. Decibels referred to one microvolt. Zero db represents one microvolt.

FALL-TIME. The time required for pulse amplitude to go from a predefined magnitude to a given level.

FOURIER TRANSFORM. A mathematical method for deriving the frequency spectrum from a time dependent function.

HARD FAILURE. A failure that requires a reset of the equipment.

JOULE. A unit of energy equal to one watt-second.

LOW-PASS FILTER. An electrical circuit which allows the passage of low frequencies and prevents the passage of high frequencies.

PIN LEVEL TEST. An EMC test in which voltage or current is applied directly to a conductor at a connector pin.

Q. The quality factor of a resonant circuit which is the ratio of the energy stored to the power dissipated per cycle.

RESONANCE. Resonance occurs in an electrical circuit when the energy stored in the inductance is equal to the energy stored in the capacitance.

RISE-TIME. The time required for a voltage pulse to reach a predefined percentage of the amplitude from a given level.

SHIELDING. Any metallic structure such as the aircraft fuselage or the woven braid on a cable that provides protection against electromagnetic fields.

SINUSOID. A wave form that follows the mathematical values of a sine function.

SOFT FAILURE. A failure which causes an alteration of data or missing data.

UPSET. A condition in which the state of a digital device is unintentionally altered, but may be restored by automatic means or by operator intervention.

## ACRONYMS AND ABBREVIATIONS

| | |
|---|---|
| AC | Alternating Current |
| APU | Auxiliary Power Units |
| BCI | Bulk Cable Injection |
| CMOS | Complimentary Metal-Oxide Semiconductor |
| dB | decibels |
| DC | Direct Current |
| DOD | Department of Defense |
| EMC | Electromagnetic Compatibility |
| EMCad$^{tm}$ | Electromagnetic Computer aided design |
| EMP | Electromagnetic Pulse |
| ESD | Electrostatic Discharge |
| kHz | Kilohertz |
| MHz | Megahertz |
| NEMP | Nuclear Electromagnetic Pulse |
| pf | picofarad |
| R-C | Resistor-Capacitor |
| RTCA | Radio Technical Commission for Aeronautics |
| TTL | Transistor-Transistor Logic |

U.S. Department of Transportation
**Federal Aviation Administration**

DOT/FAA/CT-88/10

# HANDBOOK-VOLUME II
# DIGITAL SYSTEMS VALIDATION

## CHAPTER 13
## LIGHTNING STUDIES

PREPARED BY:

COMPUTER RESOURCE MANAGEMENT, INC.
950 HERNDON PARKWAY, SUITE 360
HERNDON, VIRGINIA 22070

PREPARED FOR:

FEDERAL AVIATION ADMINISTRATION
TECHNICAL CENTER
ATLANTIC CITY INTERNATIONAL AIRPORT, NEW JERSEY 08405

NOTICE

This document is disseminated under the sponsorship of the U.S. Department of Transportation in the interest of information exchange. The United States Government assumes no liability for the contents or use thereof.

The United States Government does not endorse products or manufacturers. Trade or manufacturers' names appear herein solely because they are considered essential to the objective of this report.

TABLE OF CONTENTS


| Section | | | Page |
|---|---|---|---|
| 1. | INTRODUCTION | | 13-1 |
| 2. | BACKGROUND | | 13-3 |
| | 2.1. | History | 13-3 |
| | 2.2. | Lightning Environment | 13-8 |
| | 2.3. | Interaction and Coupling to Aircraft | 13-12 |
| | | 2.3.1. Standard Aircraft Lightning Environment | 13-12 |
| | | 2.3.2. Structural Materials Effect on Strike Zones | 13-18 |
| | 2.4. | Lightning Effects on Aircraft Systems | 13-18 |
| | | 2.4.1. Range of Lightning Effects | 13-18 |
| | | 2.4.2. Direct Effects on Non-Metallic Structures | 13-24 |
| | | 2.4.3. Indirect Effects on Non-Metallic Structures | 13-26 |
| | | 2.4.4. Effects on Aircraft Systems | 13-26 |
| 3. | INDIRECT EFFECTS PROTECTION - DESIGN APPROACH | | 13-29 |
| | 3.1. | Design Margin | 13-29 |
| | 3.2. | Requirements for Grounding and Bonding | 13-31 |
| | 3.3. | Guidelines for Cable Routing and Equipment Installation | 13-32 |
| 4. | INDIRECT EFFECTS PROTECTION - TEST METHODS | | 13-43 |
| | 4.1. | Review of Lightning Simulation Methods | 13-43 |
| | | 4.1.1. Lightning Simulation Technology | 13-43 |
| | | 4.1.2. Considerations in Selecting Simulation Techniques | 13-44 |
| | 4.2. | Lightning Pulse Generator Techniques | 13-47 |
| | | 4.2.1. Low-Level Swept Continuous Wave | 13-48 |
| | | 4.2.2. Low-Level Fast-Rise Pulse | 13-53 |
| | | 4.2.3. High-Level Fast-Rise Pulse | 13-57 |
| | | 4.2.4. Severe Full-Threat Pulse | 13-58 |
| | | 4.2.5. Shock-Excited Pulse | 13-59 |
| | | 4.2.6. Comparison of Test Methods | 13-60 |
| | | 4.2.7. Return Circuit Configuration | 13-61 |
| | 4.3. | Data Quality Verification | 13-61 |
| | 4.4. | Bench Tests | 13-63 |

TABLE OF CONTENTS (Continued)


| Section | | Page |
|---|---|---|
| 5. | INDIRECT EFFECTS PROTECTION - ANALYSIS TECHNIQUES | 13-69 |
| | 5.1. Modeling Methods for Composite Structures | 13-71 |
| | 5.2. Computer Modeling Codes | 13-72 |
| 6. | SUPPLEMENTAL EXAMPLES | 13-77 |
| | 6.1. Coupling to Wiring | 13-77 |
| | 6.2. Composite Wing Model | 13-84 |
| BIBLIOGRAPHY | | 13-87 |
| GLOSSARY | | 13-90 |
| ACRONYMS | | 13-93 |

LIST OF ILLUSTRATIONS


| Figure | | Page |
|---|---|---|
| 2.1-1 | PROBLEMS ASSOCIATED WITH PROTECTION OF DIGITAL SYSTEMS IN ADVANCED TECHNOLOGY STRUCTURES | 13-5 |
| 2.1-2 | SURFACE TRANSFER IMPEDANCE FOR VARIOUS MATERIALS | 13-6 |
| 2.2-1 | LIGHTNING FLASH CURRENT WAVEFORM | 13-9 |
| 2.2-2 | IDEALIZED LIGHTNING CURRENT WAVEFORMS | 13-11 |
| 2.2-3 | MULTIPLE STROKE CHARACTERIZATION | 13-13 |
| 2.3-1 | AIRCRAFT LIGHTNING STRIKE INCIDENTS AS A FUNCTION OF ALTITUDE | 13-15 |
| 2.3-2 | INTERACTION OF AIRCRAFT WITH CHARGE CENTERS | 13-16 |
| 2.3-3 | TYPICAL PATH OF SWEPT FLASH ATTACHMENT POINTS | 13-17 |
| 2.3-4 | LIGHTNING CURRENT COMPONENTS EXPECTED IN EACH ZONE | 13-19 |
| 2.3-5 | LIGHTNING CURRENT COMPONENTS EXPECTED IN EACH ZONE | 13-20 |
| 2.3-6 | TYPICAL LIGHTNING STRIKE ZONES | 13-21 |
| 2.3-7 | TYPICAL LIGHTNING STRIKE ZONES FOR HELICOPTERS | 13-22 |
| 3.1-1 | DESIGN MARGIN DEFINITION | 13-30 |
| 3.3-1 | METHODS OF PENETRATING THE STRUCTURE WITH VARIOUS CONDUCTORS | 13-35 |
| 3.3-2 | COMMON AND DIFFERENTIAL-MODE COUPLING CHARACTERISTICS FOR STRUCTURE RETURN AND TWO-WIRE CIRCUITS | 13-37 |
| 3.3-3 | CABLE ROUTING NEAR CONDUCTIVE STRUCTURE | 13-39 |
| 4.2-1 | TYPICAL LOW-LEVEL SWEPT CONTINUOUS WAVE TEST SETUP | 13-50 |
| 4.2-2 | EXAMPLE TRANSFER FUNCTION ANALYSIS SEQUENCE | 13-52 |
| 4.2-3 | TYPICAL PULSE SIMULATION TEST SETUP | 13-54 |
| 4.2-4 | COMPARISON OF FREQUENCY SPECTRA | 13-56 |
| 4.2-5 | SIMPLIFIED RETURN CIRCUIT | 13-62 |
| 4.3-1 | COMPARISON OF SIGNAL AND NOISE IN THE FREQUENCY DOMAIN | 13-64 |
| 4.3-2 | COMPARISON OF SIGNAL AND NOISE IN THE TIME DOMAIN | 13-65 |
| 4.4-1 | IDEAL BENCH TEST WAVEFORMS | 13-67 |
| 4.4-2 | CABLE CURRENT INJECTION TEST CONFIGURATION | 13-68 |
| 5-1 | ELEMENTS OF EM MODELING AND ANALYSIS FOR LIGHTNING INDIRECT EFFECTS | 13-70 |
| 5.2-1 | MULTILEVEL LIGHTNING COUPLING ANALYSIS APPROACH | 13-73 |
| 6.1-1 | REDIST EXAMPLE MODEL GEOMETRY | 13-78 |
| 6.1-2 | SHIELD TRANSFER FUNCTION FOR METAL/GE BOX FOR SEVERAL SHIELD LOCATIONS | 13-79 |
| 6.1-3 | SHIELD CURRENT FOR METAL/GE BOX FOR SEVERAL SHIELD LOCATIONS FOR 200 kA ATTACHED LIGHTNING | 13-79 |
| 6.1-4 | MAGNETIC FIELD CONTOURS FOR METAL/GE BOX AT 1 kHz | 13-80 |
| 6.1-5 | MAGNETIC FIELD CONTOURS FOR METAL/GE BOX AT 1 MHZ | 13-80 |
| 6.1-6 | PIN VOLTAGE WAVEFORMS FOR SEVERAL SHIELD LOCATIONS | 13-82 |
| 6.1-7 | PIN VOLTAGE WAVEFORMS FOR SHIELD LOCATION #6 FOR VARIOUS LINE LENGTHS | 13-83 |

LIST OF TABLES


| Table | | Page |
|---|---|---|
| 2.2-1 | IDEALIZED LIGHTNING WAVEFORM PARAMETERS | 13-14 |
| 2.3-1 | COMMERCIAL AIRPLANE LIGHTNING STRIKE STATISTICS | 13-23 |
| 2.4-1 | RANGE OF EFFECTS FROM ATMOSPHERIC ELECTRICITY INTERACTIONS WITH AIRCRAFT | 13-25 |
| 3.2-1 | GROUND SYSTEM CURRENTS FROM A VARIETY OF EM SOURCES | 13-33 |
| 3.2-2 | TYPICAL INTERFERENCE TOLERANCE LEVELS BETWEEN UNITS OF ELECTRONIC EQUIPMENT | 13-34 |
| 3.2-3 | TOLERABLE LEVELS FOR GROUND IMPEDANCE IMPLIED BY TABLE 3.2-1 AND TABLE 3.2-2 VALUES | 13-34 |
| 3.3-1 | ESTIMATED SHIELD CURRENTS DUE TO A DIRECT LIGHTNING STRIKE | 13-40 |
| 3.3-2 | COUPLING RESPONSES TO SUBSYSTEM WIRING DUE TO A DIRECT LIGHTNING STRIKE | 13-41 |
| 4.1-1 | PRIOR RESEARCH AND DEVELOPMENT TESTS ON FULL-SCALE AIR VEHICLES | 13-45 |
| 4.2-1 | SUMMARY OF SIMULATION TECHNIQUE CAPABILITIES | 13-49 |
| 5.2-1 | CAPABILITIES OF COMPUTER CODES FOR MODELING EM COUPLING TO AIRCRAFT | 13-74 |
| 6.2-1 | COMPOSITE WING RESISTANCE MODEL RESULTS VS. MEASUREMENTS | 13-85 |

# 1. INTRODUCTION

While the effects of lightning strikes on metal aircraft are considered minimal, there is a greater threat to advanced technology aircraft. There are two major reasons for this increased threat. Digital electronic components are more susceptible to electrical transients than analog systems in older aircraft. Also, advanced composite structural materials offer less electrical shielding than metallic structures. Both of these factors reduce the inherent protection of electronic/electrical systems found on previous technology, all-metal aircraft.

Competition in the marketplace for aircraft sales and the increasing cost of fuel is developing pressure on manufacturers to employ advanced technology electronic equipment and materials, such as bonded honeycomb, kevlar, fiberglass, or Graphite Epoxy (G/E), in the next generation of aircraft. This is evident both in large transport and in general aviation aircraft currently under development. Several general aviation aircraft that employ advanced technology are nearing or are in the certification process. These include the Beech "Starship," Lear Fan, and AVTEK-400. In addition to the all electric engine control (Pratt & Whitney 2037) for the Boeing 757, transport aircraft manufacturers are researching the use of advanced composite structures, digital data busses (beyond ARINC 429), and all-electric systems.

Advanced technology materials and new structural fabrication methods, such as using adhesives in place of fasteners to minimize drag, reduce manufacturing costs as well as corrosion and fatigue. Several potential problems have prevented widespread use of the new structural technology. These include the variability of: lightning protection, static electrification, Electromagnetic Compatibility (EMC), mechanical and electrical parameters, impact resistance, effect of environmental factors, and production controls.

This chapter addresses the topic of lightning studies and the impact of lightning strikes to aircraft and digital systems.

Section 2 outlines some of the problems associated with the use of digital systems and composite materials in advanced technology aircraft. Currently accepted information about lightning current waveforms is reviewed. The lightning environment, which an aircraft might be expected to encounter, is discussed. A few examples of lightning effects on aircraft systems are given.

Section 3 covers the design approach to lightning indirect effects protection. A design margin must be established. Requirements for grounding and bonding must be taken into account. Preferred methods for equipment installation and cable routing to provide optimum protection from Electromagnetic (EM) hazards are reviewed.

Test methods for verification of protection design are reviewed in section 4. These test methods include a variety of lightning simulation techniques and equipment bench test techniques. Several simulation techniques are discussed and a comparison of test results for four methods is given. Verification of data quality is an important aspect of these test methods.

Analysis techniques are used to validate experimental results. Computer modeling methods are covered in section 5. Several techniques must be modified to be valid for composite structures where the dominant coupling mechanism is dependent on the resistance of the structure. The capabilities of several computer codes useful for predicting EM coupling due to lightning are summarized.

Section 6 contains worked examples to illustrate prediction of EM coupling to wiring. The examples cover very simple techniques as well as the procedure for a more detailed vehicle analysis.

## 2. BACKGROUND

### 2.1. History

Aircraft are exposed to a wide variety of EM environments from onboard and external sources including the following major items:

- The electrical power and electronics equipment often cause electric currents to flow in the structure, if not by design, perhaps through equipment faults or short circuits.

- Lightning strikes to an aircraft result in large currents of short duration. These currents may flow in the structure and in any of the metallic plumbing, control cables, or wiring for the electrical/electronic systems.

- Antennas on the aircraft and nearby high energy Radio Frequency (RF) sources may cause large RF currents in the structure.

Lightning is the most severe of these EM environments. Direct strike lightning currents may be as high as 200 kA and the rate of rise may exceed 100 kA per microsecond. The effects of such currents will be subsequently discussed. Power system and electronic fault currents may reach a few thousand amperes. RF current density may reach a few tens of amperes per meter. Lightning and electric circuit faults act upon a small point of contact; therefore, the structure could be damaged at entry points. The RF currents are not likely to result in structural damage, but could cause a functional disruption of digital electronic circuits and possibly of analog circuits as well.

Aircraft lightning strikes produce two types of effects.

- Direct effects: Physical damage from arcing and sparking.

- Indirect effects: Disruption of the electronic/electrical systems from electrical transients in the wiring and structural elements.

The trend in avionic/electrical equipment toward digital circuits having lower operating voltage and power levels adds to the concerns regarding protection against the indirect effects of lightning strikes and static electrification. The poor (lower) conductivity of composite materials and the bonds between structural members make it difficult to obtain the 2.5 milliohm bonding and grounding required by current military specifications (MIL-B-5087B(ASG), 1970). If the 2.5 milliohm specification were applied, the assessment indicates that all electrical systems would be safe.

Structures composed of many of the newer materials will be far more resistive than the present 2.5 milliohm requirement. Kevlar is an insulator and GFE

structures may have resistances of a few ohms. Designs using these materials should allow the addition of enough metal so that protection against lightning strikes may be achieved. These designs should also allow the addition of metallic pathways in the grounding and bonding systems so that onboard systems will function properly in the presence of currents anticipated from normal or fault conditions in onboard equipment.

There are also concerns about potential interference between different onboard digital systems and sources of Electromagnetic Interference (EMI). The EM shielding effectiveness of a composite fuselage without seams or joints is one to two orders of magnitude less than that of an aluminum fuselage. Seams and joints reduce the shielding to a practical upper limit of 25 to 40 dB depending on size and number of seams and joints. The EM Compatibility (EMC) between digital and RF circuits using the structure as a return path is more critical for new aircraft than for older aircraft.

Since advanced aircraft digital electronics operate at a few volts compared with a few tens of volts for analog systems, the design margin in future advanced aircraft electronic systems may be reduced by two to three orders of magnitude. (See section 3.1.) Considerable design efforts may be required to accomplish the protection for these advanced technology aircraft because of the reduced margins of safety between the EM induced transients (stress) and the ability of future technology equipment to withstand these transients (strain). Figure 2.1-1 shows the problems associated with EM protection of digital systems and changing electrical parameters when replacing metal structure with composite materials. Figure 2.1-2 indicates the variability in transfer impedance of various materials.

The characteristics of electrical transients induced in aircraft wiring and avionics systems are affected by the system response of the entire aircraft to the lightning stimulus. Induced coupling and susceptibility tests may be conducted on electrical hardware and associated wiring at the subsystem level. Determination of the voltage and current levels induced into the equipment by a lightning strike must take into consideration the structural interaction with the arc, the subsequent coupling of the magnetic and electric fields generated by lightning currents on the external surfaces, and coupling to internal wiring and equipment. The presence of an aircraft in the lightning channel may modify the natural lightning current waveform as a result of the resonant responses of the aircraft structure.

The demonstration and validation of the lightning/static electrification protection design is more critical for the new materials and electronics technology than for older aircraft technology. Extrapolation of prior protection designs on metal aircraft to those on advanced technology aircraft is not valid because the protection technology is completely different. Very few protection measures were required to protect older metal aircraft from indirect effects.

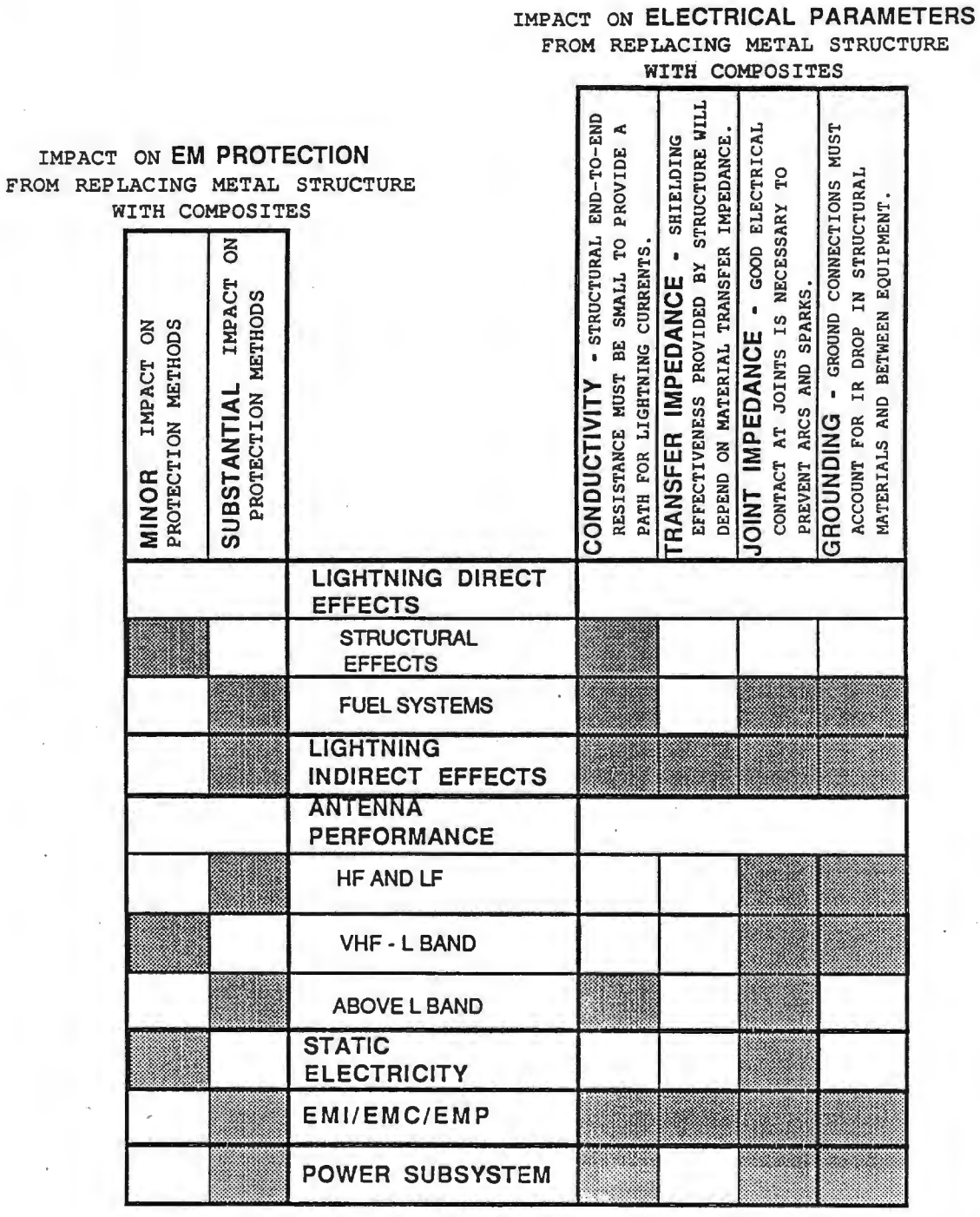



FIGURE 2.1-1. PROBLEMS ASSOCIATED WITH PROTECTION OF DIGITAL SYSTEMS IN ADVANCED TECHNOLOGY STRUCTURES

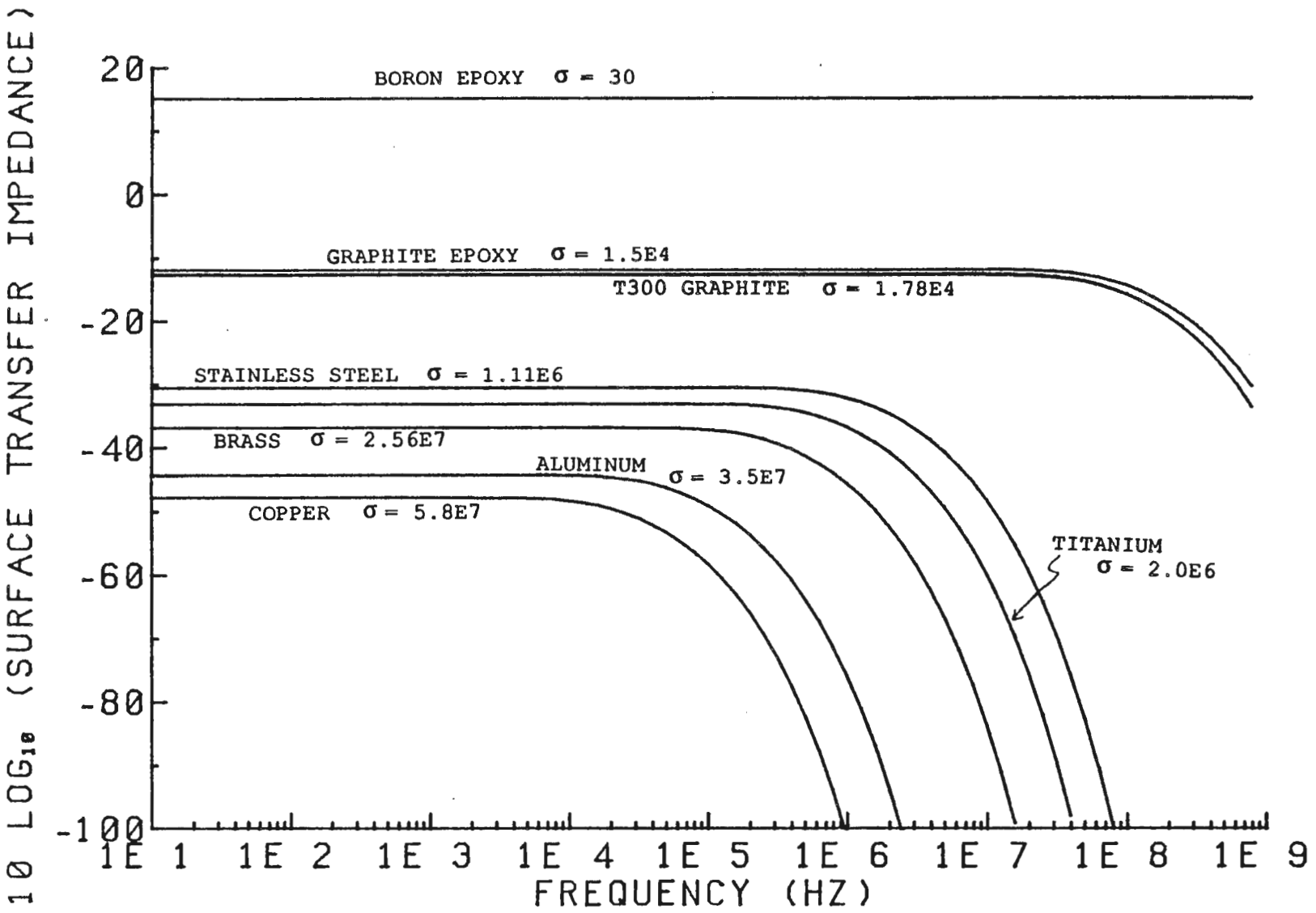



FIGURE 2.1-2. SURFACE TRANSFER IMPEDANCE FOR VARIOUS MATERIALS

The design margins did not exist for the older designs because of prior experience with the inherent insensitivity of metal aircraft to lightning strikes. Advanced technology aircraft are judged to have three to four orders of magnitude less design margin based upon the reduced shielding effectiveness of the fuselage and the sensitivity of the digital electronics (Sommer, 1981). It is natural to question the protection designs with such a great change in design margins.

Because of the increased potential for damage to electronic systems in advanced technology aircraft, simulated lightning tests must be performed to verify the adequacy of the design measures utilized to prevent serious loss or damage resulting from a lightning strike. It is recognized that the necessary protection verification test cannot be performed as a go/no-go test at full lightning threat levels, discussed in section 2.2, for several reasons as follows:

- The required electrical storage capacity in the lightning simulator would be excessive for a 200,000 amp discharge through an aircraft.
- The required fast rise-time ($2.0 \times 10^{11}$ amps/second) for a 200,000 amp discharge could not be attained.
- The discharge currents and charge densities around the aircraft would be different in flight.
- There is no predictable method to update test results for airplane modifications.
- The perceived lightning threat may be changed in the future as more data become available through the efforts of the National Aeronautics and Space Administration (NASA), Federal Aviation Administration (FAA), and others investigating the effects of lightning on aircraft in flight.

Lightning and static electrification protection design and demonstration of the design for aircraft and ground based systems are current topics for considerable technical research and development. The electrical impedance of a full-scale aircraft precludes the use of severe lightning current pulses such as those used for component indirect effects tests. Consequently, lower level current pulses have been used to evaluate the transients produced by a lightning strike. Lightning simulation testing is a very challenging technical area because it is very difficult, if not impractical, to conduct a full-scale simulated lightning test on large aircraft (MIL-STD-1757 and report of SAE Committee AE4L, 1978). The energy storage and electrical circuits capable of delivering a full-threat 200 kA peak current require simulator voltages greater than can be achieved in air (in excess of 2 million volts for a 25-foot long aircraft). Facilities to generate these conditions presently cost over a million dollars. Even with costly tests, there are no methods to update the results for airplane modifications, or methods to incorporate changes in the lightning threat. For the last several years there have been regular conferences dedicated to these topics. (See International Aerospace and Ground Conference on Lightning and Static Electricity in the Bibliography.)

Because of the practical considerations regarding simulator limitations, several different simulation test techniques are used for lightning and static electricity design demonstrations on full vehicles. The simulation and analysis methods are specialized to the unique character of lightning attachment, vehicle charging, high arc current flow, and vehicle discharging.

There are several variants on the high current pulse simulation techniques that are used for full vehicle lightning tests. The four principal lightning simulation test techniques are:

- Low-level swept Continuous Wave (CW).
- Low-level fast rise pulse.
- Full threat fast rise pulse.
- Shock-excitation.

Included in the accuracy of technique are the analytical models and computations used to verify and validate the test data. For some methods, analytical techniques provide extrapolation of the test data to other environments and guidance for modifications of the aircraft design.

## 2.2. Lightning Environment

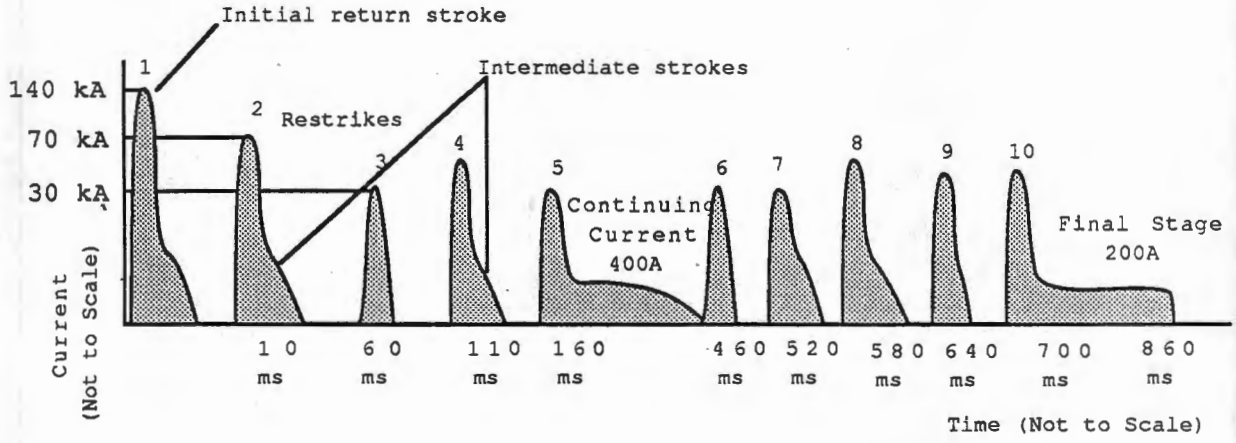
Lightning environments have varying intensities and durations because of several different physical processes involved. The variation in parameters makes it impossible to specify the environment during any particular lightning flash. Lightning occurs most frequently due to charge separation in thunderstorm clouds. During a storm, air convection currents cause charge centers to build up within clouds. Because of mutual capacitance, the potential between these charge centers increases and will eventually break down a portion of the air path between them, redistributing the charges. The number of strokes within each flash and the peak currents depend on many factors including the storm intensity, the charge levels in the cloud before the leader has completed a path to the opposite charge, and the cloud base height (Melander and Cooley, 1984). These factors are variable, and more than an order of magnitude difference may occur between the currents of successive strikes.

Lightning flashes are of two fundamentally different forms, cloud-to-ground and inter/intracloud. Discharges may originate from either a positive or a negative charge center in a cloud and terminate at the ground or in an opposite charge center in the cloud. A negative discharge is characterized by several intermittent strokes and continuing currents. A positive discharge occurs only a small percentage of the time. It is characterized by both higher average current and longer duration within a single stroke. (See figure 2.2-1.)

Once preliminary breakdown within a cloud has begun, the breakdown process proceeds by a stepped leader which propagates from the cloud toward the ground or towards another charge center. As a stepped leader approaches the source of opposite charge, high electrical fields are produced. These fields give rise to streamers initiated at the ground or charge center until one of these

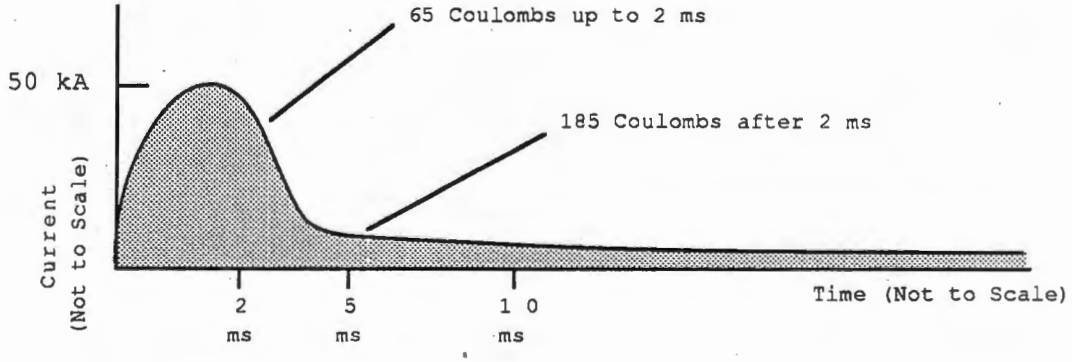
For Each Stroke:
Time to peak current = 1.5 μs
Time to half value = 40 μs

For the complete flash:
$\int i^2\,dt = 1.9E6\ A^2\ s$




(A) Severe negative lightning flash current waveform.
(Courtesy of Cianos/Pierce)

For the complete flash:
$\int i^2\,dt = 2.5E6\ A^2\ s$




(B) Moderate Positive Lightning Flash Current Waveform

FIGURE 2.2-1. LIGHTNING FLASH CURRENT WAVEFORM

streamers contacts the approaching stepped leader. The average velocity of propagation of the stepped leader is about one meter per microsecond for each step tens of meters in length and separated in time by approximately 50 $\mu$s. The typical current is 1000 amperes and the average charge in the entire stepped leader channel is about 5 coulombs.

The high peak current associated with lightning occurs after the stepped leader reaches the ground and forms the return stroke of the lightning flash. This return stroke occurs when the charge in the leader channel is suddenly able to flow into the low impedance ground. The return stroke current expected value is much larger than the leader current and ranges between 10 and 30 kA. Although in rare cases a larger current can occur, a peak current of 200 kA represents a severe stroke and is considered a practical maximum value of lightning current. In rare cases, a larger current can occur. The current in the return stroke has a fast rate of change, ranging between 10 and 20 kA/$\mu$s. The maximum practical value is considered to be 200 kA/$\mu$s. No correlation has been shown to exist between peak current and peak rate of rise. The return stroke may be followed by a dart leader which initiates one or more subsequent return strokes or restrikes. Restrikes will continue until the charge is neutralized. Subsequent strokes typically have smaller magnitudes and faster rise times but comparable rates of change to the first return stroke. A lightning flash may contain as many as 24 restrikes following the first return stroke. Restrikes occur at intervals of several tens of milliseconds as different charge centers in the cloud are tapped.

The total charge transported by the lightning return stroke is relatively small, a few coulombs. Most of the charge is transported as a continuing current. Currents on the order of a few hundred amperes may flow in the ionized channel for periods up to one second. The maximum charge transferred is 200 coulombs. Continuing currents may link successive return strokes.

A discharge between clouds or charge centers within a cloud is characterized by a multitude of relatively low level intermittent strokes and continuing currents. The average velocity of propagation of the streamers is $10^5$ meters per second. The peak current associated with an inter/intracloud discharge is in the range of 3 to 10 kA and occurs when opposite charge centers have been connected by a streamer or stepped leader. Each restrike within an inter/intracloud lightning flash may contain 20 separate pulses, known as a multiple burst. The number of restrikes is comparable to a cloud-to-ground flash.

Important parameters of a lightning flash are the peak current amplitude, maximum current rate-of-rise, energy, duration, rise-time, and the number of strokes within each flash. Statistical representations of these measured parameters are found in Melander and Cooley (1984). The great bulk of this statistical data is based on measurements of cloud-to-ground flashes. Several flight test programs within the past few years have gathered data on inter/intracloud lightning. These data are used to define parameters for lightning threat levels.

Representations of the various portions of lightning flashes have been developed by the Society of Automotive Engineers (SAE) committee AE4L as shown in figure 2.2-2. Multiple stroke and multiple burst waveforms are characterized

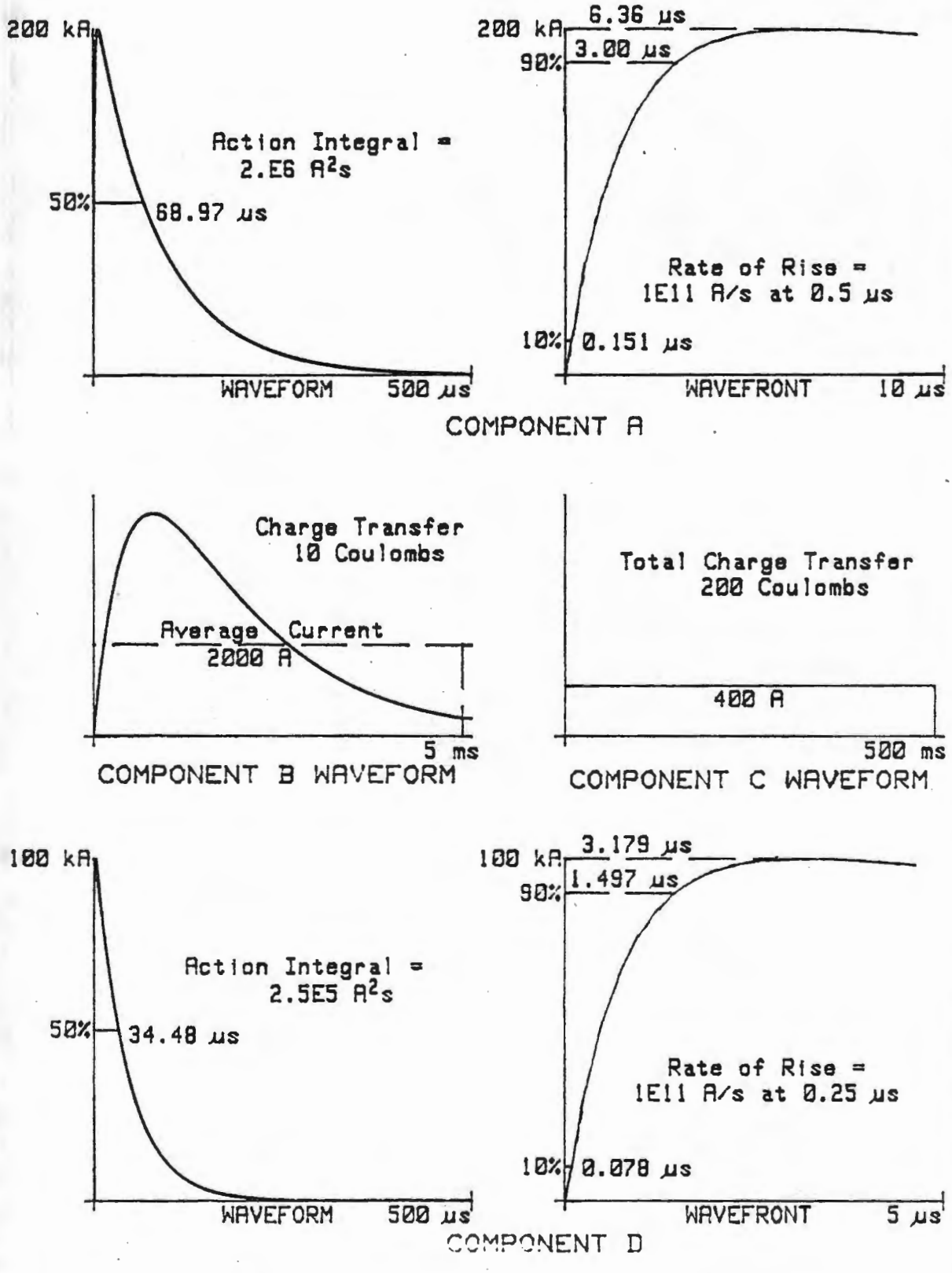



FIGURE 2.2-2. IDEALIZED LIGHTNING CURRENT WAVEFORMS

in figure 2.2-3. Table 2.2-1 summarizes the parameters for each waveform. A double exponential of the form shown below defines each component waveform.

$$I(t) = I_o \times (e^{-\alpha t} - e^{-\beta t})$$

## 2.3. Interaction and Coupling to Aircraft

It is not possible to precisely quantify the lightning environment during any particular strike to an aircraft in flight. At low altitudes, cloud-to-ground strikes are more likely. Inter/intracloud strikes are more common at higher altitudes. Statistics on the number of strikes to aircraft as a function of altitude are given in figure 2.3-1.

It is generally accepted that the presence of an aircraft may trigger the leader discharge process between charge centers by disrupting the leader path. Recent experience by the FAA and the Air Force Wright Aeronautical Laboratory (AFWAL) in-flight program using a CV-580 to intercept lightning strikes indicates leader growth out of the wing of the test aircraft. Figure 2.3-2 illustrates the interaction of the aircraft with a charge center. When the aircraft is part of a leader not much current flows. The major effect occurs when the path is complete to the opposite charge center, and the return current passes through the aircraft.

A typical lightning flash consists of 20-200 current pulses over a one to two second interval. Because the aircraft is moving, the flash will sweep aft from an initial attachment point, all the while reattaching to spots along the line of flight aft from the original point of attachment. A portion of the current will enter the aircraft at several points along the path from the initial and swept attachment points. Figure 2.3-3 shows a typical path of attachment points for a swept flash. If the attachment is a trailing edge surface, then the current will hang on at this initial point and flow through there for the entire flash.

### 2.3.1. Standard Aircraft Lightning Environment

Since different parts of an aircraft must tolerate different levels of current, a standard definition has evolved defining the levels of current applicable to different zones of the aircraft. The SAE-AE4L committee has established generally accepted definitions of the possible strike zones as follows:

- Zone 1A: Initial Attachment Point with low probability of flash hang on, such as a leading edge.
- Zone 1B: Initial attachment point with high probability of flash hang-on, such as a trailing edge.
- Zone 2A: A swept stroke zone with a low probability of flash hang-on, such as a wing midspan.
- Zone 2B: A swept stroke zone with high probability of flash hang-on, such as the wing inboard trailing edge.

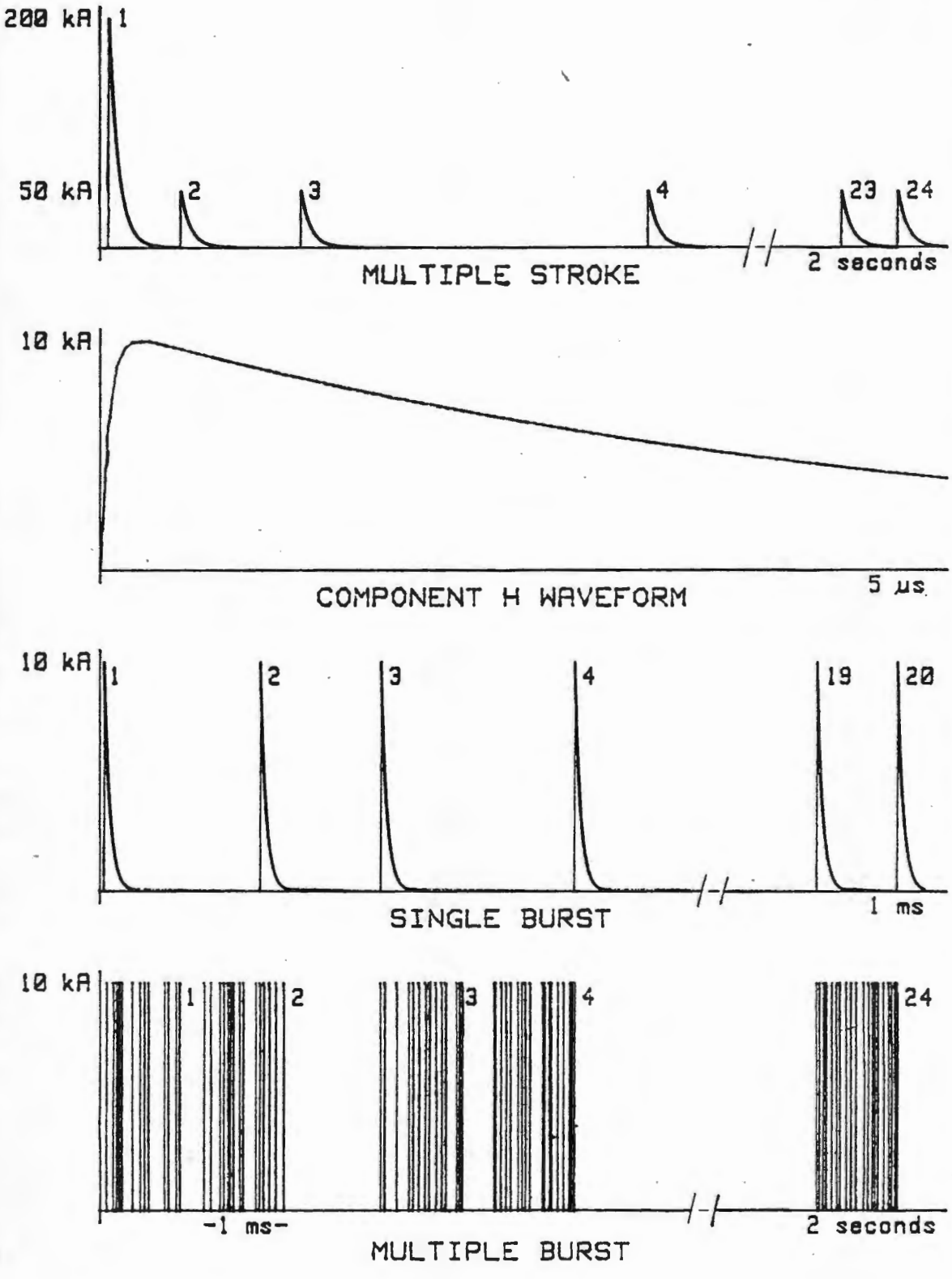



FIGURE 2.2-3. MULTIPLE STROKE CHARACTERIZATION

TABLE 2.2-1. IDEALIZED LIGHTNING WAVEFORM PARAMETERS

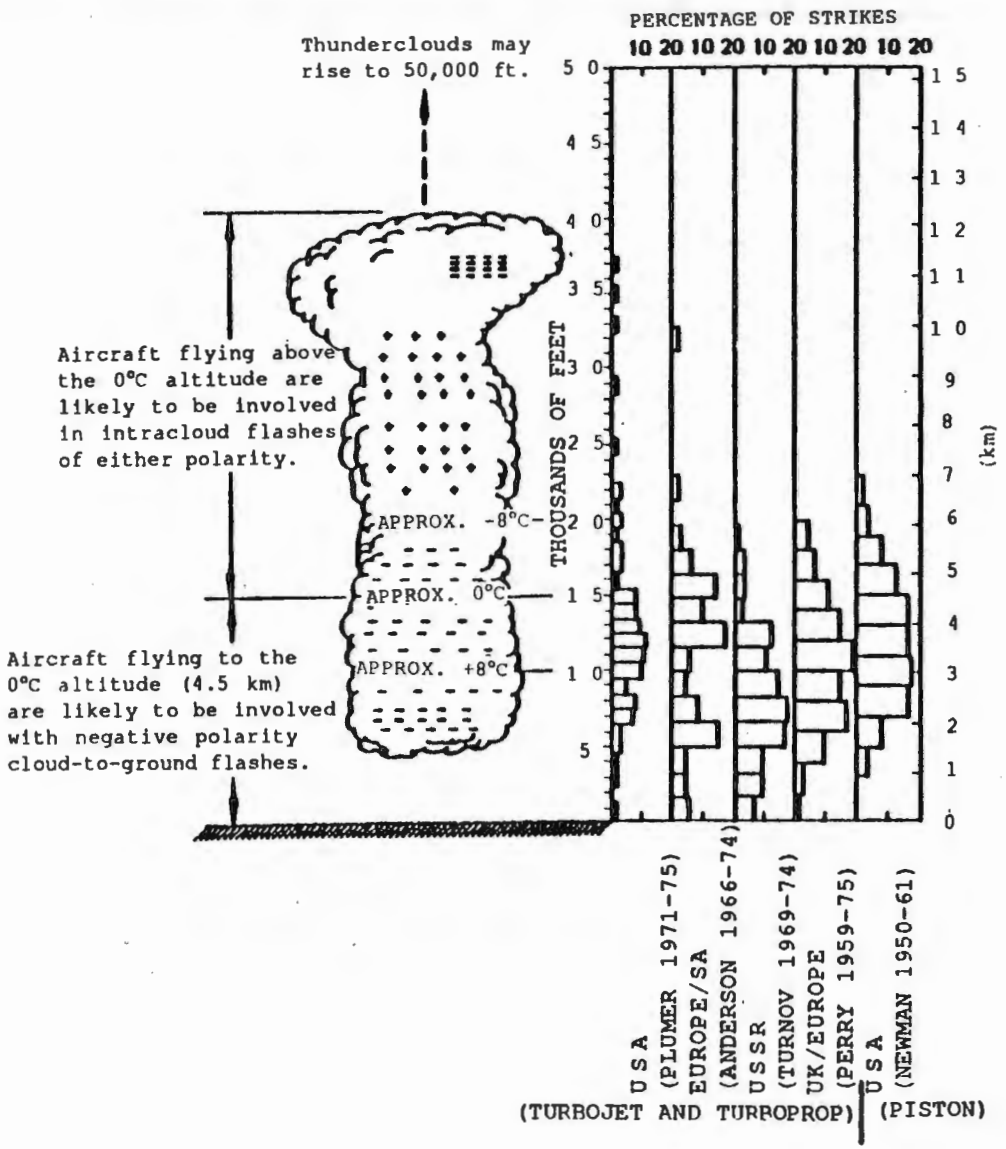
| Threat | $I_o$ (A) | Alpha (1/s) | Beta (1/s) | Peak Current (kA) | Maximum Rate-of-Rise (kA/μs) | Action Integral ($A^2s$) |
|---|---|---|---|---|---|---|
| Severe Component A[1,2,3] | 218810 | 11354 | 647265 | 200 | 140 | $2.0 \times 10^6$ |
| Intermediate Current Component B[2] | 11300 | 700 | 2000 | 4.2 | NA | NA |
| Continuing Current Component C | 400 | | | 0.4 | | |
| Restrike Component D[2,4] | 109405 | 22708 | 1294530 | 100 | 140 | $2.5 \times 10^5$ |
| Multiple Stroke Component D/2[2] | 54703 | 22708 | 1294530 | 50 | 70 | $6.2 \times 10^4$ |
| Multiple Burst Component H[2] | 10572 | 187191 | 19105100 | 10 | 200 | NA |

[1] This component represents a very severe stroke, exceeding approximately 99.5 percent of all recorded strokes.

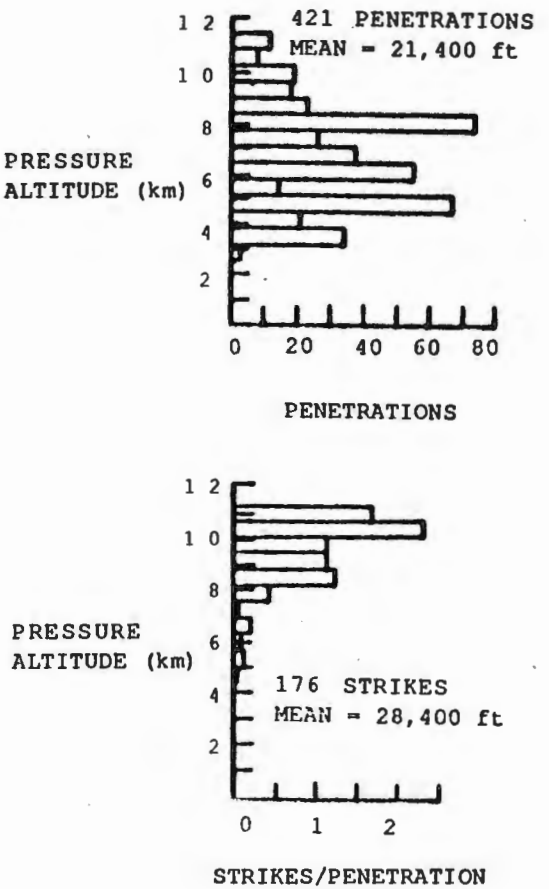
[2] The double exponential form is convenient for low level tests and analytical procedures.

[3] Rate of rise represents 90 to 95 percent severity level.

[4] Rate of rise is the same as that of Component A to recognize the fact that recorded rise times of subsequent strokes are often as short or shorter than those of the first return stroke.




Ref: Fischer & Plumer, 1977.




Ref: Fischer, et al., 1984.

FIGURE 2.3-1. AIRCRAFT LIGHTNING STRIKE INCIDENTS AS A FUNCTION OF ALTITUDE

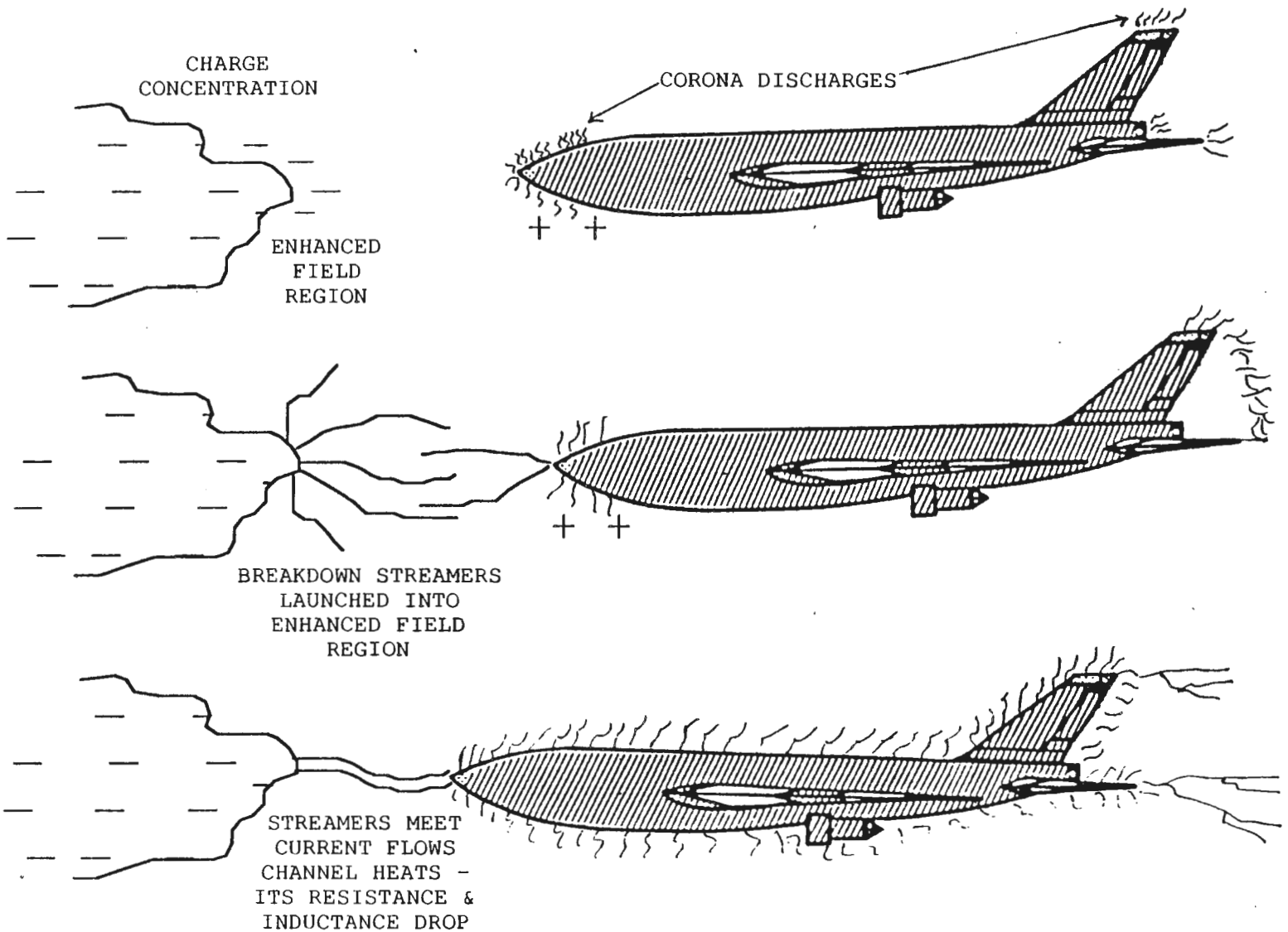



FIGURE 2.3-2. INTERACTION OF AIRCRAFT WITH CHARGE CENTERS

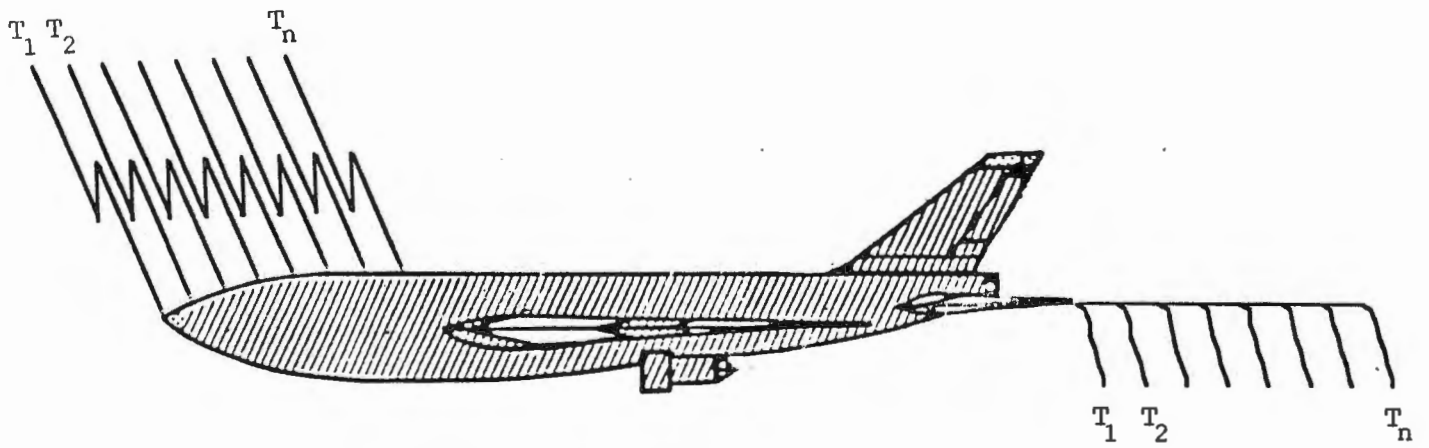



FIGURE 2 3-3. TYPICAL PATH OF SWEPT FLASH ATTACHMENT POINTS

- Zone 3: The remainder of the vehicle that is not covered by Zone 1 and Zone 2. There is a low probability of any attachment of the direct lightning flash arc. Zone 3 areas may carry large amounts of electric current, but only as a result of conduction between pairs of direct or swept stroke attachment points.

The SAE-AE4L committee has established the current levels to be expected in each of the strike zones to be used for test or certification purposes. An idealized representation of these defined current components for a complete flash is shown in figures 2.3-4 and 2.3-5.

The location of strike zones is well established for current aircraft configurations, see figures 2.3-6 and 2.3-7, but if configurations differ much from today's aircraft, locating the lightning strike zones may require careful interpretation of the particular geometry to determine the location and extent of the areas where lightning may attach, sweep, and hang-on.

A tabulation of lightning strike statistics is given in table 2.3-1 for major sections of commercial aircraft. In this grouping, the four sections most often struck by lightning are the nose, fuselage, wing, and horizontal stabilizer. The wide body jets also had high strike rates to antennas.

### 2.3.2. Structural Materials Effect on Strike Zones

A study by Grumman Aircraft Company investigated the strike attachment zones for an advanced fighter configuration (Craft, 1981). A scale model of the aircraft with replaceable panels for portions of the skin was used. The panels were made from aluminum, G/E, and kevlar. The study concluded that there is no essential difference between the strike zones for panels of G/E and aluminum. However, there is considerable difference for kevlar panels. Kevlar is a non-conductor; therefore, a lightning arc will jump over or along the surface rather than attach to kevlar.

Other studies have shown that the lightning arc will punch through kevlar to reach metal portions under the kevlar skin panels. This is also reported to be the case for radomes used to cover nose-mounted radar equipment. Lightning will readily puncture a radome to reach metallic portions of an all-metal aircraft. These items are described in a report by Fisher and Plumer.

## 2.4. Lightning Effects on Aircraft Systems

### 2.4.1. Range of Lightning Effects

Direct effects result from direct action of the lightning arc in the form of arcing and sparking on the aircraft; therefore, the term "direct effects." Evidence of this is seen as gross damage from the action of the arc on the materials. Protection against direct effects must come from the structure or protective diverters which prevent the currents from flowing into the sensitive areas. Important lightning parameters, which determine damage due to direct effects, are the dwell time on the aircraft, the peak current, the duration of

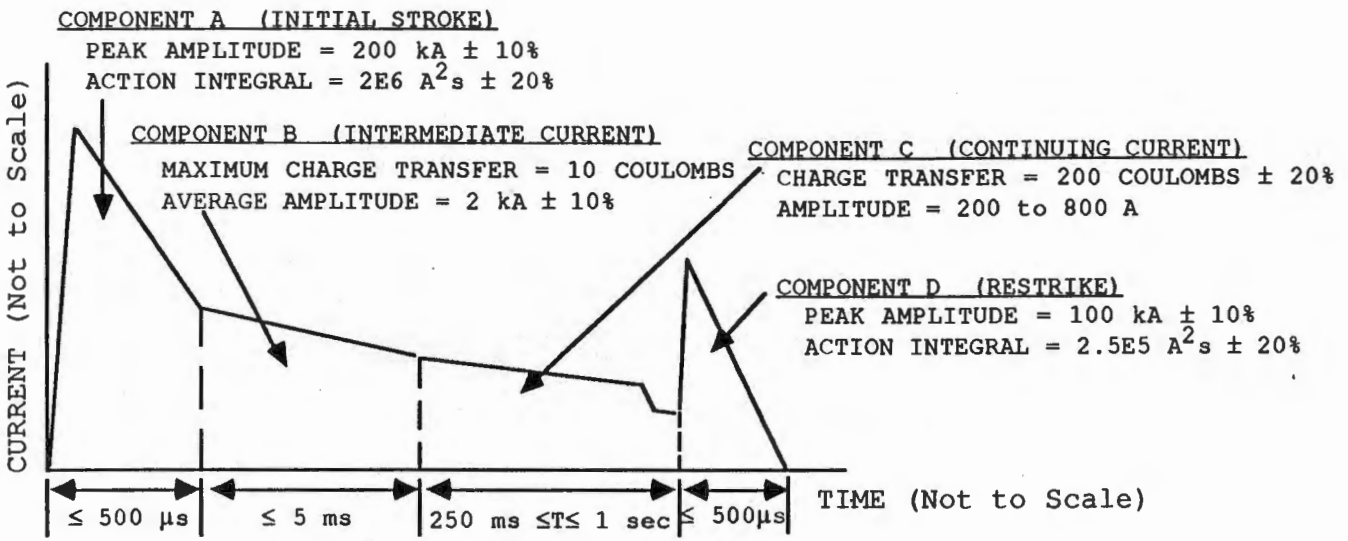



FIGURE 2 3.4. LIGHTNING CURRENT COMPONENTS EXPECTED IN EACH ZONE

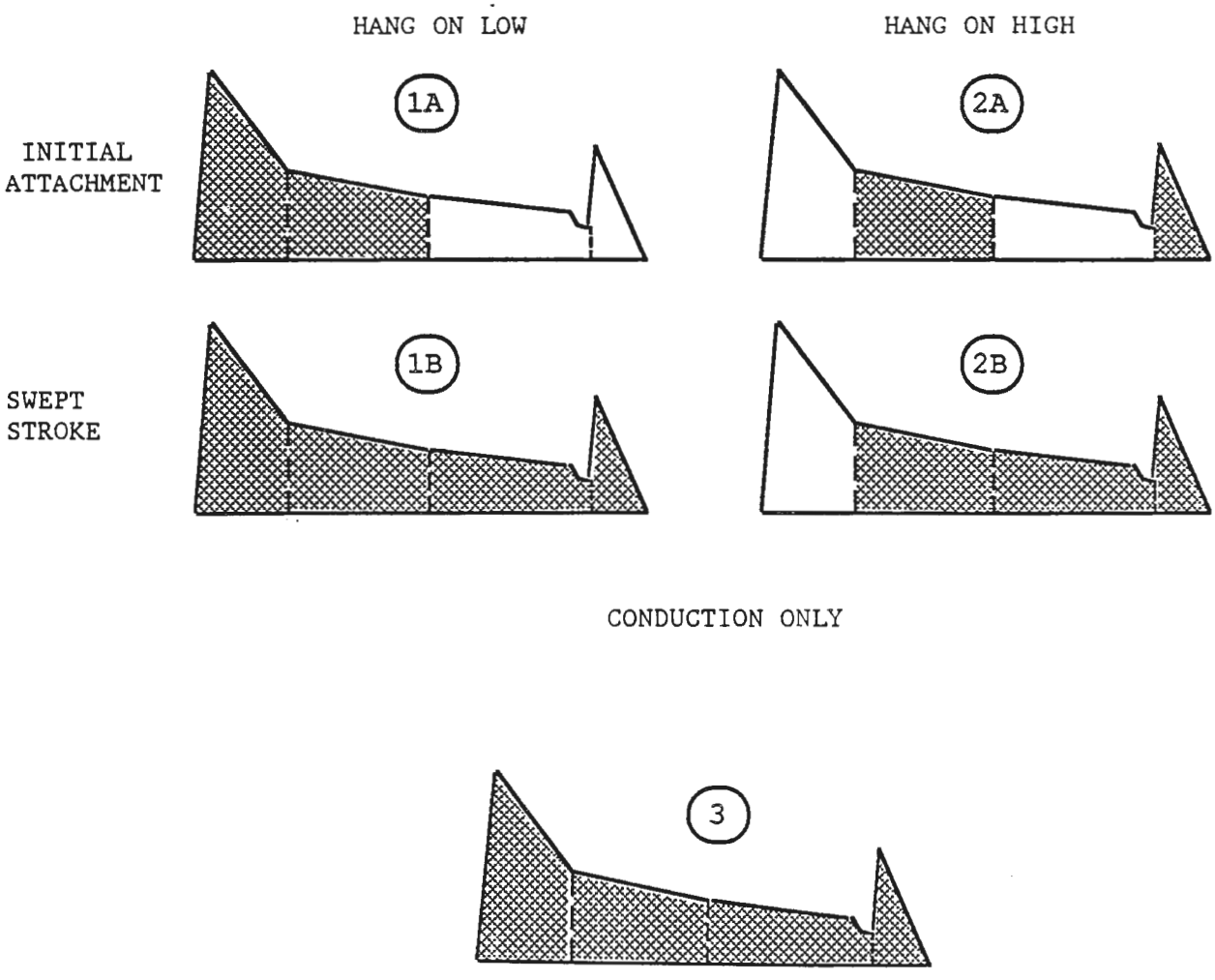



FIGURE 2.3-5. LIGHTNING CURRENT COMPONENTS EXPECTED IN EACH ZONE

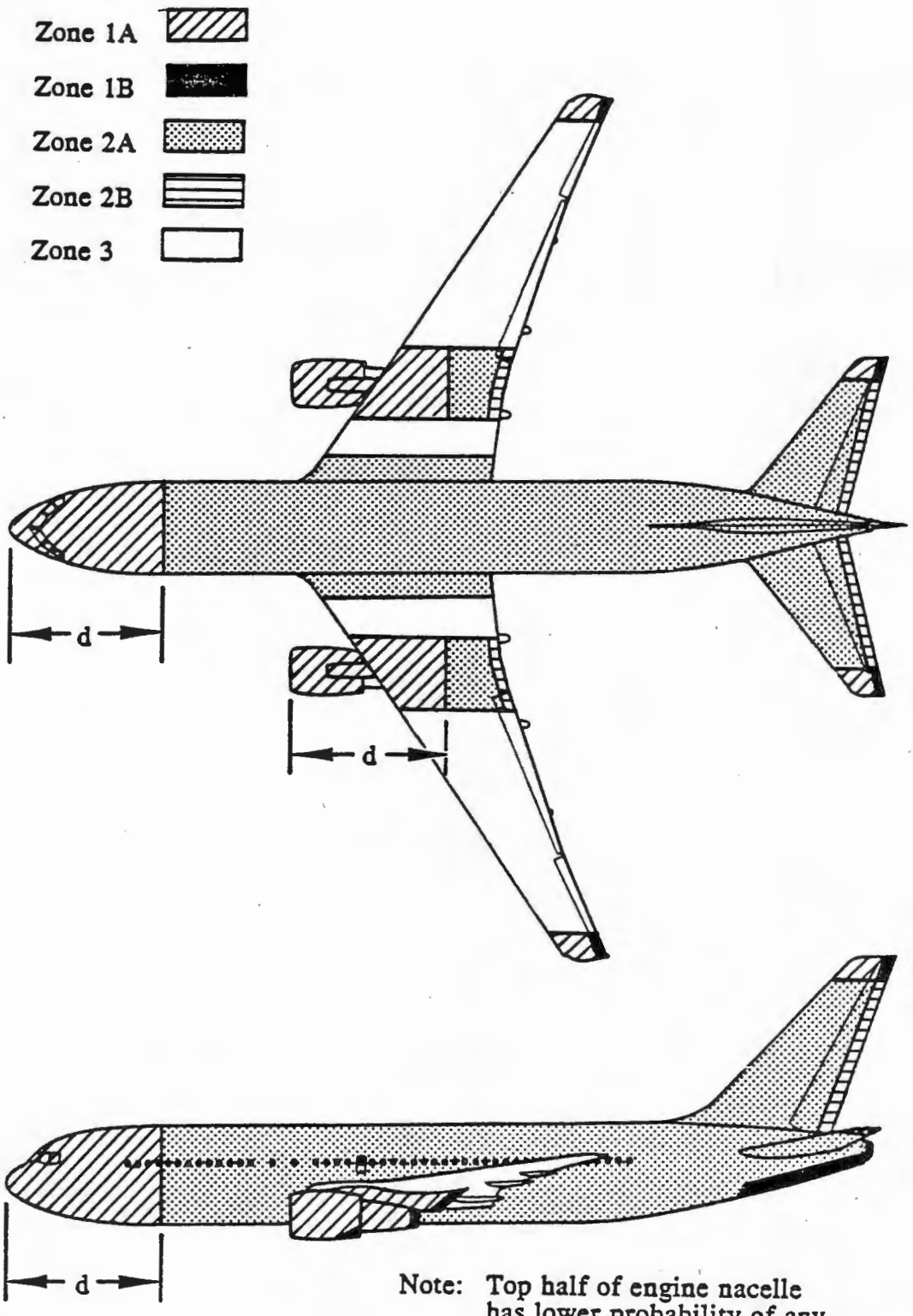



FIGURE 2.3-6. TYPICAL LIGHTNING STRIKE ZONES

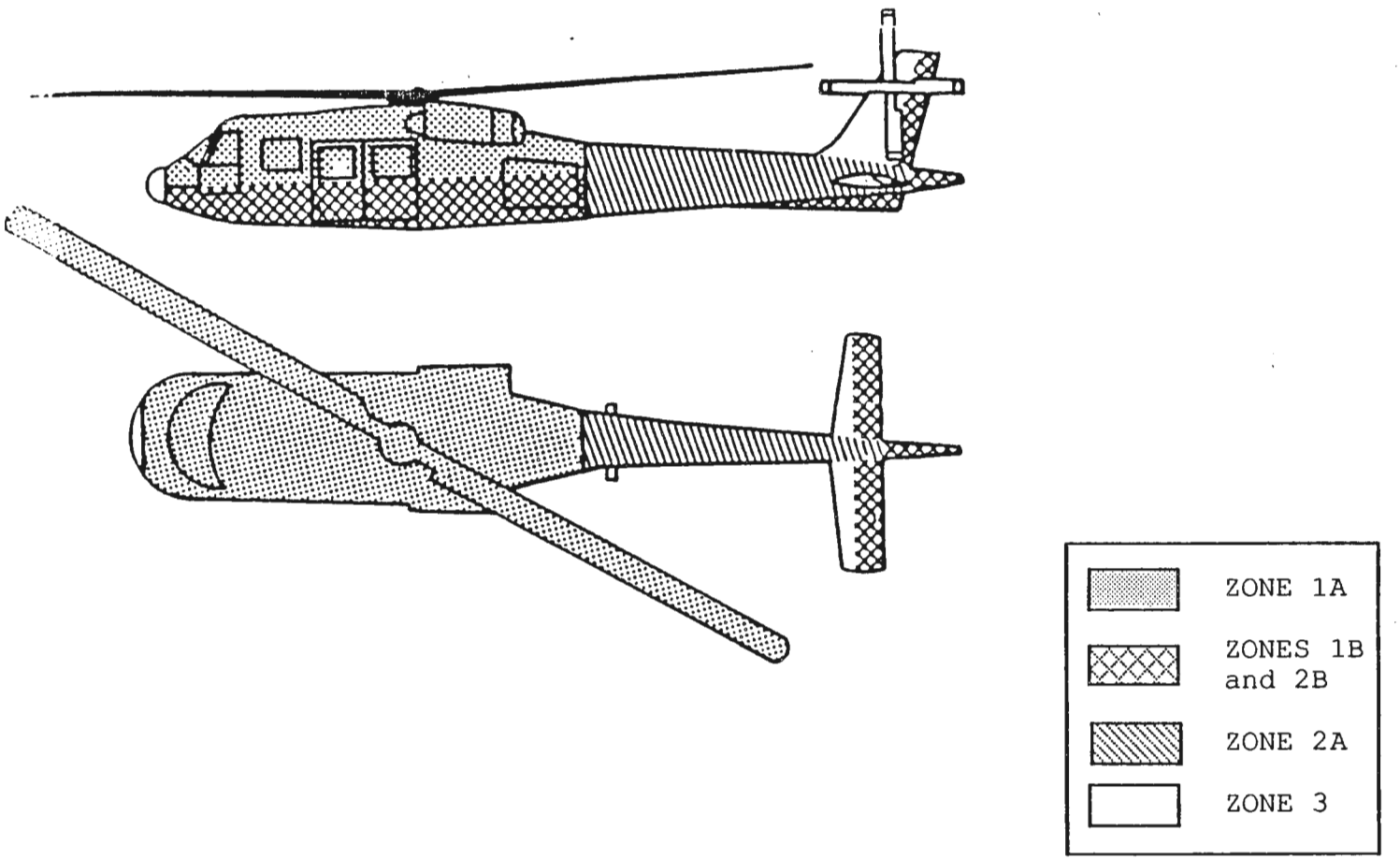



FIGURE 2.3-7. TYPICAL LIGHTNING STRIKE ZONES FOR HELICOPTERS
(User's Manual for AC-20-53, 1987)

TABLE 2.3-1. COMMERCIAL AIRPLANE LIGHTNING STRIKE STATISTICS

| | JET 2 ENGINE AFT MOUNTED T-TAIL | | JET 3 ENGINE AFT MOUNTED | | JET 4 ENGINE WING MOUNTED | | JET TOTAL NARROW BODY | | JET WIDE BODY | |
|---|---|---|---|---|---|---|---|---|---|---|
| | NO. | % | NO. | % | NO. | % | NO. | % | NO. | % |
| NOSE | 3 | 19 | 29 | 30 | 83 | 43 | 115 | 38 | 220 | 34 |
| WING | - | - | 24 | 24 | 33 | 17 | 57 | 19 | 99 | 15 |
| HORIZONTAL STABILIZER | 6 | 38 | 6 | 6 | 24 | 13 | 36 | 12 | 59* | 9* |
| VERTICAL STABILIZER | - | - | 14 | 14 | 12 | 6 | 26 | 8 | | |
| FUSELAGE | 5 | 31 | 20 | 20 | 38 | 20 | 63 | 21 | 112 | 17 |
| ANTENNAS | 2 | 13 | 1 | 1 | 2 | 1 | 5 | 2 | 100 | 15 |
| ENGINE | - | - | 1 | 1 | - | - | 1 | < 1 | 24 | 4 |
| TAIL | - | - | 3 | 3 | - | - | 3 | 1 | 35 | 5 |
| TOTAL | 16 | | 98 | | 192 | | 306 | | 649 | |

Ref: Cooley, Geren, and Melander; Sept. 1982

* HORIZONTAL/VERTICAL COMBINED

current, the maximum current rate-of-rise, and the number of restrikes. Aircraft parameters, which deteremine damage due to direct effects, are skin type, skin thickness, and paint.

Damage may also occur from indirect effects that damage components or disrupt the software operations of electrical and electronic systems. Indirect effects are generally more subtle than direct effects, because the physical damage from indirect effects is not easily seen. Indirect effects are caused by the transient voltages and currents that are induced into the wiring as a result of the lightning arc and the associated EM fields on the aircraft structure. Important parameters are the peak current, maximum current rate-of-rise, maximum EM fields rate-of-rise, number of restrikes, and for non-metal aircraft the duration of current flow.

Protection design for lightning indirect effects has many factors. Parameters to be considered when protection beyond the baseline configuration is needed include protection provided by the structure, cost, weight, reliability, impact on vehicle performance, expected life cycle, and protection necessary against other EM threats.

Table 2.4-1 lists possible effects from direct or nearby lightning strikes to aircraft in flight.

2.4.2. Direct Effects on Non-Metallic Structures

The most important direct effect of lightning strikes is the possible arcing and sparking in fuel areas. Other important direct effects include pitting and burning in structural materials.

For reasons discussed in Cooley (1985), direct effects protection design requires more test and analysis for composite materials than for metallic materials. For example, it is usually assumed, in the design of lightning protection, that if an electrically conductive fastener is located in an attachment zone having a high probability of hang-on, lightning will attach to the fastener. Because of the currents in the strike zone definitions, the direct attachment to a fastener in zone 1A, 1B, or 2B represents a severe worst case for any lightning strike. Although direct attachment is not a severe problem in aluminum structures, it is presently not possible to develop a fastener capable of joining advanced composite materials and carrying the full lightning current without seriously weakening the fastener. Current composite aircraft designs depend upon not having fasteners in critical parts of zone 1 and 2. Where fasteners are used, a repair procedure may be necessary for lightning strike damage.

Painted metal surfaces have longer hang-on times for swept strokes than do unpainted metal surfaces. These longer dwell times will cause deeper burning, and additional metal is required for painted metal skins on wet-wing aircraft.

Composite materials have various resin binders that burn and affect the dwell time for swept strokes. Therefore, the thickness of composite materials required for structural integrity protection against lightning strikes will

TABLE 2.4-1. RANGE OF EFFECTS FROM ATMOSPHERIC ELECTRICITY INTERACTIONS WITH AIRCRAFT

| EFFECT | CAUSE | CRITICALITY |
|---|---|---|
| Flight Control Disruption | Flight control systems have low tolerance to electrical transients caused by indirect lightning or static electrification effects. Transients may simultaneously affect parallel redundant systems. | Minor to Catastrophic |
| Fuel Tank Fire or Failure | Fuel vapor ignition may be caused by static electricity or lightning direct effects on structure. Fuel gauging and flow management electrical/electronics may spark from indirect effects. | Minor to Catastrophic |
| Loss of Engine Power | Power loss can result from possible direct effects which cause thermal or acoustic shock at engine inlet, or indirect effects of electrical transients on engine controls. | Minor to Catastrophic |
| Radome, Canopy, & Windshield Damage | Damage can occur due to direct effects of lightning strikes and arc discharge may be caused by static electricity buildup. | Minor to Catastrophic |
| Instrumentation Problems - Communications, Navigation & Landing System Interference | Instrumentation systems are disrupted by indirect transient effects caused by static electricity buildup and nearby or attached lightning strikes. | Minor to Catastrophic |
| Structural Damage | Structure damage is a result of burns or arcing and sparking from direct effects of lightning attachment to aircraft. | Minor to Catastrophic |
| Physiological Effects on Crew | Flash blindness & distracting electrical shock may be caused by the direct effects of nearby or attached lightning strikes. | Minor to Catastrophic |
| Unscheduled Deployment of Landing Gear or Control Surfaces | Premature activation of actuators is caused by indirect effects of lightning or static electricity buildup in electrical/ electronic systems. | Serious to Catastrophic |

depend upon the material's makeup as well as the fibers. It is expected that different materials of the same physical strength will behave differently when struck by lightning.

### 2.4.3. Indirect Effects on Non-Metallic Structures

The most important adverse feature of graphite and metal matrix materials is their lower conductivity. The conductivity factor, compared with aluminum, is 10 times less for metal matrix and 1000 times less for G/E. Lower conductivity means higher voltage drops in these structures than for a metal structure. Thus the voltage drop in a length of structure will be increased for the same level of lightning current. Because of the higher voltage drop, higher levels of currents will flow in any metallic paths such as plumbing and wiring inside an aircraft structure made from non-metallic materials. Additionally, any external EM fields will be attenuated less by a composite structure. For direct and nearby strikes, the internal EM fields will be at higher levels than for an aluminum aircraft. The exact difference in internal EM fields depends on the number of joints and openings, thickness of material, frequency range of the external EM fields, and the shape of the structure. The Atmospheric Electricity Hazards Protection (AEHP) program studies on an advanced composite (G/E) structure F-16 fuselage showed that the shielding was 15 dB for lightning pulses. It is estimated that a similar aluminum structure would have provided 30 to 40 dB of shielding, the limitation being primarily due to the number and size of openings.

In structures made from non-conductors such as kevlar and fiberglass, there is no EM shielding. This is also true for materials having very few conducting wires embedded into a non-conducting material such as Thorstrand kevlar. These materials can provide some protection against direct effects. There will be little or no improvement against indirect effects, because the magnetic shielding effectiveness of low loss factor metals, such as aluminum, is primarily due to currents flowing in the material. These currents generate EM fields which tend to cancel the incident fields. Electric field shielding is primarily due to an equipotential plane established by high conductivity metals. There are too few wires and too little metal in a material like Thorstrand to provide an effective shield against EM fields from nearby lightning sources or RF sources, such as airport radio and radar transmitters.

### 2.4.4. Effects on Aircraft Systems

Six examples of onboard system functions affected as a result of lightning and static electrification of aircraft are described below.

- The failure of engine instruments was experienced, as a result of lightning strikes, on an equipment version installed on a jet transport aircraft early in the development phase. The N1 tachometer and N2 were occasionally affected during lightning strikes. Electronic components in the engine tachometer transmitter were damaged. These components were connected to unshielded twisted pairs in a bundle of engine related wiring. The situation was repeated in laboratory tests and traced to the need for 10 dB additional shielding for the wiring. The solution to this problem was to add the necessary shielding by applying aluminum foil to the upper half

of the inside surface of the kevlar leading edge. This solution resulted in lighter weight than shielding the wiring.

- Possible sparks between fuel tank wiring and mechanical structures are a serious concern. For example, fuel plumbing can carry substantial currents in the event of a strike to the overflow vent tube. The fittings and plumbing must be capable of carrying these currents without sparking. Fuel gauge electrical wiring must be protected from currents on the structure path between lightning attachment and exit points along the leading and trailing edge. Currents from the cable shield to structure must not spark within the fuel tank or the fuel gauge.

- The pitot boom is often the point of attachment for lightning because of its shape and location near the front of an aircraft. The very high currents near the initial attachment can burn out the pitot heater and possibly damage electronic equipment inside the aircraft. This is particularly severe for pitot booms on aircraft with nose radomes.

- Hydraulic plumbing for flight control actuators needs to be grounded and bonded to the structure to remove high currents from the tubing fittings. These fittings have been damaged by the currents flowing in the structure and tubing from lightning strikes to the empennage. Damage was evidenced by loss of fluid in hydraulic systems.

- Static electricity results from triboelectric charging under normal flight conditions, from vehicle charging caused by nearby leaders, or as part of the initial attachment phase. High electric field breakdown discharges the accumulated charge in a repeated cycle that results in a series of sparkovers and therefore a repeating transient pulse. The electrical waveforms have wide band energy up to and including the Very High Frequency (VHF) band. Effects of static discharges appear as excessive radio noise and loss of sensitivity in the navigation and radio systems. The high electric fields have the potential to damage windscreens and damage electronic components in the window heater control.

- Lightning induced transient currents in electrical cables can cause errors in the digital data. Errors in digital bit streams are characterized by single bits or a burst of errors depending upon the baud rates. These errors are no different than those that should be expected due to other common EMI sources such as power transients, power faults, and RF and radar transmitter modulation frequencies. Digital systems designers need to consider possible disruptions in data streams similar to those that occur during long distance transmission over noisy lines and to include error detection and correction for flight critical data transmissions.

  In addition to bit stream errors, errors in digital data can occur in the discrete data signals. Discrete data input signals can be modified from true, "1", to false, "0", by transient currents, as for example, in the sensing of flaps up, wheels down, or other aircraft configuration sensors which operate as a high/low switch position. Discrete signals should not be directly used by onboard digital systems without some checks for reasonableness or low pass filtering or both. Mode change commands should

be verified over five to ten samples. Discrete data outputs to indicators and actuators may also be subject to upset or damage. Faulty indicators increase crew workload and may contribute wrong information leading to hazardous flight conditions. Errors in these discrete signals are serious because they lead to wrong information regarding the aircraft configuration. Early raising of the flaps on takeoff or lowering the landing gear at cruise altitude are two obvious examples.

# 3. INDIRECT EFFECTS PROTECTION - DESIGN APPROACH

Lightning protection for aircraft should be addressed during the design phase of the vehicle. This allows optimization of the lightning protection design and possible integration with protection designs for other EM threats. A design margin provides the definition of protection needed. This section discusses design considerations for grounding and bonding, installation of equipment, and routing of cables.

## 3.1. Design Margin

Adequate protection of equipment or systems is defined by a design margin. The design margin represents the difference between the transients induced at the interface level (by a worst case lightning strike) and the transients that the equipment is designed to withstand. This should be a positive number. A negative design margin indicates that some redesign is necessary. The size of the design margin is inversely proportional to the confidence which is given to the tests and/or analysis results used for verification.

The first step in defining a margin is to determine a threshold where the equipment becomes damaged so that it can no longer perform its intended function. Both analytical and experimental techniques are available for assessing the damage threshold. The techniques might be used during equipment design where the thresholds are specified, or during overall vehicle assessment where thresholds are used to establish Equipment Transient Design Levels (ETDL).

The equipment transient susceptibility level is another name for the damage threshold. This level will normally be somewhat higher than the ETDL, which represents the amplitude of voltage and/or current that the equipment is required to withstand or tolerate and remain operational. The ETDL, in turn, will be set higher than the maximum amplitude of transients that are allowed to occur on interconnecting wiring, the Transient Control Level (TCL). The relationship between transient control, equipment transient design, and susceptibility levels is illustrated in figure 3.1-1.

Standardized definitions of induced voltage and current waveforms and amplitudes that are representative of transients which appear in interconnecting wiring are found in AC 20-53. ETDLs may be selected from among the amplitude levels presented; TCLs can be derived by providing an adequate margin. Normally the transient control and design levels will be established by the airframe manufacturer or system integrator, who will compare the penalties of vehicle or interconnecting wiring protection with those of equipment hardening to establish the most efficient levels.

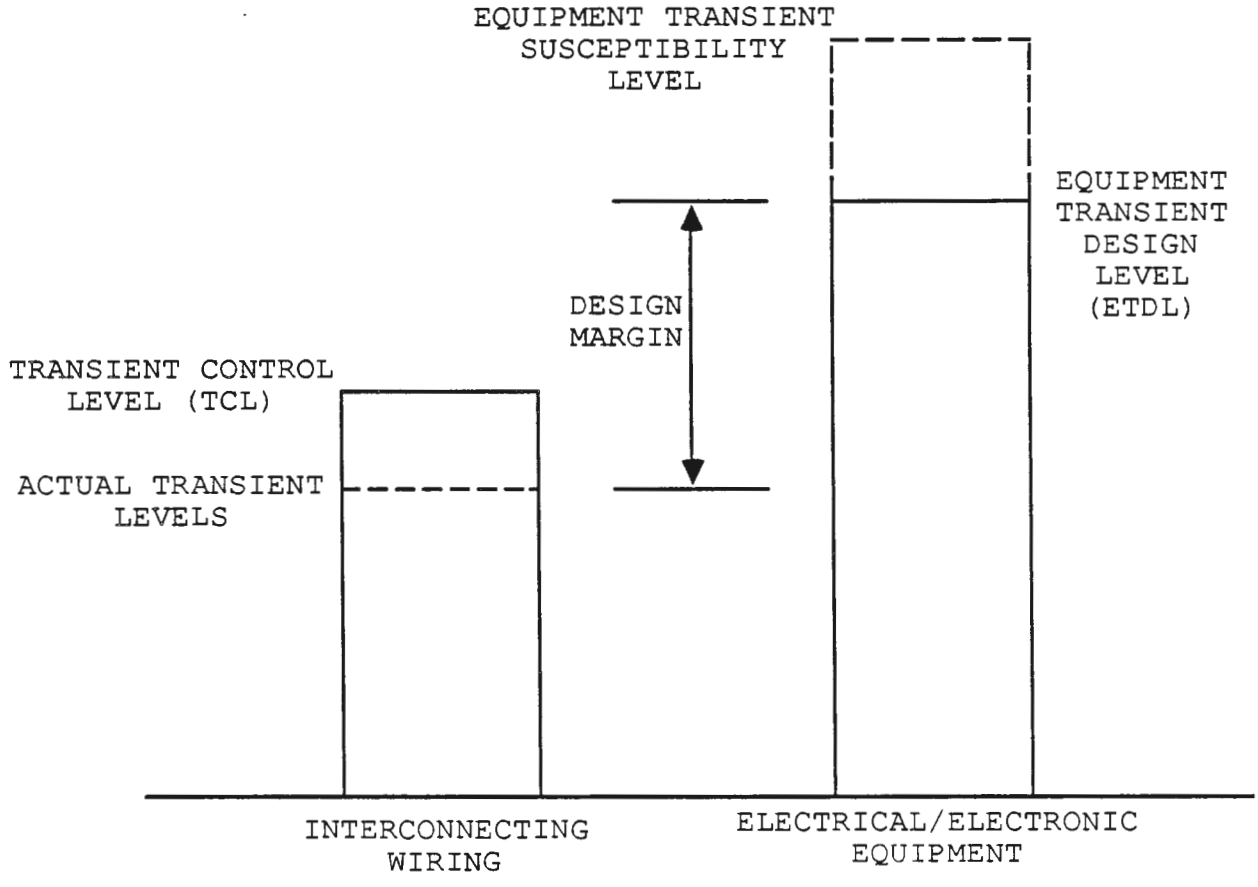



FIGURE 3.1-1. DESIGN MARGIN DEFINITION

## 3.2. Requirements for Grounding and Bonding

Grounding and bonding of the electronic equipment in an aircraft presently has a requirement for 2.5 milliohms from chassis to chassis (MIL-B-5087B, 1970). The resistance requirement was established early to assure that potential interference sources in different equipment would be isolated from sensitive electronics. The basis for the number chosen was one of engineering judgement. If the resistance was zero, there would be no interference or hazard to personnel due to power system faults or lightning. It is particularly convenient to use the 2.5 milliohm value for assessment of system protection against lightning. If 200,000 amps flows in a 2.5 milliohm resistor, the maximum voltage drop will be 500 volts. If every bond has less than 2.5 milliohms resistance, then no end-to-end validation testing is necessary, providing each unit of equipment will withstand 500 volts.

For the most part, aluminum and titanium alloys present no major difficulties in meeting the 2.5 milliohm specification. Maintaining adequate conductivity throughout an airframe has been a problem only when good contact between mating parts could not be maintained. Breaks in conduction paths are necessary at doors, hinges, and control surfaces. Corrosion control often requires a nonconductive coating on parts. Where interruptions in the grounding path occur, bonding requirements are met by reestablishing the conduction path using a metal jumper or strap. Utilizing present metal aircraft design approaches for composite craft will not yield the 2.5 milliohm resistance required.

Requirements for direct effects protection require that the grounding and bonding system be spark free in fuel areas, and that load bearing elements are not seriously degraded upon experiencing a lightning strike. Meeting these requirements requires an experimental approach.

Requirements for indirect effects protection of electrical/electronic systems against lightning strikes depend upon meeting the operational requirements of the equipment. The need for adequate grounding and bonding is well established by operational experience. For example, aircraft having poor grounding and bonding (i.e., poor or no electrical conduction throughout the structure and wiring) may be subject to operational hazards varying from major to minor. When different parts of the aircraft and wiring are not electrically connected, differences in potential can build up between them. Since the ultimate breakdown potential of air is lowered with altitude, corona or sparks may occur. Furthermore, fuses or circuit breakers to protect against shorts in equipment and wiring may not open properly. These factors lead to the following:

- Fuel explosion hazards from sparking in fuel gauges and wiring.
- Static discharges that may rupture windows (dielectric).
- Personnel electric shock hazards.
- Electronic equipment damage - burn out.
- Electrical/electronic systems malfunctions.

- Excessive radio noise when communicating with airport or other aircraft.

The electrical function of a bonding and grounding system for protection of electronic and electrical equipment is to provide a low impedance ground path for current flow. Table 3.2-1 indicates the current levels that may be expected from various sources.

The path impedance should be low enough that these interference sources will not produce a high voltage value that can disrupt equipment functions. Electrical/electronic equipment can be made to tolerate voltage differences of a few volts between units. Typical values for equipment tolerance against transients are shown in table 3.2-2.

Tolerable levels of ground impedance, based upon values in tables 3.2-1 and 3.2-2, are shown in table 3.2-3. This data indicates that the range of acceptable ground impedance values depends upon the source of interference and the degree of protection built into the installation and equipment. Equipment containing large scale integrated circuits can be designed and built to withstand 600 volt potential differences. This level of protection requires balanced circuits, high impedance input networks, and twisted pair wiring. More typical circuitry can withstand 10 volts between units by using balanced circuits. Single ended digital circuits can only tolerate a few volts between units for TransistorTransistor Logic (TTL) circuits that operate with 5 volt logic levels.

## 3.3. Guidelines for Cable Routing and Equipment Installation

The aircraft structure has a significant influence on the lightning protection measures needed for a specific design. The structure provides the first layer of protection from lightning. If the structure were a solid shell and a perfect conductor, no other protection measures would be necessary. In reality, there are apertures, joints, and imperfect conductors. These lead to leakage of fields and currents penetrating to the interior of the aircraft. It becomes necessary to select ways to minimize coupling to internal wiring.

Protection may be provided by increased shielding on exterior cables, good bonding at apertures, selective cable routing, surge arrestors, and wiring configuration between circuits.

Proper ways of penetrating the structure or equipment are illustrated in figure 3.3-1. These techniques maintain the protection integrity of the outer structure rather than allowing currents to penetrate the vehicle interior.

As a general rule, wiring should be routed near a good ground or reference plane and away from joints and apertures. A ground plane provides a common low impedance reference for equipment and also provides a fault return for power systems. Multiple connections between the ground plane and the structure allow division of the lightning current between the metal ground and structure. This is essential for composite structures.

TABLE 3.2-1. GROUND SYSTEM CURRENTS FROM A VARIETY OF EM SOURCES

| CURRENT SOURCE | RANGE OF CURRENT |
|---|---|
| 1. Lightning currents, broad band pulses. | 20 KA to 200 kA |
| 2. Power system fault currents, continuous wave 400 Hz. | 100 A to 1000 A |
| 3. RF current returns from on board transmitters and transponders, continuous wave HF, VHF, UHF, and microwave. | 1 A to 10 A |
| 4. Currents from nearby sources of RF energy such as radar and radio transmitters, continuous wave HF, VHF, UHF, and microwave. | 1 A to 10 A |
| 5. Power currents resulting from relays, strobe lights, and other high pulse current devices. | 5 A to 50 A |
| 6. Digital and analog signals between units of equipment, CW and Pulses 100 Hz to 10 MHz. | 10 ma to 5 A |

TABLE 3.2-2. TYPICAL INTERFERENCE TOLERANCE LEVELS BETWEEN UNITS OF ELECTRONIC EQUIPMENT

| | |
|---|---|
| 1. Lightning Pulses (a) | 600 volts |
| 2. Power System Faults (b) | 10 volts |
| 3. Continuous Waves HF Frequencies | 50 volts |
| 4. Continuous Waves VHF Frequencies | 100 volts |
| 5. Continuous Waves UHF Frequencies | 100 volts |
| 6. Tracking Radar (Sweep Frequency) | 10 volts |

NOTES: (a) Specially protected against lightning by balanced signals, high impedance, and twisted pair circuits.
(b) Circuits unprotected except balanced signals, twisted pair wiring.

TABLE 3.2-3. TOLERABLE LEVELS FOR GROUND IMPEDANCE IMPLIED BY TABLE 3.2-1 AND TABLE 3.2-2 VALUES

| | |
|---|---|
| 1. Lightning Pulses | 3 to 30 milliohms |
| 2. Power System Faults 400 Hz | 10 to 100 milliohms |
| 3. Continuous Waves HF | 5 to 50 ohms |
| 4. Continuous Waves VHF | 10 to 100 ohms |
| 5. Continuous Waves UHF | 10 to 100 ohms |
| 6. Tracking Radar (Sweep Frequency) | 0.1 to 1 ohm |
| 7. Power Currents From Onboard Pulsed Current Devices | 20 to 200 milliohms |
| 8. Digital and Analog Signals | 0.2 to 100 ohms |

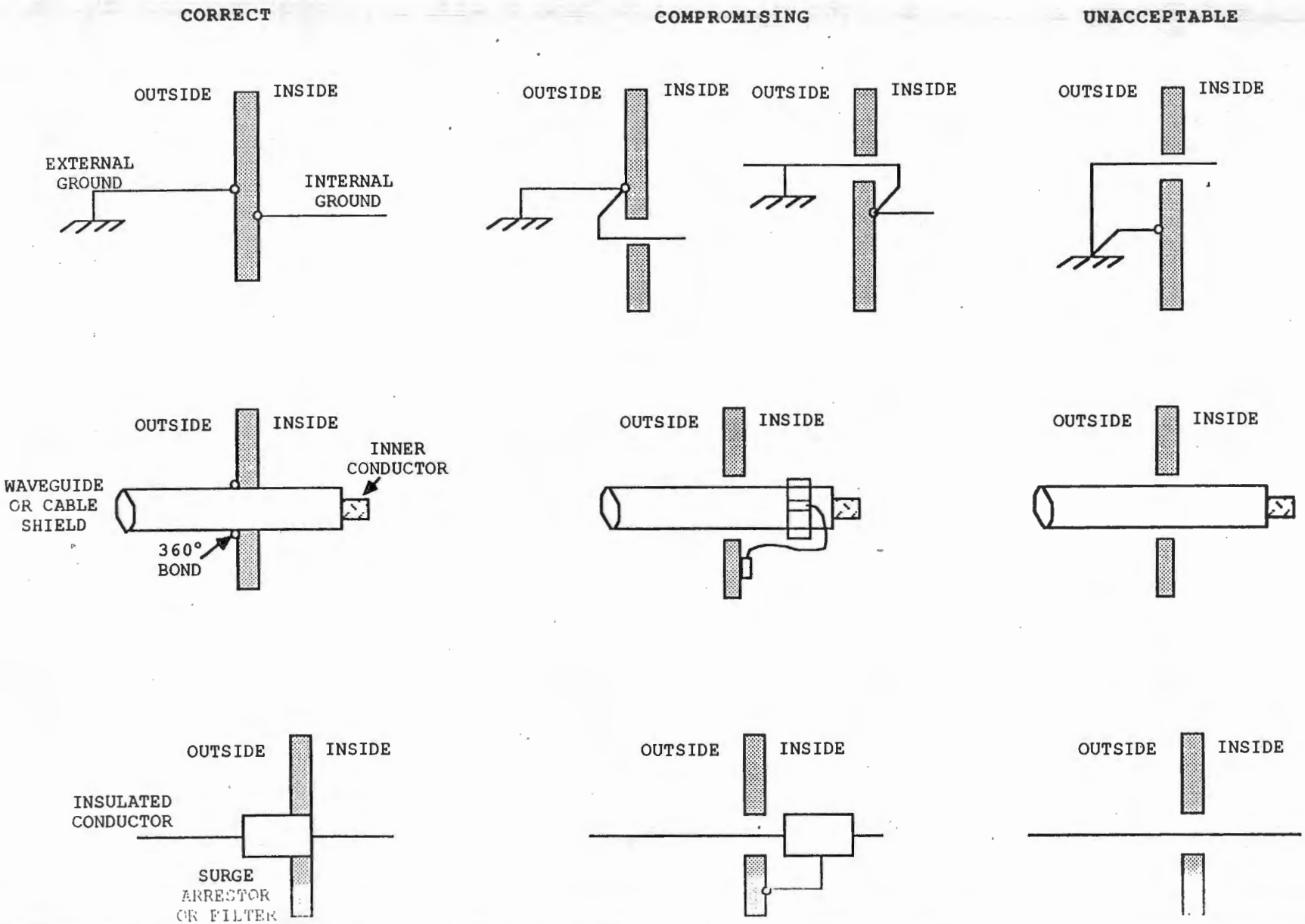



FIGURE 3.3-1. METHODS OF PENETRATING THE STRUCTURE WITH VARIOUS CONDUCTORS

Wiring is subject to differential mode and common mode coupling. Usage of twisted pair reduces the differential mode coupling. Shields tied to ground at each end of the cable significantly reduce the common mode coupling.

Figure 3.3-2 shows the coupling characteristics associated with various wire types and configurations. The wiring is assumed to be internal but routed near an aperture or non-metal structure.

Cable routing may not reduce transients from lightning indirect effects sufficiently to protect circuitry. Terminal protection devices to provide additional shielding can be placed at a vehicle penetration or within a circuit. Devices are typically in-line modules and may include limiters, filters, attenuators, buffers, and RF control devices. Each has different characteristics and must be evaluated for inclusion in specific designs. Important parameters, which should be evaluated for each device, are low shunt impedance, noise floor, insertion loss, insertion phase, and clamping capabilities at low and high levels.

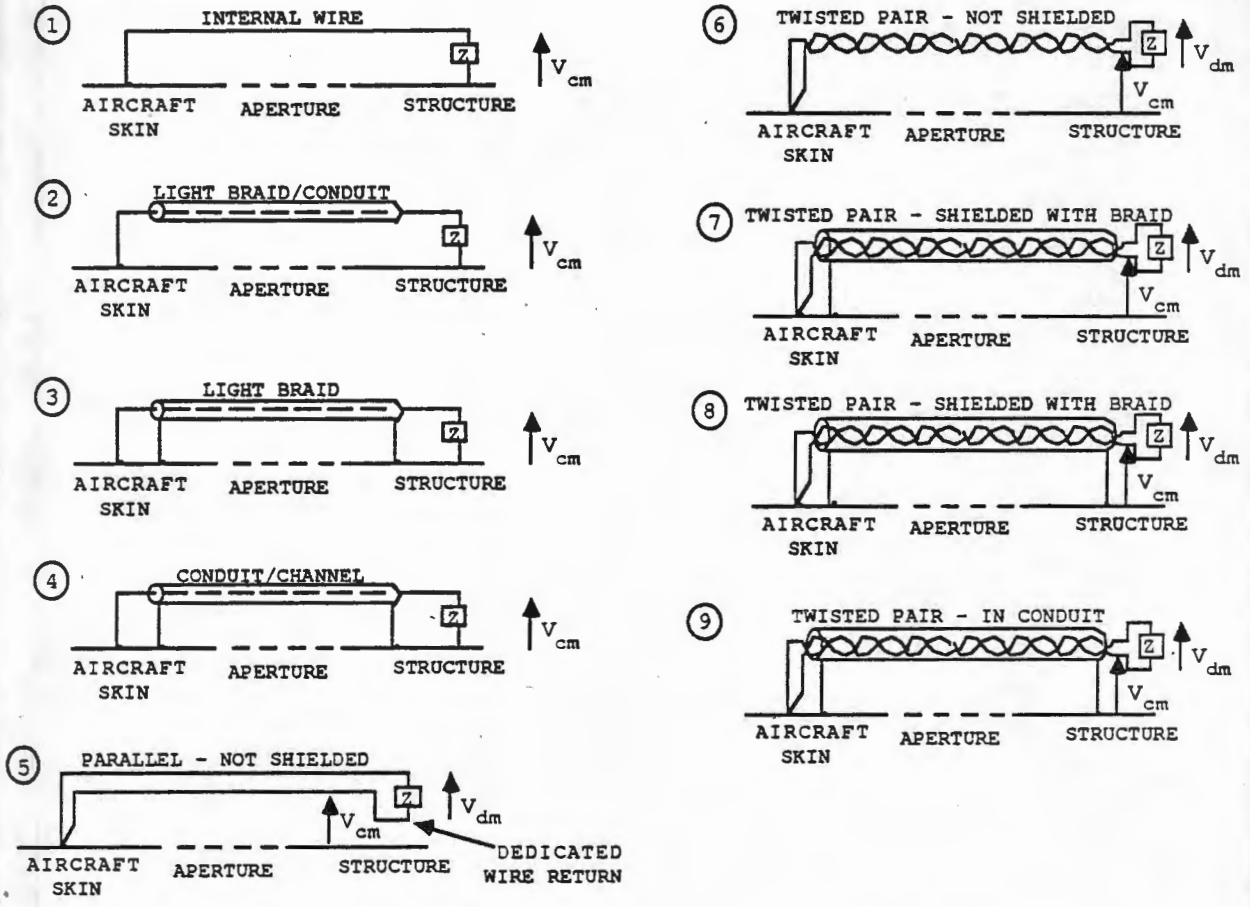
AEHP has defined the following guidelines for equipment installation to maximize the inherent shielding provided by the aircraft structure (Orange Book, 1987).

- Orient equipment where coupling with the EM fields is minimized. Avoid locations where fields are the highest (e.g., Zone 1 locations).
- Locate equipment as far from apertures and joints as possible.
- Cluster equipment in areas where the lowest flux density is expected, i.e., away from locations with small radii of curvature.
- Locate equipment in compartments with shielding, if possible.
- Where equipment must be located near apertures such as equipment bay hatches, the bay should be shielded and EMI gaskets used to improve bonding of the hatch to the compartment.
- Equipment in the canopy area, as well as the wiring leading to and from it, should be shielded. Equipment should be kept below the canopy sill, if possible.

A typical requirement for equipment protection is that both upset and damage protection be provided. This protection can be provided by either hardware or software.

The following AEHP guidelines apply to wire placement:

- Route the wiring in areas where lowest flux intensity is expected:
  - Within conductive structural enclosures.
  - Where minimum aperture coupling is predicted.
  - As close as possible to highly conductive structural members.




| MAGNETIC FIELD INDUCTIVE/RESISTIVE VOLTAGES | | ELECTRIC FIELD CAPACITIVE VOLTAGES | |
|---|---|---|---|
| (1) | FULL COMMON MODE VOLTAGE | FULL COMMON MODE VOLTAGE | |
| (2) | FULL COMMON MODE VOLTAGE | > 40 dB ATTENUATION | |
| (3) | 20 TO 35 dB ATTENUATION | > 40 dB ATTENUATION | |
| (4) | > 60 dB ATTENUATION | > 60 dB ATTENUATION | |
| COMMON MODE | DIFFERENTIAL MODE | COMMON MODE | DIFFERENTIAL MODE |
| (5) FULL VOLTAGE | 20-30 dB LESS THAN Vcm | FULL VOLTAGE | NOT EASILY SPECIFIED, BUT 30-40 dB LESS THAN Vcm |
| (6) FULL VOLTAGE | 50-55 dB LESS THAN Vcm | FULL VOLTAGE | |
| (7) FULL VOLTAGE | 55-60 dB LESS THAN Vcm | > 40 dB ATTENUATION | |
| (8) 20 TO 30 dB ATTENUATION | 55-60 dB LESS THAN Vcm | > 40 dB ATTENUATION | NEGLIGIBLY SMALL |
| (9) > 60 dB ATTENUATION | > 60 dB LESS THAN Vcm | > 60 dB ATTENUATION | |

FIGURE 3.3-2. COMMON AND DIFFERENTIAL-MODE COUPLING CHARACTERISTICS FOR STRUCTURE RETURN AND TWO-WIRE CIRCUITS

- Close to a skin that can be considered a good shield, such as aluminum with controlled joints.

- Away from conducting structures with small radii of curvature..

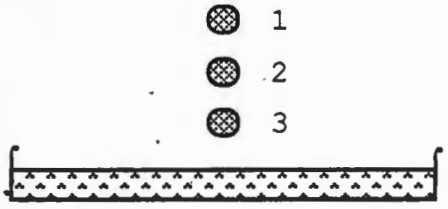
- Orient cables perpendicular to primary current path and joints to minimize coupling.

- Separate wiring according to degrees of hardness and threat exposure (i.e., isolate wiring to equipment located in a lightning zone).

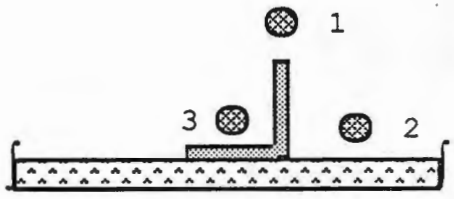
Figure 3.3-3 shows various locations for routing cables near structure. For all-metal structures, the numbers 1 through 4 indicate worst to best locations for cables. This is not the case with composite structures.

Coupling to these configurations was quantified using a two-dimensional method-of-moments code (Geren, Melander, and Hall, 1987) using typical thicknesses and conductivities for aircraft structures. Predictions tabulated in table 3.3-1 were made based on three structural materials. First, each configuration was assumed to be entirely made up of aluminum. Second, the skin was assumed to be G/E and primary structure was aluminum. The third configuration was identical to the second, except that kevlar, a non-conductor, replaced the G/E skin. These predictions were made with only the structure shown and assumed a 200,000 amp double exponential waveform input to the model. The values illustrate the differences in cable placement for metal and composite structures. See section 6.1 for an additional example of coupling to wiring in various locations.

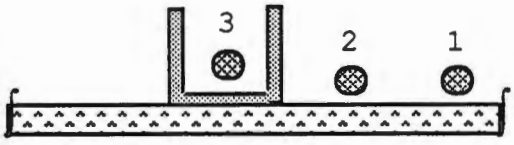
Peak current and voltage transient responses on wiring between aircraft systems, such as autopilot, engine control, fuel pump, fuel gauge, and electric power, due to a 200 kA direct lightning strike are tabulated in table 3.3-2. These values were obtained from results measured during a low-level swept CW lightning simulation test on a composite test bed representative of single engine general aviation aircraft (Cooley and Shortess, 1987). Wiring interconnections were representative of digital and analog signals, power, and grounds.



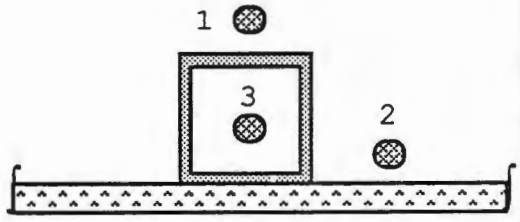

CONDUCTORS OVER A PLANE



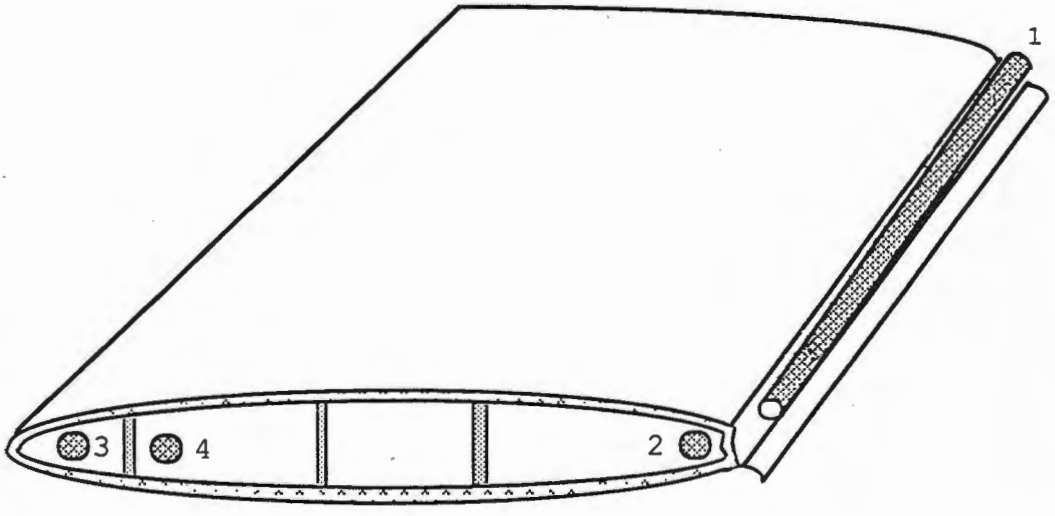

CONDUCTORS NEAR AN ANGLE




CONDUCTORS NEAR A CHANNEL




CONDUCTORS NEAR A BOX




CONDUCTIVE WING STRUCTURE

NOTE: The numbers 1 to 4 indicate worst to best cable locations.

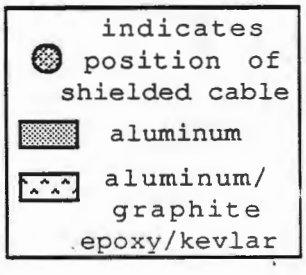



FIGURE 3.3-3. CABLE ROUTING NEAR CONDUCTIVE STRUCTURE

TABLE 3.3-1. ESTIMATED SHIELD CURRENTS DUE TO A DIRECT LIGHTNING STRIKE

| Aluminum Skin | | | Graphite Epoxy Skin | | | Kevlar Skin | | |
|---|---|---|---|---|---|---|---|---|
| Shield Location | Peak Current (kA) | Peak Time ($\mu$s) | Shield Location | Peak Current (kA) | Peak Time ($\mu$s) | Shield Location | Peak Current (kA) | Peak Time ($\mu$s) |
| **Shields over a plane** | | | | | | | | |
| 1 | 4.76 | 6 | 1 | 110 | 50 | 1 | 200 | 6 |
| 2 | 3.67 | 6 | 2 | 110 | 50 | 2 | 200 | 6 |
| 3 | 2.48 | 6 | 3 | 110 | 50 | 3 | 200 | 6 |
| **Shields near an angle** | | | | | | | | |
| 1 | 5.91 | 6 | 1 | 33.8 | 30 | 1 | 50.3 | 6 |
| 2 | 2.51 | 6 | 2 | 39.4 | 30 | 2 | 57 | 6 |
| 3 | 2.46 | 6 | 3 | 13.6 | 32 | 3 | 21.7 | 6 |
| **Shields near a channel** | | | | | | | | |
| 1 | 3.05 | 6 | 1 | 46.7 | 20 | 1 | 55.4 | 6 |
| 2 | 1.00 | 6 | 2 | 20.6 | 30 | 2 | 30.5 | 6 |
| 3 | .77 | 6 | 3 | 4.62 | 32 | 3 | 6.7 | 6 |
| **Shields near a box** | | | | | | | | |
| 1 | 4.32 | 6 | 1 | 17.9 | 26 | 1 | 26.2 | 6 |
| 2 | 2.14 | 6 | 2 | 35.6 | 24 | 2 | 46.4 | 6 |
| 3 | .04 | 104 | 3 | .88 | 124 | 3 | .88 | 108 |
| **Shields in a wing** | | | | | | | | |
| 1 | 10.5 | 6 | 1 | 35.9 | 30 | | | |
| 2 | .38 | 16 | 2 | 35.9 | 32 | | Not | |
| 3 | .02 | 112 | 3 | 3.4 | 28 | | Calculated | |
| 4 | .46 | 12 | 5 | 28.4 | 20 | | | |

TABLE 3.3-2. COUPLING RESPONSES TO SUBSYSTEM WIRING DUE TO A DIRECT LIGHTNING STRIKE

| System | Coaxial (RG-58) | Twisted Pair (no shield) | Twisted Pair (shielded) | Twisted Triple (no shield) | Single Wire |
|---|---|---|---|---|---|
| **Nose-to-Tail Attachment** | | | | | |
| Fuel<br>Electrical | 100-300 V<br>35 A | | | | |
| Fuel<br>Mechanical | | | 30-210 V<br>10 A | | |
| Engine<br>Control | 10-50 V<br>800 A | | 55 V | | |
| Lights<br>Power | | | | 40-1500 V<br>15-50 A | |
| Autopilot | | 50-1070 V | 50-230 V<br>1800-7000 A | | 540 V |
| Autopilot<br>shield disconnected | | 550-2400 V | 250-1600 V | | 1300 V |
| **Wing-to-Tail Attachment** | | | | | |
| Fuel<br>Electrical | 800-2600 V<br>20 A | | | | |
| Fuel<br>Mechanical | | | 90-1200 V<br>60 A | | |
| Engine<br>Control | < 10 V<br>140 A | | 150-400 V | | |
| Lights<br>Power | | | | 10-32,000 V<br>15-210 A | |
| Autopilot | | 1100-1070 V | 250-2600 V<br>15,000-60,000 A | | 1850 V |
| Autopilot<br>shield disconnected | | 1500-20,000 V | 700-11,000 V | | 2500 V |

# 4. INDIRECT EFFECTS PROTECTION - TEST METHODS

## 4.1. Review of Lightning Simulation Methods

This section presents a review of the lightning simulation technology used for evaluating the effects of lightning and static electrification on aircraft. The evolution of the simulation technology and considerations in selecting a simulation technique are discussed.

### 4.1.1. Lightning Simulation Technology

Several lightning simulation test techniques are currently being utilized for verification and validation of aircraft lightning protection. These different methods are the outgrowth of research using ground based simulations of lightning interactions with aircraft in flight. Under these test methods several aspects of natural lightning are simulated by pulse generators having various waveshapes and energy content as well as swept CW generators. The methods are often utilized, singly and in combinations, throughout an aircraft development cycle. These test methods, together with analysis procedures, characterize the indirect effects of lightning on aircraft electrical/electronic equipment.

The main reason for the different test methods is the difficulty of accurately generating natural lightning in the laboratory for large sized objects. It is very difficult (if not impossible) to accurately simulate severe lightning environments on an aircraft. This is because of the high levels of energy, current, and electric and magnetic fields associated with naturally occurring lightning. Simulation of these high energy levels requires very large and expensive facilities for an object the size of an airplane. Tests on aircraft and electrical/electronic equipment more commonly attempt to simulate portions of the lightning environment.

High level simulations of lightning appropriate for protection against direct effects, such as structural damage, are easier to implement because only small portions of the aircraft need to be exposed during a test. These simulations for direct effects of lightning on aircraft have been codified and are generally accepted as published in the report of the SAE Committee AE4L (1978).

Simulations for indirect effects, such as for the potential damage to flight control electronics from induced electrical transients, could require the use of the entire aircraft and several attachment locations for full evaluation. Thus, simulations for the indirect effects of lightning on aircraft and hardware have evolved along with the research efforts leading to a better understanding of these effects. It was recognized early on that electronic equipment would be affected by induced transients from lightning and other EM energy. However, the aircraft developments were allowed to go forward while answers were sought for protection against these effects. Research efforts were initiated in the

mid 1970's to develop the means of testing, assessing, and protecting the advanced electronics equipment. Since indirect effects testing necessarily involves the full vehicle and simulation of severe lightning levels on the full vehicle is impractical, approximate test methods were sought.

Several different investigators arrived at a variety of simulation techniques and test equipment because of different starting points and available resources. The data in table 4.1-1 summarizes the prior research and development tests on full scale aircraft. Major research and development efforts were concentrated in the AEHP program sponsored by the Air Force Flight Dynamics Laboratory (AFFDL) and supported by Tri-Services, FAA, NASA, and the Defense Nuclear Agency (DNA).

### 4.1.2. Considerations in Selecting Simulation Techniques

Many techniques and facilities are currently available for simulation of atmospheric electricity effects on aircraft. The selection of an appropriate method is challenging; it depends upon the ultimate use of the data and state of development of an aircraft.

Existing techniques for simulating static electrification, such as precipitation static, provide adequate data for aircraft design and protection. Nearby lightning strikes may be simulated using simulators developed for Electromagnetic Pulse (EMP) tests. The real challenge is to adequately simulate effects of lightning directly attached to an aircraft. A simulation technique that imposes all features of the lightning environment in a proper time sequence is desirable for full aircraft level tests. However, full vehicle tests are not time- and cost-effective for subsystem tests or to provide design data. It is especially important that the simulation technique provide data on the system, subsystem, component equipment, or Line Replaceable Unit (LRU). This data can be extrapolated to the values that occur when the aircraft is exposed to the real lightning environment.

Lightning simulation techniques must account for the significant features of natural lightning and the natural EM environments. These environments ultimately determine the indirect effects on advanced aircraft electronic equipment. Features necessary in the simulation setup include:

- The number of strokes in a lightning flash and the time between them.
- The peak current amplitude, rise time, and decay time reflecting parameters recently measured on natural lightning strikes.
- The proper EM environment geometric configuration around the test bed and the proper time phasing.
- The high voltage and electric fields associated with approaching leaders and static charging, preceding the high current and magnetic fields associated with the lightning return strokes.
- Lightning channel impedance and attachment.

TABLE 4.1-1. PRIOR RESEARCH AND DEVELOPMENT TESTS ON FULL-SCALE AIR VEHICLES

| TEST | DESCRIPTION | RESULTS |
|---|---|---|
| AFFDL F-16<br>Mockup; G/E<br>Composite<br>1983 | Direct Attached<br>- Coaxial<br>- CW Tests<br>- Moderate Threat<br>20 kA; 50 kA/$\mu$s | - V, I, B-Dot, D-Dot Data<br>- Detailed Model<br>- CW response within 4 dB of measurements at 28 kA |
| AFFDL ALCM<br>1983 | Direct Attached<br>- Coaxial<br>- Full Threat<br>200 kA; 200 kA/$\mu$s<br>- Moderate-Level 50 kA | - V, I Data<br>- EM Model<br>- Extrapolated values 7.7 dB greater than measurements at 200 kA |
| NASC F/A-18<br>1982 | Direct Attached<br>- Coaxial<br>- High-Level<br>100 kA; 100 kA/$\mu$s<br>- Low-level CW<br>760 A | - Functions<br>- V, I Data<br>- EM Model<br>- No Upset/Damage<br>- Predicted CW results 7.2 dB greater than measurements at 173 kA |
| NASC F-14<br>1982 | Direct Attached<br>- Coaxial<br>- High-Level<br>180 kA; 180 kA/$\mu$s | - Functions<br>- V, I Data<br>- EM Model<br>- Predictions |
| NASA F-106<br>Calibration<br>1982 | Direct Attached<br>- Shock-Excited<br>- Radiated | - V, I, B-dot, D-dot Data<br>- Spectral Analysis<br>- Simple Model |
| RAE UK JAGUAR<br>1982 | Direct Attached<br>- Damped Sine Wave<br>- Several Levels<br>100 kA; 20 kA/$\mu$s | - V, I, E, H Data<br>- Spectral Analysis<br>- External Currents/Fields |
| NSWC F-16<br>Mini-Test<br>1982 | Direct Attached<br>- Low-level<br>30 kA; 30 kA/$\mu$s | - Currents<br>- Spectral Analysis |

TABLE 4.1-1. PRIOR RESEARCH AND DEVELOPMENT TESTS ON FULL-SCALE AIR VEHICLES (continued)

| TEST | DESCRIPTION | RESULTS |
|---|---|---|
| AFFDL C-130<br>1981 | Direct Attached<br>- Radiated<br>- Low-level | - B-dot, D-dot Data<br>- Simple Models |
| AFFDL F-111<br>1978 | Direct Attached<br>- Low-level; 2.5 kA | - Voltages<br>- Simple Model |
| AFFDL YG-16<br>Composite; Fwd.<br>Fuselage; 1978 | Direct Attached<br>- Moderate to High-Level<br>30-100 kA; 100 kA/$\mu$s | - Voltages and Fields<br>- External EM Models |
| AFFDL A-7<br>1977 | Direct Attached<br>- Low-level<br>2 kA; 1.25 kA/$\mu$s | - Voltages<br>- Simple Model |
| GENERAL<br>DYNAMICS F-16<br>1975 & 1976 | YF-16 #1,2 Lightning Tests<br>2-30 kA; 50 kA/$\mu$s | - Voltages<br>- Simple Model |
| NASA F-8<br>1975 | Direct Attached<br>5 kA; 5 kA/$\mu$s | - Voltages<br>- Data Analysis |
| UH-60<br>Helicopter | Direct Attached<br>- Low-level; 5 kA | - Unknown |
| 757 Transport | Direct Attached<br>- Low-level CW Test<br>250 A | - Spectral Analysis |
| ACAP<br>Helicopter | Direct Attached<br>- Low-level CW Test<br>< 20 A<br>- Moderate-level Pulse<br>24 kA; 18 kA/$\mu$s<br>- High-level Pulse<br>200 kA; 146 kA/$\mu$s | - Spectral Analysis<br>- Voltages, Currents, and Fields<br>- EM Models<br>- Extrapolated CW results 6 dB greater than measurements at 200 kA |

- Voltage and current levels high enough to excite arcs and nonlinear effects, if any.

It is not possible to meet all these requirements with any one of the simulation techniques currently in use. The best present techniques only satisfy a few of the above requirements in any one setup.

The desired output from the lightning simulation technique determines which of these features are implemented and the degree of analysis required to demonstrate lightning protection. For example:

- Full vehicle, severe lightning protection demonstration tests with a minimum of analysis require simulation of the extreme values for the lightning parameters.

- Protection design demonstration tests can be performed with lower levels of pulse or with frequency domain measurements but require the support of analysis.

- Avionics and equipment susceptibility can be determined using bench tests on equipment outside the aircraft under test but require analysis or additional testing to relate to specific vulnerabilities of any aircraft.

The most widely applicable simulation methods are the pulse testing at various levels and rates of rise and the swept CW testing. The CW method differs from the pulse methods in the type of generator used and in the amount of analysis needed for interpretation of the data.

## 4.2. Lightning Pulse Generator Techniques

Many pulse techniques exist for generating simulations of lightning waveforms to investigate the induced effects of lightning on aircraft and avionic equipment. These techniques are based on discharging a high voltage source into the test aircraft, using switches and wave shaping elements to produce the appropriate current waveform. Most lightning simulators use variations of an RLC circuit to produce some of the relevant lightning characteristics. The circuit may be configured in either the underdamped, critically damped, or overdamped configuration. Four variants of the RLC circuit are in use today: the linear damped sine wave, the critically damped, the double exponential, and a nonlinear generator having sine wave rise and an exponential tail.

The underdamped RLC circuit generates the fast rise times and a moderate-level action integral desired with a practical circuit, but the waveform is oscillatory rather than unipolar as in natural lightning. J. Robb of Lightning and Transients Research Institute (LTRI) and B. Burrows of Culham Laboratory (England) have used damped sine wave pulsers for achieving high currents and fast rise-times (Riley et al., 1984).

The critically damped circuit gives a unipolar waveform, but the decay-time to rise-time ratio is smaller than that of natural lightning. However, the simplicity of constructing this type of pulser makes it attractive to generate a moderate-level pulse. With a low inductance aircraft and return circuit this

type of pulser can provide moderate rise-times and peak currents. The AFFDL has built moderate-level pulse generators using the standard RLC circuit arrangement. Significant improvements to the pulse generator switch and transient measurement system have been incorporated into this system (Walko, 1974).

An overdamped circuit requires a large amount of stored energy to produce the desired waveform. Therefore, this configuration is generally not used to provide both fast rise times and high current levels on a single pulse. B. Burrows utilized a pulse generator with an overdamped RLC circuit. This arrangement was combined with an approximately coaxial return conductor arrangement to produce moderate-level current pulses of 20 kA with a peak rate-of-rise 170 kA/$\mu$s (Butters, et al., 1982). Field distribution calculations were used to determine the placement for the return conductors.

The nonlinear "crow-bar" generator provides an efficient means for generating high peak currents with high energy in the decay. The sine wave efficiently transfers energy from the main capacitor bank into the test aircraft inductance. This energy is then short circuited by a spark gap and discharges with a Resistive/Inductive (RL) decay, producing the long decay tail on the test airplane's current pulse. Sandia National Laboratories has built a high-level lightning simulator for the Department of Energy (DOE) using a crowbar circuit design. This produces an essentially nonlinear circuit, because before the crowbar switch is closed it is an RLC circuit and after the switch is closed, an RL circuit.

An evaluation of four simulation methods for testing for indirect effects of lightning on full-size aircraft, having advanced technology structural materials and electronics equipment, is given in Cooley and Shortess (1987). The capabilities and limitations of these methods are summarized in table 4.2-1. A detailed description of each test method is given in the following sections.

### 4.2.1. Low-Level Swept Continuous Wave

This lightning simulation technique utilizes a CW rather than a pulse to excite the transients. Because the levels of testing are generally at low levels of current relative to the severe threat lightning current, there is more uncertainty. Consequently, more analysis is required to extrapolate to severe current levels.

Basically, this method utilizes a network analyzer to measure transfer functions (amplitude and phase) from a lightning attachment point to test points (voltage or current) within the aircraft. Figure 4.2-1 shows an example test setup.

The transfer functions, measured over a wide frequency range, are subsequently processed by multiplying by the severe lightning current spectrum to develop test point spectral responses. The test point responses are then numerically Fourier transformed to generate the pulse response expected from severe lightning. The method is generally applicable to other EM threats as well. Standard swept CW network analyzers provide coupling transfer function data as amplitude and phase versus frequency.

TABLE 4.2-1. SUMMARY OF SIMULATION TECHNIQUE CAPABILITIES

| REQUIRED FEATURES FOR DIRECT STRIKE SIMULATION ON FULL VEHICLES | LOW LEVEL | | | HIGH LEVEL |
|---|---|---|---|---|
| | SWEPT CW BENCH TEST | FAST-RISE PULSE TEST | SHOCK EXCITATION | FAST-RISE PULSE TEST |
| 1. High Voltage - E Field Phase Simulating a Leader Stroke | Linear Only | Limited By Pulse | Limited By Pulse | OK |
| 2. High Current - H Field Simulating a Return Stroke Represented by Component A | Yes | Yes | Yes | OK |
| 3. Continuing Currents Which Link Successive Return Strokes Represented by Component C | Yes | Yes | Yes | OK |
| 4. Fast Rise Times (30 - 100 ns) To Match Recent Measurements | Yes | Yes | Yes | Limited By Pulse |
| 5. Coaxial Transmission Line Test Setup of Test Object/Return Circuit | | | | |
| • Simulation of In-Flight Conditions | Yes | Yes | ? | Yes |
| • Simulation of Uniform Currents and Fields | Yes | Yes | ? | Yes |
| 6. Lightning Channel Attachment | Yes | Limited By Pulse | Limited By Pulse | Limited By Pulse |
| 7. Test for Equipment Functional Responses | | | | |
| • Upset | Yes | No | ? | Yes |
| • Damage | Yes | No | ? | Yes |
| 8. Adapt Data to New Threat Parameters | | | | |
| • EM Coupling | Yes | Limited By Pulse | Limited By Pulse | Limited By Pulse |
| • Functional Responses | Yes | Limited to Observed | Limited to Observed | Limited to Observed |

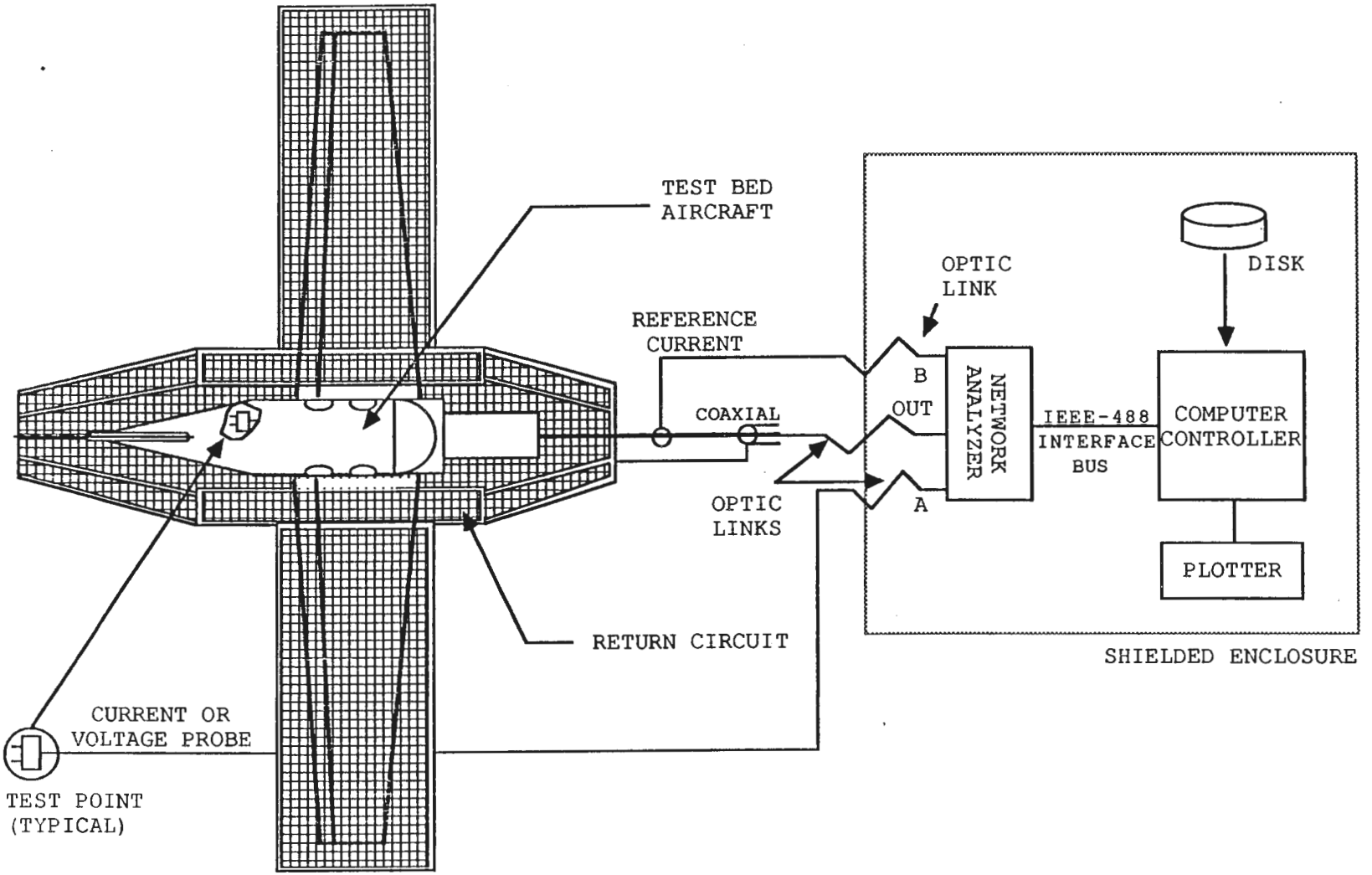



FIGURE 4.2-1. TYPICAL LOW-LEVEL SWEPT CONTINUOUS WAVE TEST SETUP

Figure 4.2-2 provides an example of measured transfer function data, (a), and the pulse response, (d), obtained by Fourier transform processing of the CW measurement. This analysis requires a significant amount of numerical processing to extrapolate values to in-flight results. The intermediate data processing steps are illustrated in (b) and (c). Between 10 and 100 MHz in (a), the noise becomes more dominant than the signal. Here a judgement must be made as to the validity of the data. If the data are not valid, an estimate of the signal response must be made over the frequency range in question. The plot in (b) shows the expected signal with the noise removed. The 200 kA double exponential threat convolved with data in (b) is shown in (c). Fourier analysis of this spectrum yields the pulse response in (d).

A principal advantage of the CW test method is the low level of injected current that can be applied (a few amperes) while attaining a high level of signal-to-noise ratios. Low levels of current can be used because of the wide frequency range contained in the lightning induced transient's interaction and coupling within the aircraft. Therefore, the energy delivered to the test object can be averaged over a relatively long time compared to pulse methods. This averaging provides an improved signal-to-noise ratio particularly when a narrow band receiver is used in the network analyzer.

Due to the transfer function complexity for most coupling responses, data are much easier to understand when obtained with CW measurements versus pulse techniques. The transfer function complexity contains clearly displayed and quantified resonances, superior signal-to-noise ratios, and lower required generator levels than the pulse methods. Under the CW method, an attached current of a few amperes can provide as high a signal-to-noise ratio as a severe lightning pulse. With these low levels of testing the CW measurement method offers several advantages for obtaining parametric sensitivity data, for example, the quick evaluation of a variety of mock-up protection options.

Nonlinearities, such as arcing and sparking at joints, are not represented with this test method because of the low currents injected. However, it has been demonstrated that extrapolated results from CW measurements are six to eight dB more conservative than higher level pulse measurements (Shortess, Cooley, and Melander, 1988 and Walen, 1988).

Swept CW measurement techniques for full vehicle lightning protection design demonstration are as follows:

- Conduct time-delay-reflectometer (very low-level pulse) measurements to identify apertures and locations where EM coupling occurs.

- Conduct CW measurements to measure transfer functions from the external lightning current attachments to the EM coupling sources, and develop transfer functions from the sources to wiring and from the wiring to equipment.

- Combine the CW measured transfer functions with a model of the external lightning threat to obtain an overall computer model for EM response prediction.

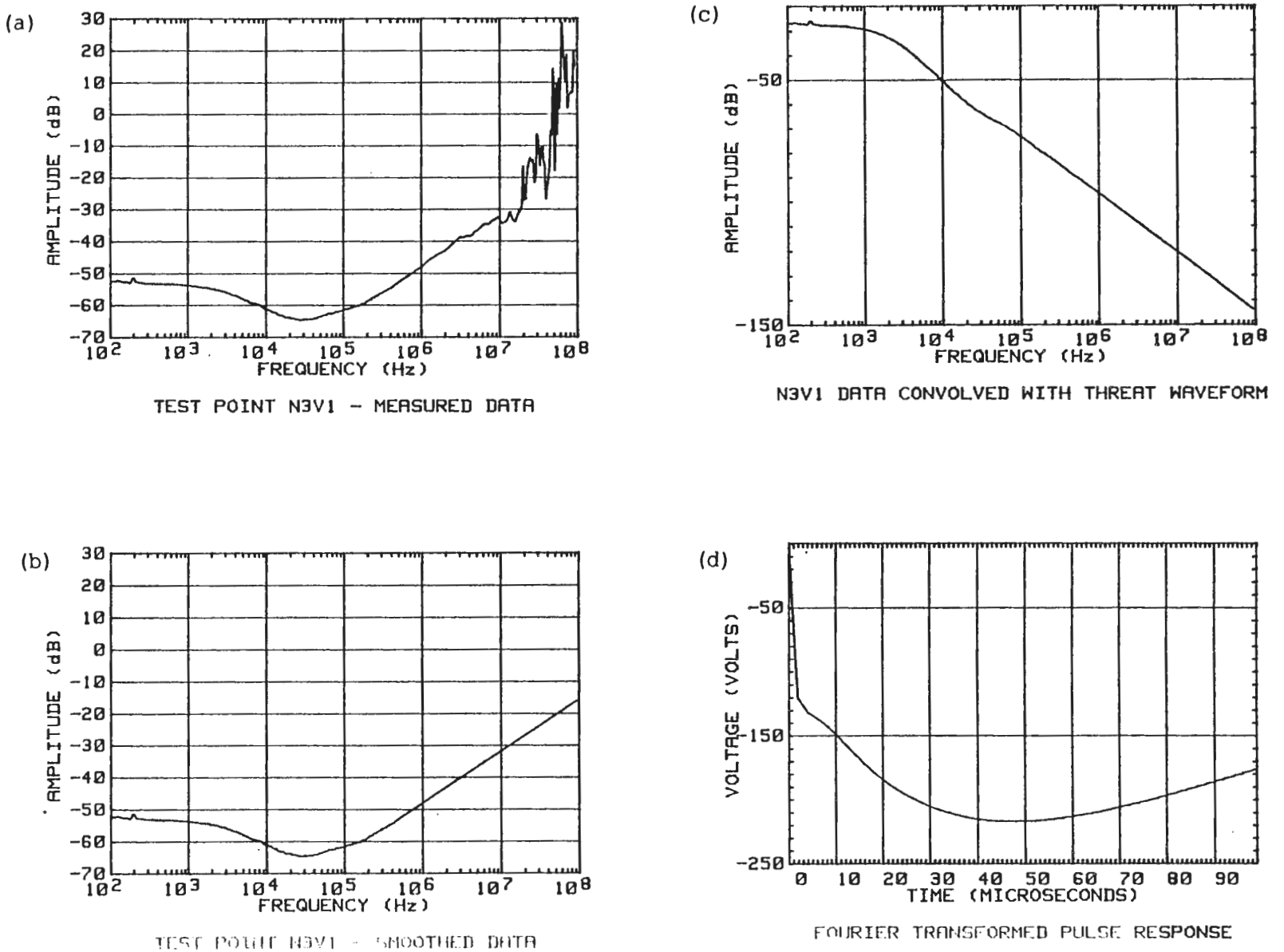



FIGURE 4.2-2. EXAMPLE TRANSFER FUNCTION ANALYSIS SEQUENCE

- Excite the computer models with the applicable severe lightning current spectrum and transform to time domain determining the full-threat level transients at critical circuits.

- Perform bench tests (induced cable current or pin voltage injection tests), using the predicted severe lightning pulses (possibly including timing from apertures and restrikes), at critical circuits on the aircraft.

- Determine protection design margins for the equipment. If the protection is inadequate, determine designs to reduce transients or reduce equipment susceptibility to the transients.

This method is also applicable to evaluating coupling coefficients for the high-voltage leader-attachment phase and nearby strike, as well as for the high-current phase by means of piece-wise linear simulation techniques. For these techniques, the generator, airplane, and return circuit are configured to represent the EM environment for each of the phases, one at a time. The coupling parameters included for the current and EM fields response of a given subsystem to the various aspects of the threat may be used to define the appropriate threat waveforms for a bench test. These time-ordered threats may then be imposed upon the component or subsystem in the proper sequence.

4.2.2. Low-Level Fast-Rise Pulse

Under this method, current pulses are generated by means of a pulse generator that is tightly coupled to the test aircraft (see figure 4.2-3). For this application, the test aircraft return circuit inductance and pulse generator capacitance combine to provide a critically damped circuit. This results in a current pulse having a high rate of rise with a relatively short decay time. The parameters are chosen so that the peak rate-of-rise matches the rate-of-rise for severe lightning currents while the peak amplitude matches the peak amplitude for low levels of lightning current. The generator decay time is much faster than natural lightning.

This simulation provides stress levels that match the severe rate-of-rise of natural lightning. This is appropriate when the principal EM coupling mechanism is mutual inductance or mutual capacitance, as is the case for balanced circuits isolated from structure. For composite structures where the coupling mechanism may be due to resistive drops, the simulated currents are not high enough to match the expected levels of moderate lightning strikes.

Since such a generator produces less than a moderate lightning stress on much of the equipment in an aircraft, the survival of equipment under the test condition only provides confidence in the capability to survive minor lightning strikes. The low-level of current injected will not be sufficient to cause non-linear responses, functional upset, or damage responses.

Careful analysis and interpretation of the test data is necessary to provide confidence in the capability to survive severe lightning strikes. This analysis must include interpretations of the coupling mechanism (I-dot or I) for each cable run and equipment area on the aircraft. The coupling mechanisms are necessary to extrapolate measured voltages and currents to higher severe

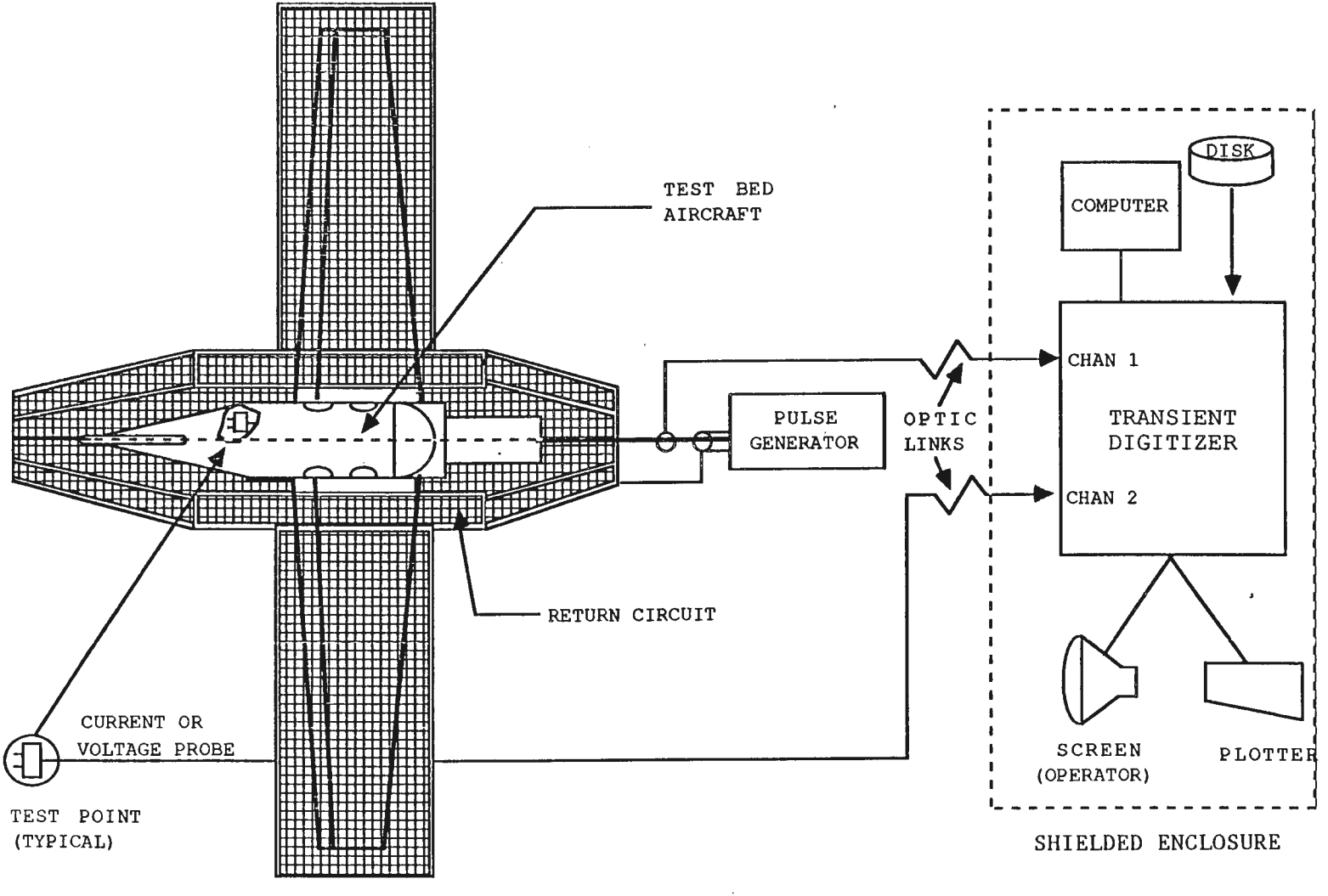



FIGURE 4.2-3. TYPICAL PULSE SIMULATION TEST SETUP

lightning responses. The I-dot driven responses scale according to the rate-of-rise parameters while the resistive driven responses scale according to I parameters. For many responses it will be found that they are driven by both I and I-dot coupling mechanisms. Since the generator circuit is tightly coupled to the aircraft under test, it may not be possible to separate the I and I-dot coupling from the data. This is because the internal coupling in the aircraft provides a different scale factor for I and I-dot at different test points, due to internal RLCM parameters, as opposed to the fixed I/I-dot parameters built into the generator. Therefore, the separation of the test results into purely I or I-dot will not be possible without considerable test efforts using different generator parameters.

If the generator waveform parameters match lightning parameters except for peak amplitude, then the response can be scaled by a multiplicative constant. For any other generator or choice of waveform parameters this will not be the case.

Based upon linearity arguments, Walko (1974) developed such a test based upon the use of a very low-level double exponential current pulse with a peak magnitude as low as 200 A. If rise and fall time constants match the values for lightning, then this current pulse theoretically contains all of the frequency components of a 200,000 A waveform in proper proportion, thereby allowing transients to be scaled linearly. The pulse is passed through the aircraft structure (from nose to tail, for example). The resulting transients produced on critical wire runs inside the aircraft are monitored by connecting voltage/current measurement instrumentation across the circuit to be monitored. The peak values of the measured transients are then extrapolated upward to the severe lightning current level. In the case of a 200 A, 6 x 70 microsecond test pulse, the peak levels are multiplied by 1000 to determine the anticipated transients for a 200,000 A strike with the same waveform. Since the test wave shape is the same as the threat waveform, the I-dot and the peak current, I, have the same scaling factor. This is illustrated for 200 kA and 200 A double exponential current pulses in figure 4.2-4.

Unless the generator rise and fall time constants match lightning parameters, it is probably better to use the basic measured responses as representative of a low to moderate lightning stress and not try to extrapolate the measured responses to higher levels.

The most accurate approach to extrapolation of the voltage and current measurements to higher levels of stress is through an analytical model based upon electrical equivalent circuits. This approach to modeling is similar to the analysis required for the CW method. The major difference is that the measured data for comparison with the models is derived from the pulse measurements.

The digital recording equipment used to acquire data from pulse testing typically has a dynamic range of only 40 dB. This is several orders of magnitude less than the dynamic range obtainable when measuring in the frequency domain.

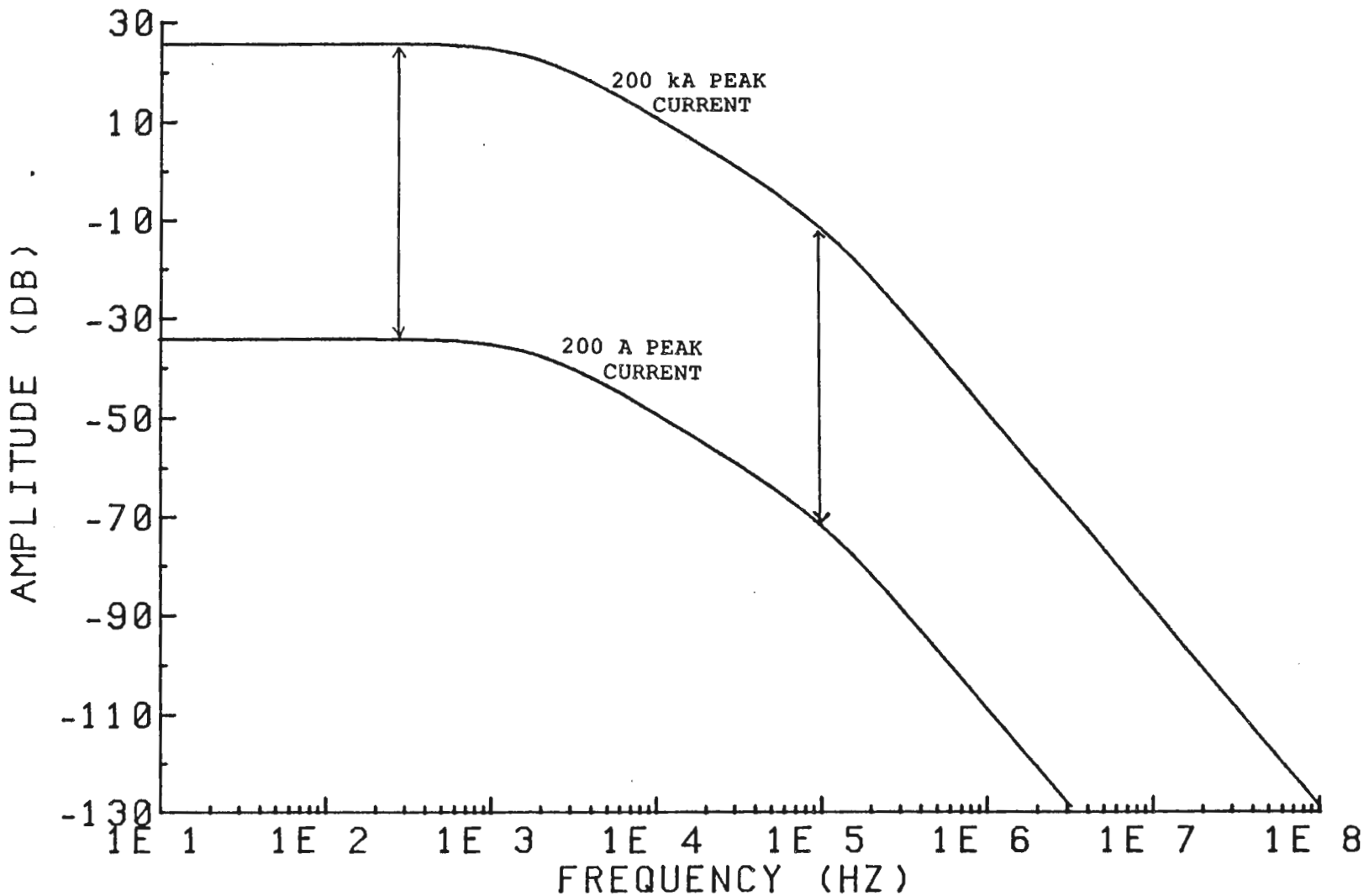



FIGURE 4.2-4. COMPARISON OF FREQUENCY SPECTRA

Low-level pulse measurement techniques for full vehicle lightning protection design demonstration are as follows:

- Conduct time-delay-reflectometer (very low-level pulse) measurements to identify apertures and locations where EM coupling occurs.

- Conduct CW measurements to measure transfer functions from the external lightning current attachments to the EM coupling sources. Develop transfer functions from the sources to wiring and from the wiring to equipment.

- Combine the CW measured transfer functions with a model of the low-level pulse generator to obtain an overall computer model for EM response prediction of the test responses.

- Compare the measured pulses with the model predictions to provide confidence in the computer models. Determine the error bars in the model responses from the measurement comparisons.

- Excite the computer models with the applicable severe lightning current spectrum and transform to time domain determining the full-threat level transients at critical circuits.

- Perform bench tests (induced cable current or pin voltage injection tests), using the predicted severe lightning pulses (possibly including timing from apertures and restrikes), at critical circuits on the aircraft.

- Determine protection design margins for the equipment. If the protection is inadequate, determine designs to reduce transients or reduce equipment susceptibility to the transients.

It should be noted that second and third bullet items utilize results from CW testing. Bullet items utilizing the CW method provides a higher degree of confidence in the modeling of the various coupling mechanisms. The pulse data could also be used to refine the models but there would be lower confidence in the extrapolation using a model that had been refined with the pulse test alone. This is due to the difficulty of separating the various RLCM parameters from the time domain test responses.

4.2.3. High-Level Fast-Rise Pulse

Under this method, current pulses are generated by means of a large pulse generator that is tightly coupled to the test aircraft and return circuit. There is considerably more energy required in the main pulse generator for this method than the low-level fast-rise test method. In the usual application, the test aircraft and return circuit resonate with the pulse generator capacitance to provide high rate-of-rise current pulses having a damped sinusoidal waveform. The generator and return circuit parameters are chosen so that the pulse rise time matches the rise time for severe lightning currents. The peak amplitude is near enough to the severe lightning current levels that the data extrapolations can be made with high confidence. The generator decay time can be adjusted to the same as natural lightning or much faster than natural lightning.

This simulation provides levels at and above the moderate lightning strike current amplitude. This test method is appropriate when the amount of analysis is to be minimized. The analysis provides high confidence in the survival of the aircraft and equipment under severe lightning strikes.

The high-level pulse simulation technique provides a level of stress that is somewhat greater than the expected value for lightning strikes. This test stress increases the confidence in the protection design over that of a low level test technique used alone. For example, portions of the protection system, such as cable shields and grounding schemes, could function at levels associated with low-level testing but be damaged at higher levels. The test conditions provide confidence in the survival of the aircraft against all but severe lightning strikes.

The principal advantage of this test method is that the amount of scaling or extrapolation is not very great. Careful interpretation of the test data is necessary in order to provide confidence in an aircraft survival of severe lightning strikes. However, there is greater confidence in scaling factors of five to ten (from 20 or 40 kA to 200 kA) than factors of 200 required for the low-level pulse method. The data analysis must of course include interpretations of the measurements to validate the measurement process. The data analysis effort is much simpler and more readily accepted than the analysis required for lower level test methods.

4.2.4. Severe Full-Threat Pulse

Unfortunately, it is difficult or impossible to generate the severe lightning currents in large aircraft as can be demonstrated by a relatively simple analysis. The primary reason for this limitation is due to the inductance and resistance of the return circuit and the airplane. For an ideal conductor and the return circuit in a coaxial configuration, the test circuit surge impedance is typically 50 ohms. For 200 kA, this requires 1 MV to drive the test aircraft and 1.3 MV for the charge voltage of the pulser. These values are achievable but require specialized high voltage construction practices. For a large aircraft, the circuit impedance is more inductive. The inductance is approximately given by:

$$L = 120 \ln(h/2d)\ h/c$$

where h is the aircraft length, d is the aircraft diameter, c is the speed of light, and the units of the constant are henrys per second. For a 50-meter aircraft 3 meters in diameter, this results in an inductance of approximately 40 $\mu$henrys. It requires 8 MV to drive 200 kA/$\mu$s into this inductance. Such a generator is not presently achievable. However, for a small general aviation size aircraft the inductance can be on the order of 3 to 4 $\mu$henrys and only 600 to 800 kV is required to reach the 200 kA/$\mu$s rate-of-rise. While this is a large order for most general aviation manufacturers, such generators are well within the reach of several lightning test organizations.

### 4.2.5. Shock-Excited Pulse

The pulse techniques described above focus on directly attached current waveform simulation and associated EM fields. There is some concern that the high voltage and electric fields induced on an aircraft, just before a strike and during the aircraft charging, should also be considered in the simulation. The electric field mode can be simulated by using a spark gap between the aircraft and the pulse generator and between the generator and the ground. These spark gaps are to be placed at a structural extremity away from the pulser attachment. When the pulser is fired, the aircraft will be charged, through the pulser gap, toward the pulser voltage. The charging will be terminated when the aircraft-to-ground spark gap discharges to ground. This technique produces high voltage and electric field gradients on the aircraft to simulate the natural lightning effects.

This test method simulates the presence of corona, the leader attachment during the charging phase, and a return stroke as the aircraft discharges. The coupling mechanisms are more easily separated for analysis because the charging and discharging events are separate in time.

The major setup difference from prior techniques is that, in the shock-excitation test, the aircraft functions as the peaking capacitor. If the aircraft is isolated from ground, an additional peaking capacitor may be unnecessary. The discharge of the aircraft, when the output gap arcs over, should excite the internal circuits in much the same way as the discharge of an aircraft in a preionized lightning channel would when it is discharged by the return stroke.

Shock-excitation tests are unique because of the high-voltage aspects of the technique. The aircraft must be well isolated above ground. Specially designed high-voltage isolation pads placed under the wheels allow testing up to 400 kV on most aircraft.

A shielded high-voltage impulse generator is used as the excitation source. This shielding is required so that a clean output voltage waveform may be applied to the vehicle. To simulate the natural lightning charging and discharging of the aircraft, it is desirable to charge the aircraft to over one million volts with respect to the coaxially arranged return lines. At these voltages, realistic corona should be developed so that, when the aircraft discharges to ground, the transients produced will be representative of natural lightning. Essential elements of the test are fiber-optic data links which allow system transients to be monitored while the system is charged to very high voltages.

An important result of the shock excited test investigation (Clifford, Crouch, and Schulte, 1982) was that capacitive coupling (C dV/dt) was found to be the dominant coupling mechanism in some very important cases. In particular, the transients on high-impedance signal circuits used for single-wire computer logic circuit interconnections were found to be dominated by capacitive coupling. The situation is reversed for low-impedance circuits where inductive (L dI/dt) coupling dominates.

For these tests, the maximum values of the lightning parameters of interest (I, dI/dt, E, and dE/dt) are determined by the amount of charge stored on the aircraft before discharge occurs.

The major implication of these results for simulation testing is that sufficiently high voltages and rates of change of voltage must be present in the test to reproduce the natural lightning conditions. Since nonlinear corona and streamering effects are expected to play a role in the induced voltages and currents experienced by an aircraft struck by lightning, testing with low voltages (grounded vehicles) may not yield an accurate simulation.

The amplitude and duration of the oscillatory currents produced on the structure by the rapid discharge of the aircraft to ground (free response) are controlled by the dissipation factors in the test setup. These factors include arc impedance and corona streamering effects. When the test article is grounded, neither of these dissipation factors is present. It still remains to determine exactly what the correct values for the dissipation factors are, but additional laboratory studies, coupled with forthcoming flight data, should resolve those uncertainties.

4.2.6. Comparison of Test Methods

In a comparison of lightning simulation techniques on a composite test article (Cooley and Shortess, 1987), it was found that mean values of predicted currents due to severe lightning strikes for different techniques compared as follows:

- low-level swept CW +6 to +8 dB
- low-level pulse -1 to -5 dB
- moderate-level pulse baseline
- shock-excitation +2 to +6 dB

The moderate-level pulse results were used as a baseline. These data were considered more accurate than data measured using other techniques for the following reasons:

- The scaling factor from 7 kA to 200 kA had less uncertainty than for low levels of injected current.
- Non-linear effects were noted at several test points. Non-linear effects were not in evidence for the low-level tests.
- The spectral content was similar to the SAE-AE4L defined threat, Component A. This allowed a minimal amount of analysis because a determination of resistive or dI/dt response was not necessary.

The CW predictions were larger (by a factor of two) and therefore more conservative than the predictions resulting from the moderate-level pulse tests. The

low-level pulse results predicted responses about 1.5 times less than the moderate-level pulse. Shock-excitation was also a factor of two more conservative than the moderate-level pulse results.

4.2.7. Return Circuit Configuration

The return circuit for the lightning simulator and test article determines the field distribution around the object under test. More uniform field, voltage, and current distributions are produced by configuring the test article and return circuit as a coaxial transmission line, see figure 4.2-5. The electric and magnetic fields around the test object approximate field distributions for an aircraft in free space. An additional benefit is that this arrangement provides low circuit inductance and resistance which is required by some pulsers to produce a satisfactory waveform. This is particularly true for the RLC pulse generators.

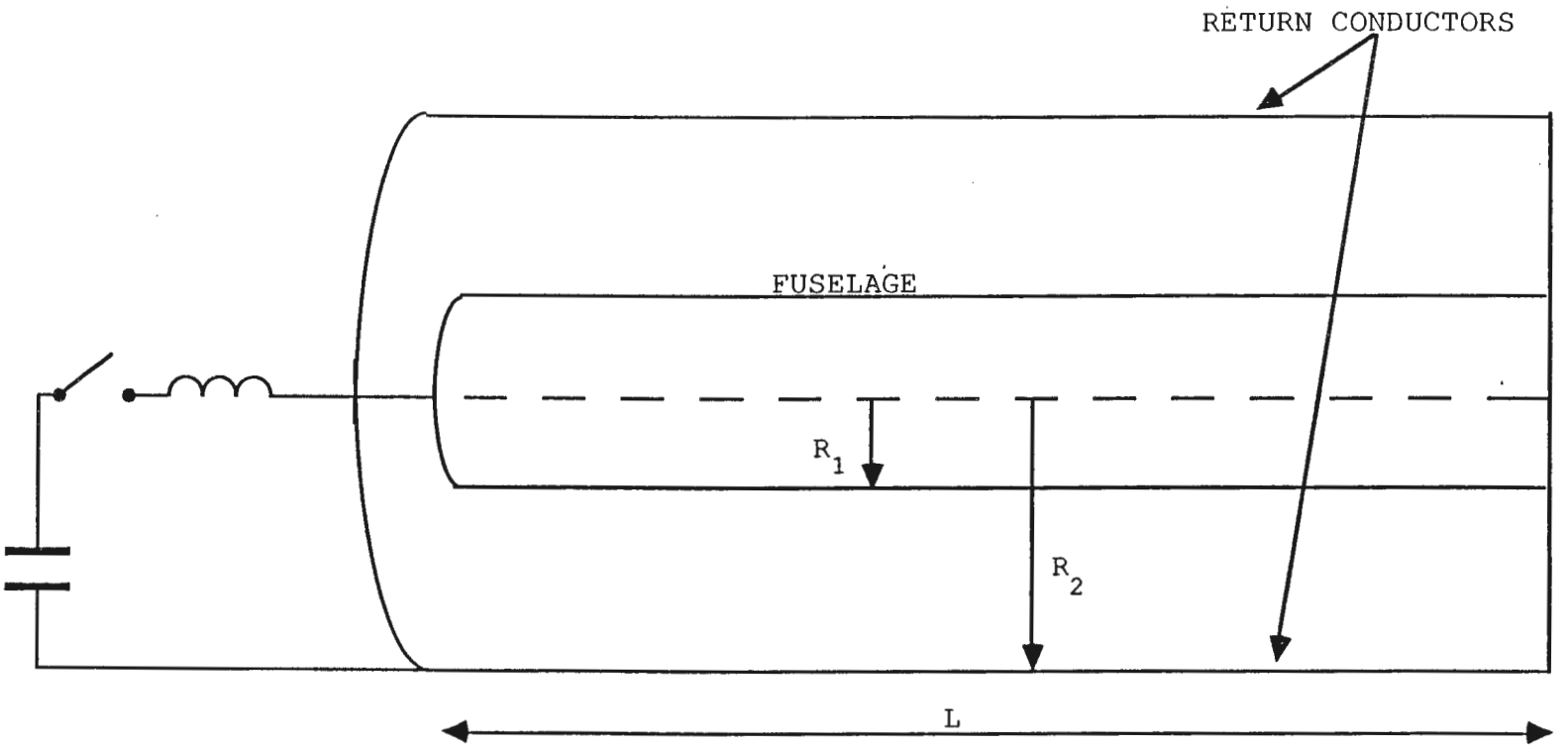
Some special considerations must be given for the construction of a return circuit. For high-level testing, the return circuit must sustain high currents and withstand high voltages. The return circuit must be made of a low resistance material, such as wire mesh or solid sheet, and maintain good physical contact to be suitable for low-level testing. The attachment points must be low inductance, have a high current carrying capability, and shield the test object from the fields of the pulser switch. The configuration should be built to accommodate all test configurations, e.g., nose-totail and wing-to-tail current injection. The construction should be such that the spacing between the return circuit and the test object maintains a constant impedance transmission line and any changes in return circuit geometry should be gradual compared to a wavelength of the highest frequency of interest. This is essential to prevent reflections due to impedance mismatches.

4.3. Data Quality Verification

Several recommendations are given in this section and can be used to assure that measured data is of high quality and is valid. These recommendations can be utilized for both CW and pulse tests.

Fiber optics should be used in both drive and signal transmission, if possible. The optics electrically isolate equipment to prevent ground loop currents. The data acquisition equipment should be placed in a screen room to provide shielding from pulser fields and to eliminate noise pickup. Pulser noise can be a major concern at high levels of current.

System calibrations must be performed daily to verify the correct operation of the CW measurement equipment and to obtain inherent system responses for proper data calibration. The measurements should be verified twice daily or at any time the system configuration changes. The data obtained from system calibrations are then subtracted from measurement data to remove inherent system responses. Calibration of some fiber optic units is necessary to define the amplitude of a known signal into a digitizer.




SIMPLIFIED AIRCRAFT GEOMETRY - THE AIRCRAFT FUSELAGE IS SHOWN AS A CYLINDER $R_1$ METERS IN RADIUS, L METERS LONG WITH RETURN CONDUCTORS SPACED AT AN EQUIVALENT RADIUS OF $R_2$ METERS.

FIGURE 4.2-5. SIMPLIFIED RETURN CIRCUIT

Noise measurements should be taken at each measurement location in each test configuration. The sensors are either a coaxial cable terminated in 50 ohms or a current probe with nothing through its center near test wiring as input to the fiber optics. These measurements can then be compared to the corresponding test point measurement to determine the signal-to-noise ratio. This is much easier to do in the frequency domain; however, a reasonable judgement can be made for a time domain measurement. A signal level 10 dB above the noise level is considered reliable data. Examples are shown in figures 4.3-1 and 4.3-2 of signal and noise measurements at the same location for CW and pulse tests.

End-to-end resistance measurements of the test setup should be made before and after each test configuration. This will be a measure of how well structural connectivity (joints, etc.) is maintained after the injection of current. An ohmmeter capable of accurately measuring a milliohm is necessary equipment for this check.

The input current should be monitored for every measurement. This current is used as a reference measurement during CW testing. During pulse testing, the input current must be checked to ensure proper pulser operation. The pulse amplitude will vary with each shot and should be recorded in the case of large variations. Data must be taken not only to characterize the simulator performance, but also to characterize the interaction and coupling of the simulated lightning environment with the test aircraft and the coupling effects on electronic/electrical systems.

Questionable current measurements can be checked in the following manner. With the current probe reversed, repeat the measurement and check the signal polarity. If the signal does not change polarity, then noise is being measured.

Validity checks should be performed by the experimenter during each test. These checks consist of comparing measured responses against theoretical models for their behavior to make sure the measured response curves are "reasonable." Any failure to produce "reasonable" data should be considered a failure in the instrument setup and/or test connections, and steps should be taken to correct the problem.

When using digital recorders, a vertical resolution should be set to obtain the measured response with the peak at least one-third of full scale to achieve adequate resolution of the signal response and to utilize the maximum bandwidth capabilities of the digitizer.

## 4.4. Bench Tests

Bench tests are used to test the susceptibility of individual components or subsystems to predicted levels. These predictions are based on measured data or analytic models for currents and fields at the location where the equipment would be installed.

Voltage and current transients induced in interconnecting wiring may be resistive, following the double exponential lightning source; a derivative, dI/dt, of the excitation source; or oscillatory, due to excitation of cable resonances.

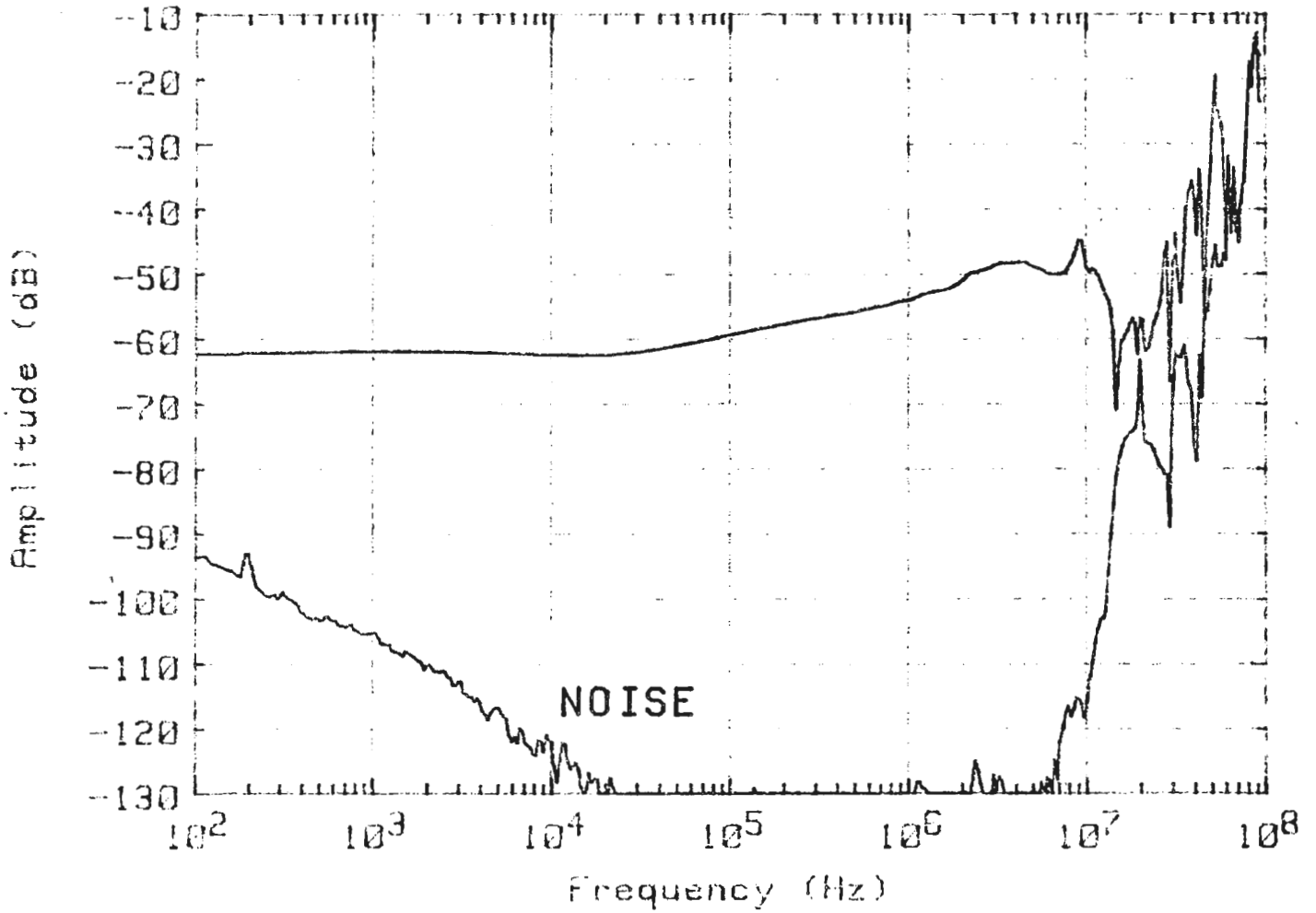



FIGURE 4.3-1. COMPARISON OF SIGNAL AND NOISE IN THE FREQUENCY DOMAIN

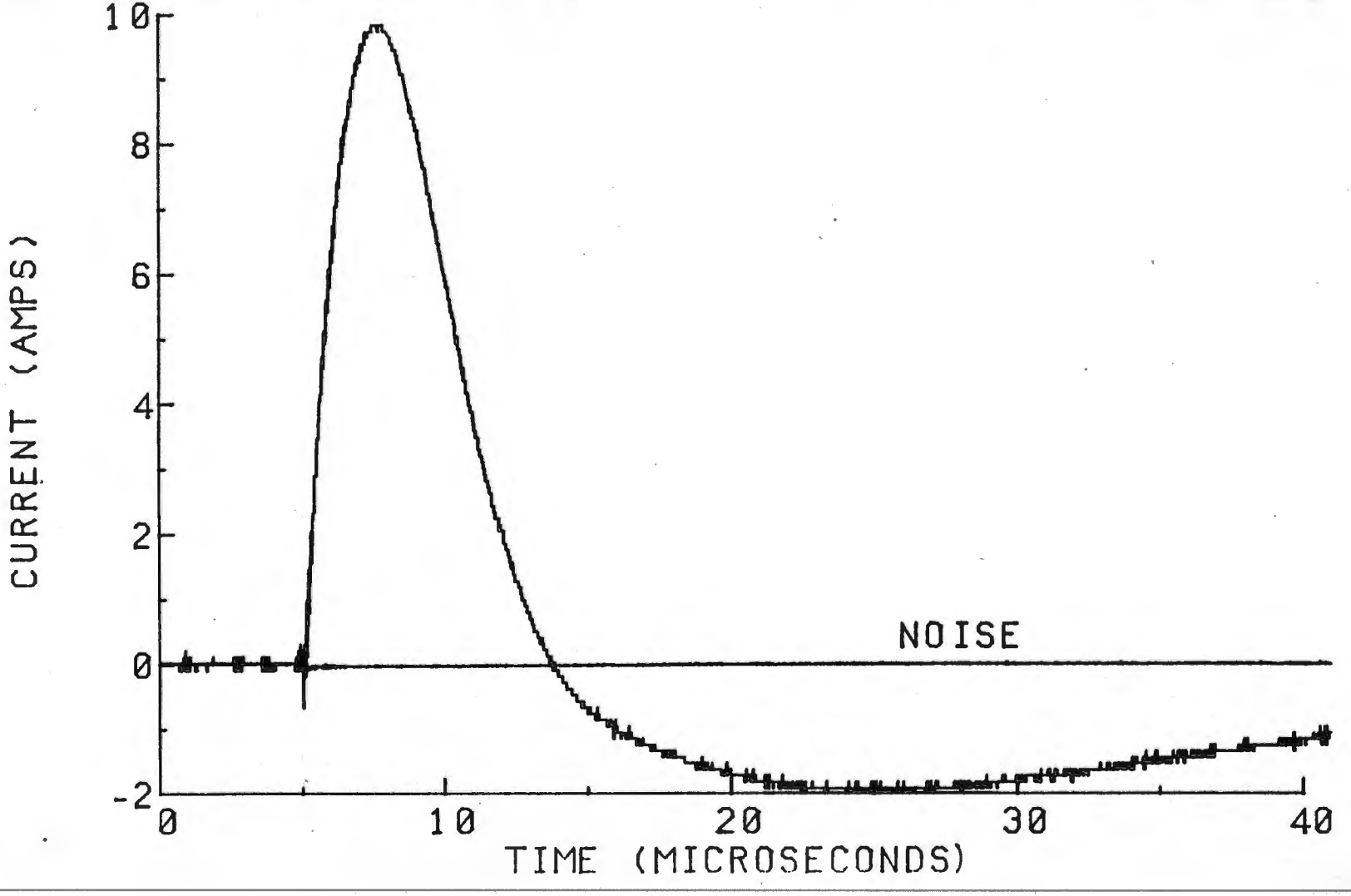





FIGURE 4.3-2. COMPARISON OF SIGNAL AND NOISE IN THE TIME DOMAIN

Waveforms, which define these responses, are shown in figure 4.4-1 (AC 20-53). These waveforms also represent the threat imposed on equipment at the cable interfaces.

Currents are injected into interconnecting cables by magnetic coupling using a clamp-on current transformer. Alternatively, electric and magnetic fields are injected into cables and enclosures by a parallel plate transmission line excited by a high current pulser. Power system transients are induced by direct series connection of a coupling capacitor and pulse generator across the power supply. A typical bench test setup for current injection is shown in figure 4.4-2.

Equipment is connected via appropriate cabling to simulate actual installed configurations. During the pulsing, the equipment is powered up and in an operational mode to simulate in-flight conditions. Equipment is monitored for damage or upset throughout the test sequence.

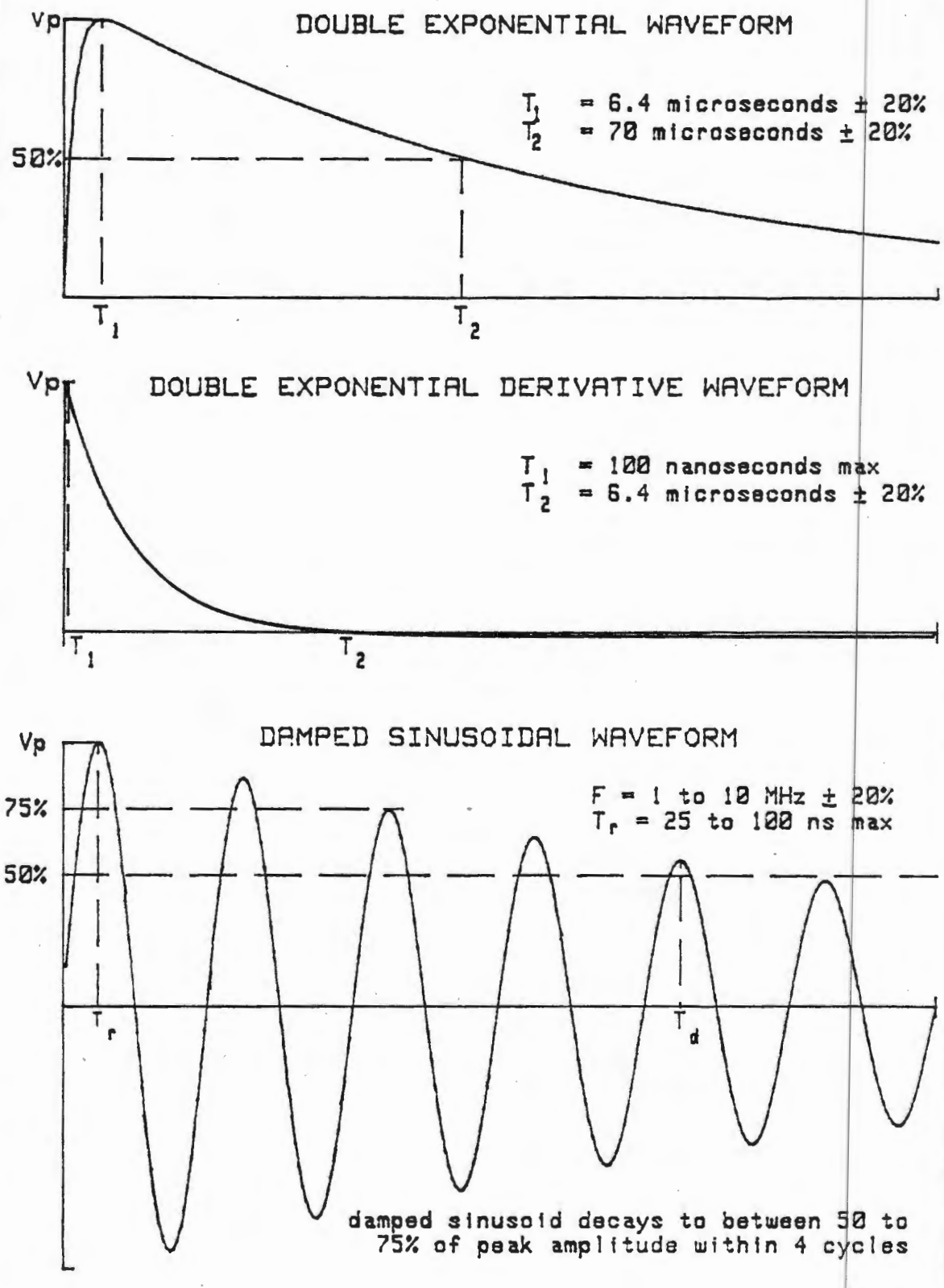
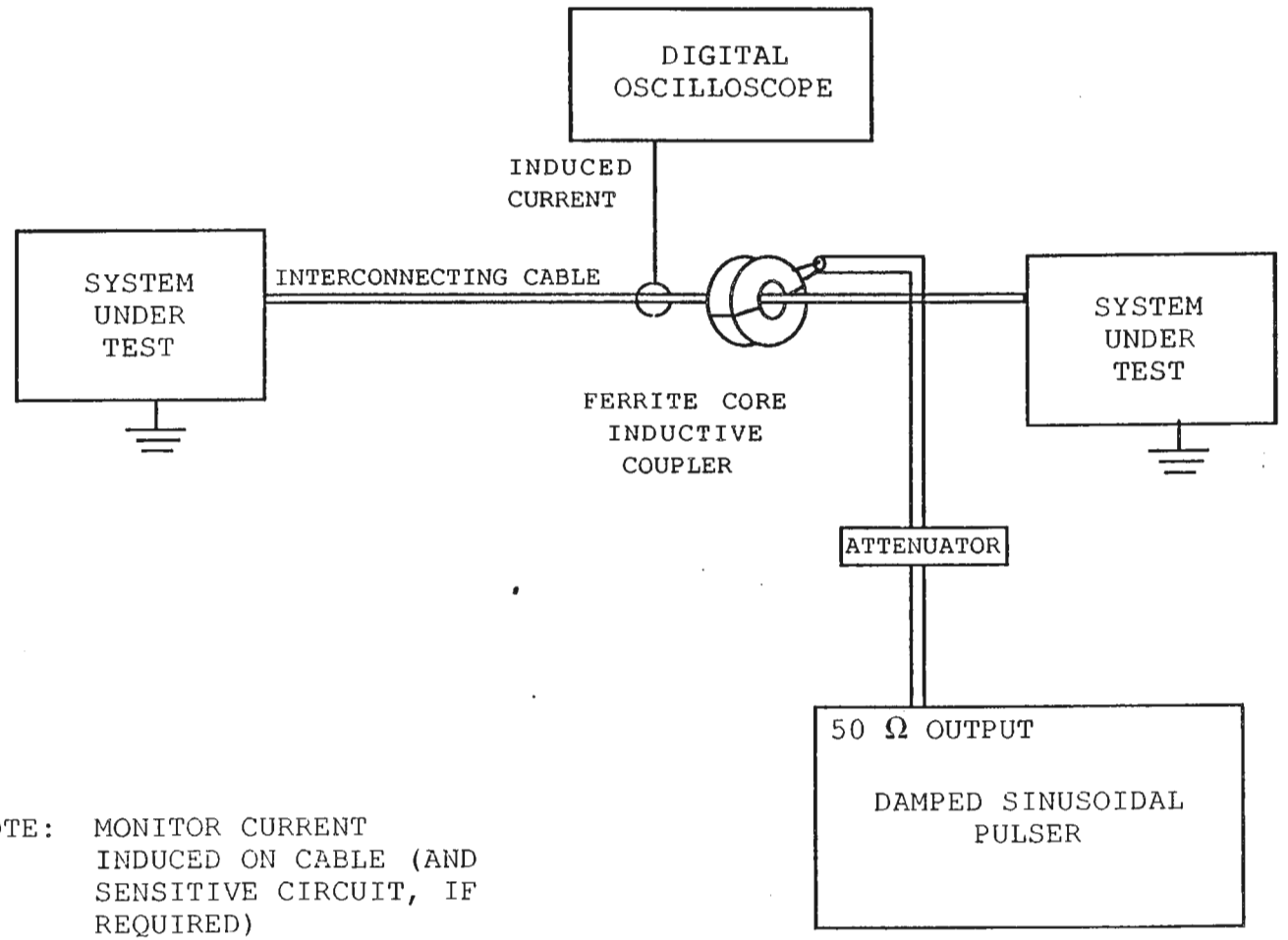



FIGURE 4.4-1. IDEAL BENCH TEST WAVEFORMS




NOTE: MONITOR CURRENT INDUCED ON CABLE (AND SENSITIVE CIRCUIT, IF REQUIRED)

FIGURE 4.4-2. CABLE CURRENT INJECTION TEST CONFIGURATION

## 5. INDIRECT EFFECTS PROTECTION - ANALYSIS TECHNIQUES

Accurately predicting the coupling of lightning to equipment within a structure such as an aircraft is a complex process. Figure 5-1 illustrates the principal elements of EM coupling prediction for lightning indirect effects. These elements follow the flow of energy from external environments to internal equipment.

The external coupling region contains resistive and inductive terms. The early time, inductive term appears as magnetic fields around the aircraft. These fields will build up and change rapidly in accordance with the lightning current pulses. The late time, resistive term appears as a voltage along the structure. While both terms are always present, the relative size of each depends upon the lightning pulse shape.

The internal fields present depend on several factors: penetrations through apertures, antennas, external wiring, and joints; diffusion through the skin; shape factors; and intrinsic shielding provided by the structure materials. External EM fields will leak into the interior of the aircraft through openings in the structure such as windows, radomes, access panels, and doors. Electrical imperfections such as joints, gaps, and holes also allow the entry of some EM fields. In addition to the EM field coupling, there may be resistive voltage drops in the structure as lightning currents flow. Currents may also flow inside the structure as a result of structural interconnections which can affect the internal EM environment. Leakage through apertures contributes to fast rise voltages. Effects due to diffusion through the skin will have a slow rise time compared to lightning pulses. Diffusion effects become important at frequencies where the structure thickness is equivalent to or greater than an electrical skin depth of the material.

Sources of internal currents in the wiring include B-dot coupling, E-dot coupling, and resistive voltage drop (IR) coupling. The magnitude of these sources is dependent on local structure, currents, and fields coupled to the interior.

Voltages are coupled into aircraft wiring in several ways. If part of a circuit connecting electronic equipment connects to structure, there will be a voltage difference between the wires and the structure. Voltages are also coupled between wires by electric and magnetic induction, even if the wires are not connected to the structure. The effects of induction depend upon the time rates of change of the lightning currents and EM fields. Since the total structural voltage drop depends upon both the inductive and resistive terms, voltages in the wiring depend upon the lightning current time rate of change as well as the peak current values.

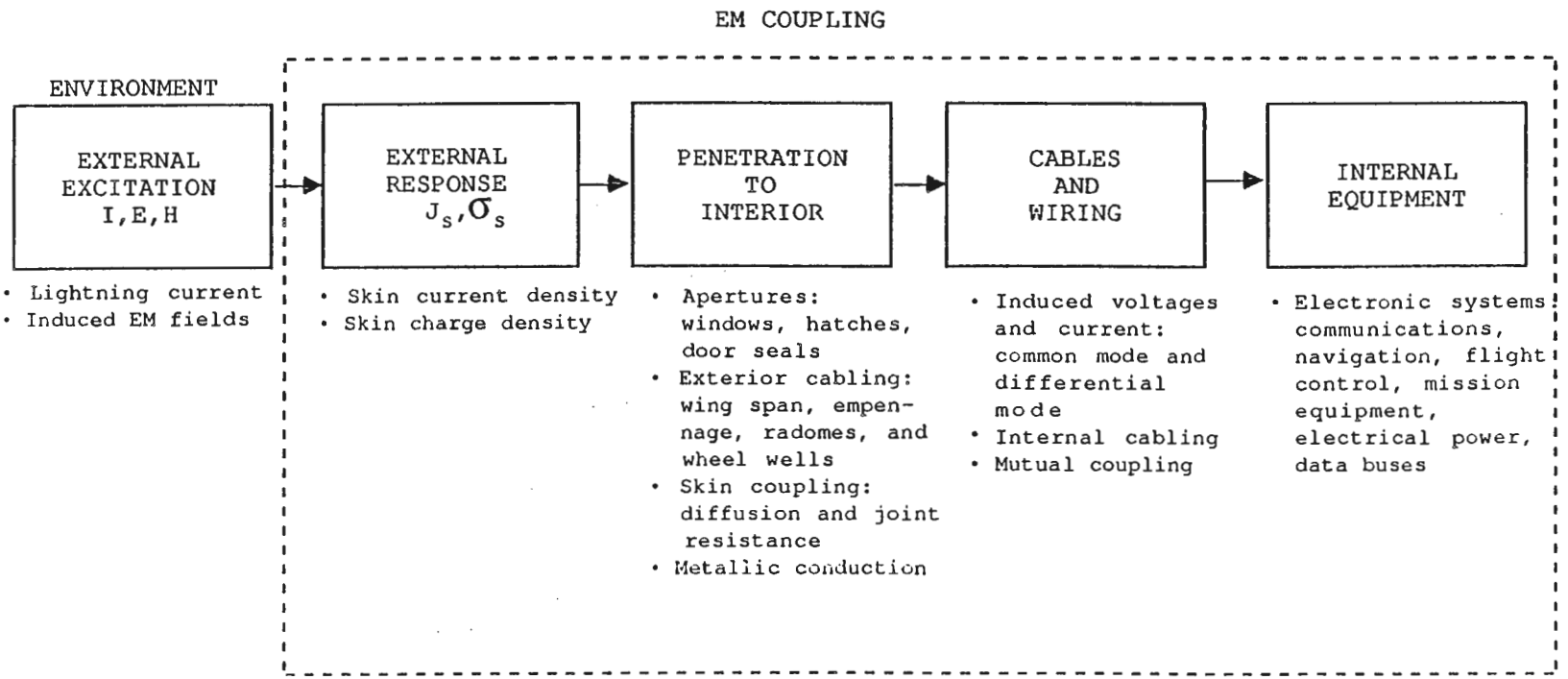



FIGURE 5-1. ELEMENTS OF EM MODELING AND ANALYSIS FOR LIGHTNING INDIRECT EFFECTS

For an aircraft made of aluminum, the coupled voltages are rarely important except when the lightning current flows through joints and hinges. However, the resistance of an advanced structural material such as G/E is many times that of aluminum. Voltages of a few tenths of a volt have no effect in an aluminum structure. Voltages that are larger by a factor of several hundred to a thousand times than those for aluminum, because of the lower conductivity of the composite materials can become very serious.

Coupling to electronic equipment is dependent on the termination and source impedances, the circuit configuration, and the wire bundles which connect to each box.

## 5.1. Modeling Methods for Composite Structures

In advanced technology aircraft, EM coupling models for lightning protection design must account for the composite structural materials and the fastener joining techniques. Many of the present models (such as the environment, external response, cable and wiring, and internal equipment) used for metal aircraft are useful for composites. The penetration to interior models require changes for particular aircraft and wiring configurations.

The modeling of energy penetration from the exterior to the interior has several common elements for any aircraft particularly in the apertures and exterior wiring and cables. The principal difference between metal and composite aircraft lies in the increased contribution of the IR in the fuselage and wing skins and in the increased current flow paths in the structure. The G/E structure has typically 1000-3000 times higher resistance than the similar aluminum metal structure. This higher resistance leads to strong interactions between the external and internal EM regions for composite structures.

For frequencies above a few megahertz, or for early time (i.e, on the order of a few microseconds), there is no difference in the lightning response between aluminum and graphite structures. This is because at high frequencies, the EM "skin effect" forces the currents to the outside of materials, to outside surfaces, and to sharp radius of curvature portions of the structure. Protection against the early pulse is essentially the same for both structural materials. Wiring must be kept near metallic structure to reduce the magnetic field B-dot coupling factors.

The main difference in energy penetration between graphite and aluminum structures is in the frequencies below the megahertz range, or for late time (i.e., on the order of a few tens of microseconds). For this range of frequencies, considerable current flows into the interior as a result of the range of thicknesses and materials used in aircraft structure. Since the current flow is resistive for G/E materials, the protection measure of keeping the wiring close to the structure will not have any protective effect. Furthermore, the IR voltage drop will be much larger for G/E than for aluminum.

## 5.2. Computer Modeling Codes

Analytic models are useful tools in determining the effectiveness of various protection designs. Analytic models can be used in parallel with testing to validate test results or to assess various protection methods for use on particular equipment.

There is no single analytic method which can provide accurate results at all levels of EM coupling, from lightning interaction with an aircraft to transients in the wiring. A multi-level analysis, as shown in figure 5.2-1, can be used to provide answers to the overall coupling problem. Using this approach, the exterior currents on the airframe due to a lightning strike are calculated first. Next, the interior fields and shield currents are calculated. These are determined by penetrations due to joints, apertures, and diffusion. The final level of analysis is to compute the transients at the component level. The input currents to each level are those calculated from the preceding level.

This type of analysis assumes that the coupling between levels is negligible. This implies that metal structure or wiring inside the aircraft does not significantly affect the exterior skin currents. This assumption is not valid for an aircraft consisting of mainly G/E skin. In this case, internal metal structure will carry the bulk of the current at low frequencies. Internal fields will be dominated by these currents rather than diffusion through the composite material. Models of mixed materials aircraft need to include internal structure in the external coupling model to accurately model current flow through the aircraft.

A description of some of the various computer codes available, their limitations, and the type of analysis to which they are suited is given below. Table 5.2-1 summarizes the capabilities of each program. The User's Manual for AC 20-53 gives additional details and references for some of these codes.

Surface currents have been calculated using finite difference time domain and method-of-moments techniques. Method-of-moment codes use a wire grid, surface patches, or strips to model the vehicle surface.

WIRANT, Numerical Electromagnetics Code (NEC), General Electromagnetic Model for the Analysis of Complex Systems (GEMACS), REDIST, THREDE, T3DFD, Thin Wire Time Domain (TWTD), Lumped Parameter Network (LPN), and transmission line models have all been used to determine surface currents. Care must be taken in the model definition to provide valid results.

WIRANT is a wire grid method-of-moments code which can be run on an IBM-XT or compatible personal computer, or a mainframe computer. The calculated currents and fields are output in the frequency domain. The code handles a frequency range which is suitable for lightning. It has been used to model several aircraft, e.g., Boeing 757, to determine EM coupling due to lightning. Results have compared favorably with experimental data. Plane wave excitation or current injection may be used to drive the models. Mixed materials may be incorporated into these models.

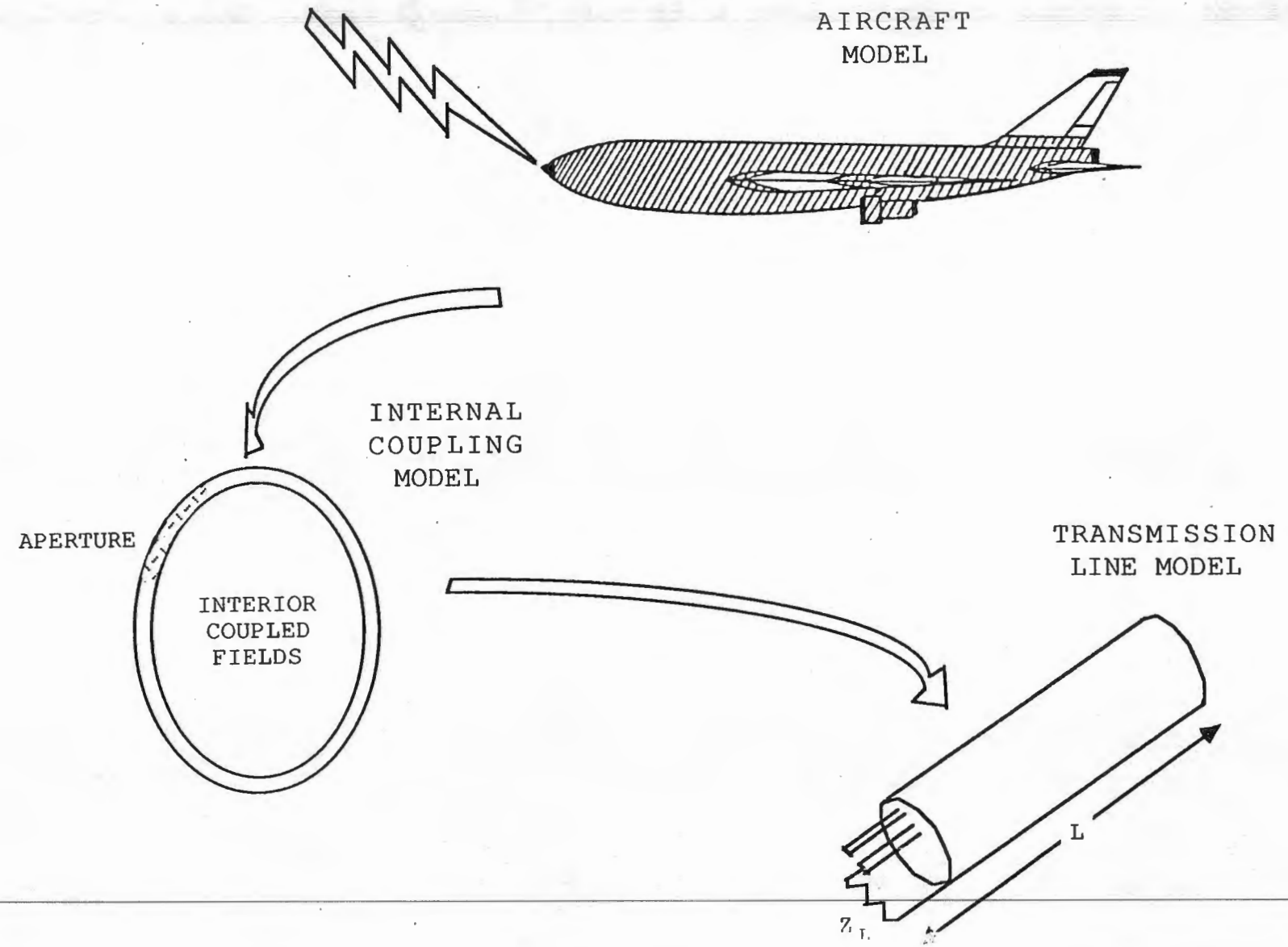



FIGURE 5.2-1. MULTILEVEL LIGHTNING COUPLING ANALYSIS APPROACH

TABLE 5.2-1. CAPABILITIES OF COMPUTER CODES FOR MODELING EM COUPLING TO AIRCRAFT

| | COMPUTER CODE | LIGHTNING THREAT | EXTERNAL COUPLING $J_s$, $\sigma_s$ | INTERNAL COUPLING (APERTURES & DIFFUSION) | CABLE PENETRATION (SHIELDS & BUNDLES) | INTERNAL WIRING | PIN TRANSIENTS | EQUIPMENT UPSET |
|---|---|---|---|---|---|---|---|---|
| EXTERNAL MODELS | WIRANT | OK | OK | LIMITED SIZE | OK | N/A | N/A | N/A |
| | THREDE | OK | SHORT TIME | OK | N/A | N/A | N/A | N/A |
| | TRANSMISSION LINE | OK | OK | N/A | N/A | N/A | N/A | N/A |
| | GEMACS | OK | OK | OK | OK | OK | N/A | N/A |
| | T3DFD | OK | OK | OK | N/A | N/A | N/A | N/A |
| | NEC2 | OK | OK | N/A | N/A | N/A | N/A | N/A |
| | LPN | OK | OK | ESTIMATE | N/A | N/A | N/A | N/A |
| | TWTD | N/A | ESTIMATE RESONANCE | N/A | N/A | N/A | N/A | N/A |
| | RESISTOR | OK | LATE TIME | N/A | LATE TIME | N/A | N/A | N/A |
| | REDIST | OK | 2-D LIMITS | 2-D LIMITS | OK | OK | N/A | N/A |
| INTERNAL MODELS | SCEPTRE | N/A | ? | ? | OK | OK | OK | N/A |
| | SPICE | N/A | OK | N/A | OK | OK | OK | OK |
| | TRANSSIM | N/A | ? | ? | OK | OK | ? | N/A |
| | SINLINE | N/A | N/A | N/A | CABLE/WIRE | OK | ? | N/A |
| | TRAFFIC | N/A | N/A | N/A | CABLE/WIRE | OK | OK | N/A |
| PIN TRANSIENTS | HANAP | N/A | N/A | N/A | N/A | N/A | OK | OK |
| | TCAP | N/A | N/A | N/A | N/A | N/A | OK | OK |
| | CIRCUS | N/A | N/A | N/A | N/A | N/A | OK | OK |
| | SCORCH/ SUPERSAP | N/A | N/A | N/A | N/A | N/A | OK | N/A |

NEC is a method-of-moments code which uses segmentation by patches. The upper frequency limit is about 100 MHz. Both free space excitation and current injection can be modelled. This code requires a large amount of memory and can only be run on a mainframe computer.

GEMACS is a method-of-moments code which can also be run on an IBM-AT or mainframe computer. Predictions of external coupling to a model (segmented into a wire grid or patches), fields penetration due to aperture coupling, and the induced currents on a wire in a cavity behind the aperture can all be obtained. An example of the use of the code is given in Coffey and Hebert (1986). An entire session of the Applied Computational Electromagnetics Society (ACES) conference was devoted to the capabilities of this code (1988).

REDIST is a two-dimensional method-of-moments code which utilizes thin strips to form models. This code was developed on an IBM-XT compatible personal computer. It provides frequency domain currents and fields, both external and internal, and can be used for mixed material structures. Coupling to wiring can also be determined with this code. Limitations arise where the structural cross-section changes rapidly and when the structure thickness is greater than a skin depth of the material. Verification using scale models has provided good agreement (Geren, Melander, and Hall, 1987). Results have been compared to experimental data from a composite test bed aircraft (Cooley and Shortess, 1987).

THREDE is a three-dimensional finite difference computer code which runs on a mainframe computer. The primary limitation of this code for application to lightning is that most of the Low Frequency (LF) response is not included in the time domain response calculated by the code. Since a dominant coupling mechanism for composite structures is late time (after several microseconds) resistive drops in the structure, this code would not be suitable for modelling mixed material aircraft.

T3DFD is similar to THREDE, but allows mixed skin conductivities within the model.

TWTD utilizes a thin wire approximation and gives solutions in the time domain. It provides quick approximations of aircraft resonances. The code cannot treat current injection and so is not applicable to lightning.

LPN codes model physical structures with an equivalent circuit. This has the advantage of simple development and implementation. Much less memory is required for these simpler models. Currents can be determined by using a transmission line code. LPN theory may be applied to internal coupling provided calculations of the external coupling are available to be used as input to the internal model.

Transmission line models are useful to provide resonances and current densities. These models are usually simpler than models using a wire grid code. The SAE AE4L Orange Book and Simpson and Katzer (1988) give examples of the use of this type of code. Transmission line theory is applicable at all levels of coupling analysis.

Resistor models can be used to determine late time responses, due to structural IR drops, with fair accuracy. An example of this very simple model is given in section 6.2. Internal structure and cable bundles can be included. The resistance can be easily calculated by known physical parameters or measured values. Cooley and Shortess (1987) contains another example of this type of modelling.

TRAFFIC, SINLINE, SCEPTRE, TRANSSIM, and SPICE may be used in conjunction with results obtained from one of the codes described above which calculates external coupling to determine transients on internal wiring. TRAFFIC consists of a library of codes and is a part of PRESTO (see PRESTO User's Guide). It provides frequency domain output for wiring transients. SPICE and SCEPTRE are time domain codes and can thus include non-linear elements.

HANAP calculates pin transients in the frequency domain and design margins using a mainframe computer. Threshold Circuit Analysis Program (TCAP) can perform similar calculations for simple circuits on a hand calculator (Pryzby and Plumer, 1984). For non-linear analysis, CIRCUS, a code within the TRAFFIC library, will calculate desgin margins. SCORCH and its predecessor, SUPERSAP, may be available as a commercial product to calculate pin transients.

# 6. SUPPLEMENTAL EXAMPLES

## 6.1. Coupling to Wiring

An example of a simple two-dimensional model is used to illustrate coupling to wiring within a mixed materials structure. The model, shown in figure 6.1-1, is a box which has three metal sides and one G/E side. The conductivity and thickness of each side is indicated on the figure. The thicknesses used are for typical aircraft structure. The code RESIST was used to evaluate this model.

Numbers 1 through 6 inside the box indicate the shield locations evaluated. Each model run contained a single wire at a different location. The output was given as a transfer function of the shield current relative to the input current for the entire structure. Figure 6.1-2 contains the transfer functions for each shield location.

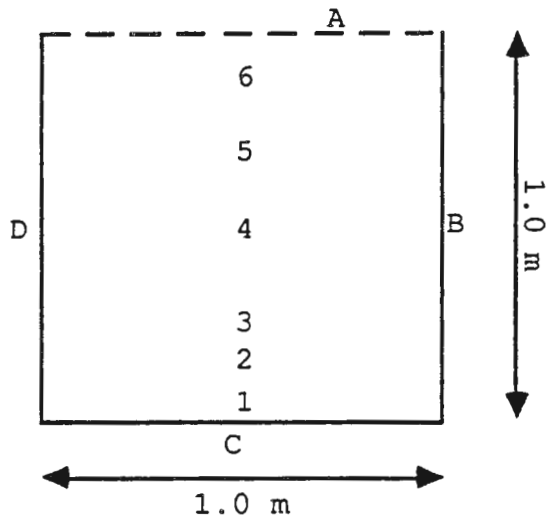
A worst case estimate of shield currents can be obtained by multiplying the calculated shield transfer function by the threat, in this case a double exponential lightning spectrum, and Fourier transforming the resultant spectrum to the time domain. Current pulse responses for each shield location are shown in figure 6.1-3. The peak currents vary by an order of magnitude. Induced shield currents are small for locations near the metal bottom of the box and increase steadily as the shield is moved toward the graphite lid.

These estimates of current represent worst case (upper bound) strikes, because the entire 200,000 A is assumed to flow through the box and shield. This would not be the case if the box were part of an aircraft. In order to obtain a more realistic estimate of internal shield currents, a vehicle external model must be analyzed to obtain the division of current between the modeled box and the remainder of the vehicle. This transfer function is then multiplied by the shield transfer function (calculated previously) and the lightning threat to obtain the shield current relative to the aircraft current.

Leakage through the graphite can be illustrated by looking at the magnetic fields calculated with this model. Figures 6.1-4 and 6.1-5 show magnetic field contour lines for the box at 1 kHz and 1 MHz. At 1 kHz, the G/E is transparent to the fields. The fields are excluded from the box at higher frequencies due to the skin effect. This indicates why early time responses for graphite or metal structures are similar.

After determining the shield currents, the current and voltage transients at the component level can be calculated using an appropriate method. A transmission line formulation is used here.

REDIST MODEL
METAL BOX WITH GRAPHITE LID




Shield locations denoted by numbers 1-6.

| WALL | STRUCTURAL PARAMETERS | |
|---|---|---|
| | THICKNESS (m) | CONDUCTIVITY (mhos/m) |
| A | 2.3E-3 | 1.5E4 |
| B | 1.3E-3 | 3.5E7 |
| C | 1.3E-3 | 3.5E7 |
| D | 1.3E-3 | 3.5E7 |
| SHIELD | 2.54E-4 | 3.5E7 |

FIGURE 6.1-1. REDIST EXAMPLE MODEL GEOMETRY

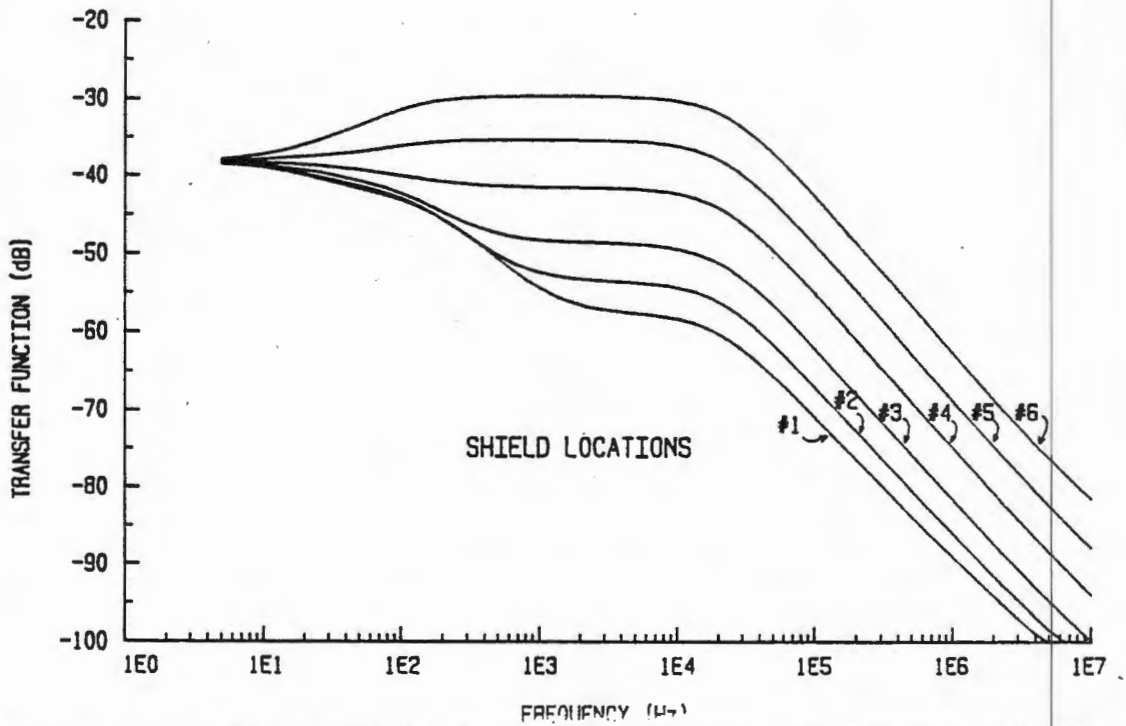



FIGURE 6.1-2. SHIELD TRANSFER FUNCTION FOR METAL/GE BOX FOR SEVERAL SHIELD LOCATIONS

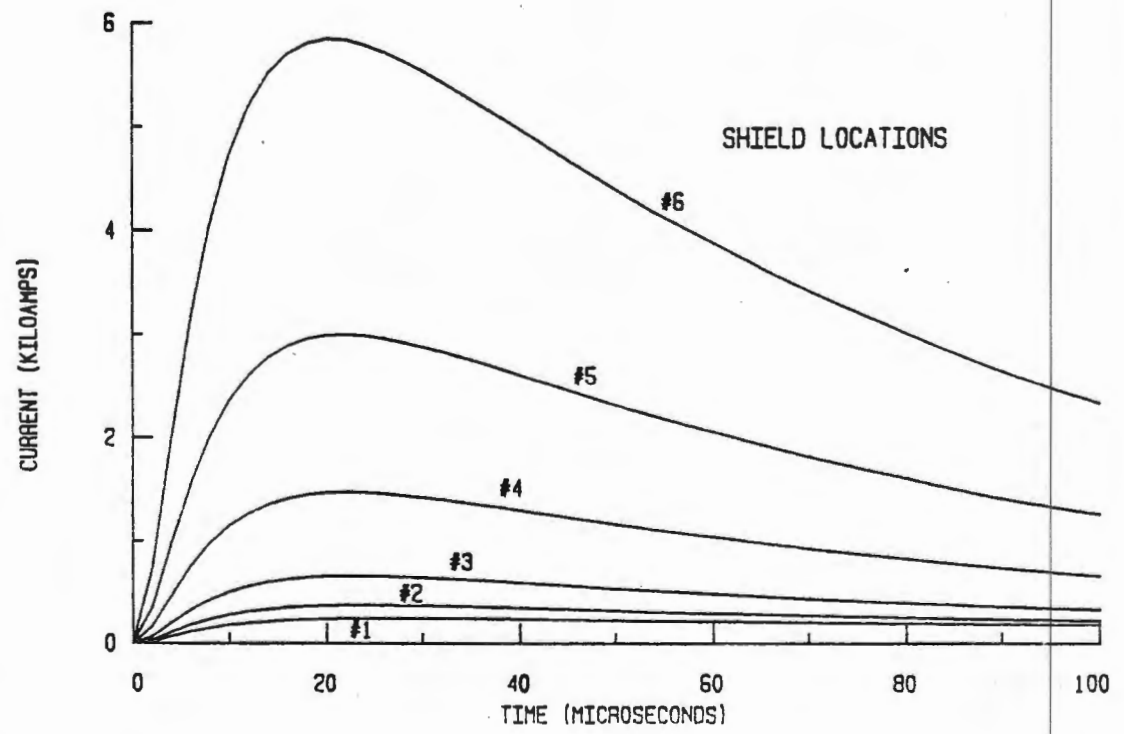



FIGURE 6.1-3. SHIELD CURRENT FOR METAL/GE BOX FOR SEVERAL SHIELD LOCATIONS FOR 200 kA ATTACHED LIGHTNING

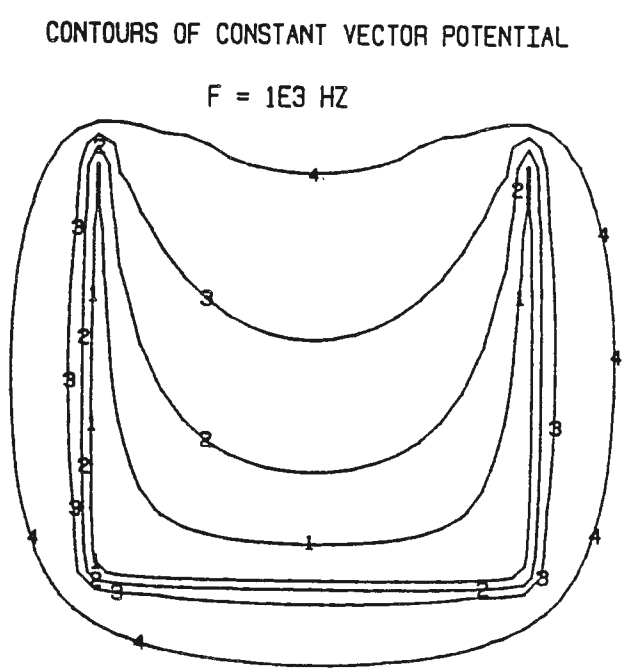



FIGURE 6.1-4. MAGNETIC FIELD CONTOURS FOR METAL/GE BOX AT 1 kHz

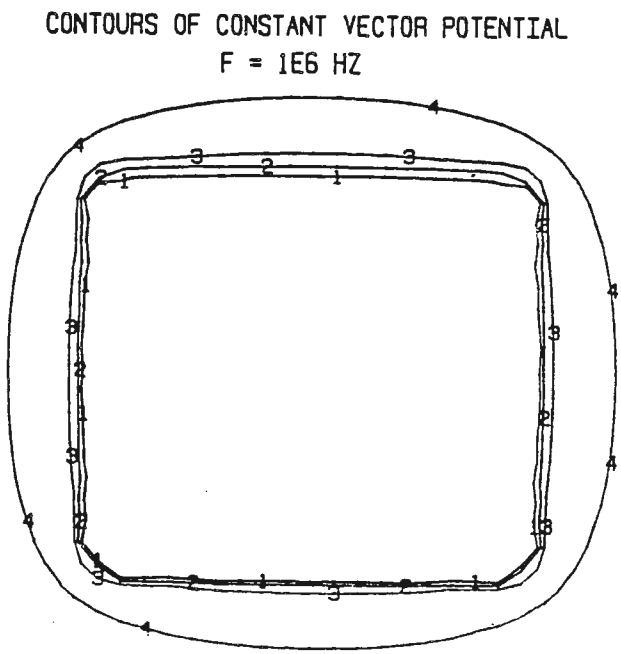



FIGURE 6.1-5. MAGNETIC FIELD CONTOURS FOR METAL/GE BOX AT 1 MHZ

The coupling to the shields is a continuous magnetic or electric field interaction along the length of the shield. This method of calculation uses a continuous voltage source term running the length of the transmission line model to the equipment terminations at each end. Basic assumptions made for this model are:

- The shield transfer function (voltage relative to current) is given by $R_{shield} + jw \times L_{shield}$

  where

  $R_{shield} = 1.78 \times 10^{-3}$ ohms per meter

  $L_{shield} = 8.96 \times 10^{-10}$ henrys per meter

  These numbers were obtained from measurements made on braided aluminum shields.

- The termination impedance was assumed to be 100 ohms at each end.

- The voltage and currents were calculated at one end of a transmission line with a continuous voltage source assumed equal to the shield current times the shield transfer function.

- The line length, L, was assumed to be 10 meters.

The results can be seen in figure 6.1-6. The time domain voltage waveforms for each of the six shield locations are shown. The peak voltages can be estimated using the following equation.

$$V_{peak} = V_{oc} / 2$$

$$= (I_{peak} \times R_{shield} \times L) / 2$$

where $V_{oc}$ is the open circuit voltage and $I_{peak}$ is the peak shield current.

For shield location #5 with a peak current of about 3 kA,

$$V_{peak} = (3 \text{ kA} \times 1.78 \times 10^{-3} \text{ ohms/m} \times 10 \text{ m}) / 2$$

$$= 26.7 \text{ V}$$

Larger voltages will be induced on longer cables and smaller voltages will be induced on shorter cables. This is due to resonance effects on the cables. Figure 6.1-7 illustrates the variation in induced voltages for several line lengths.

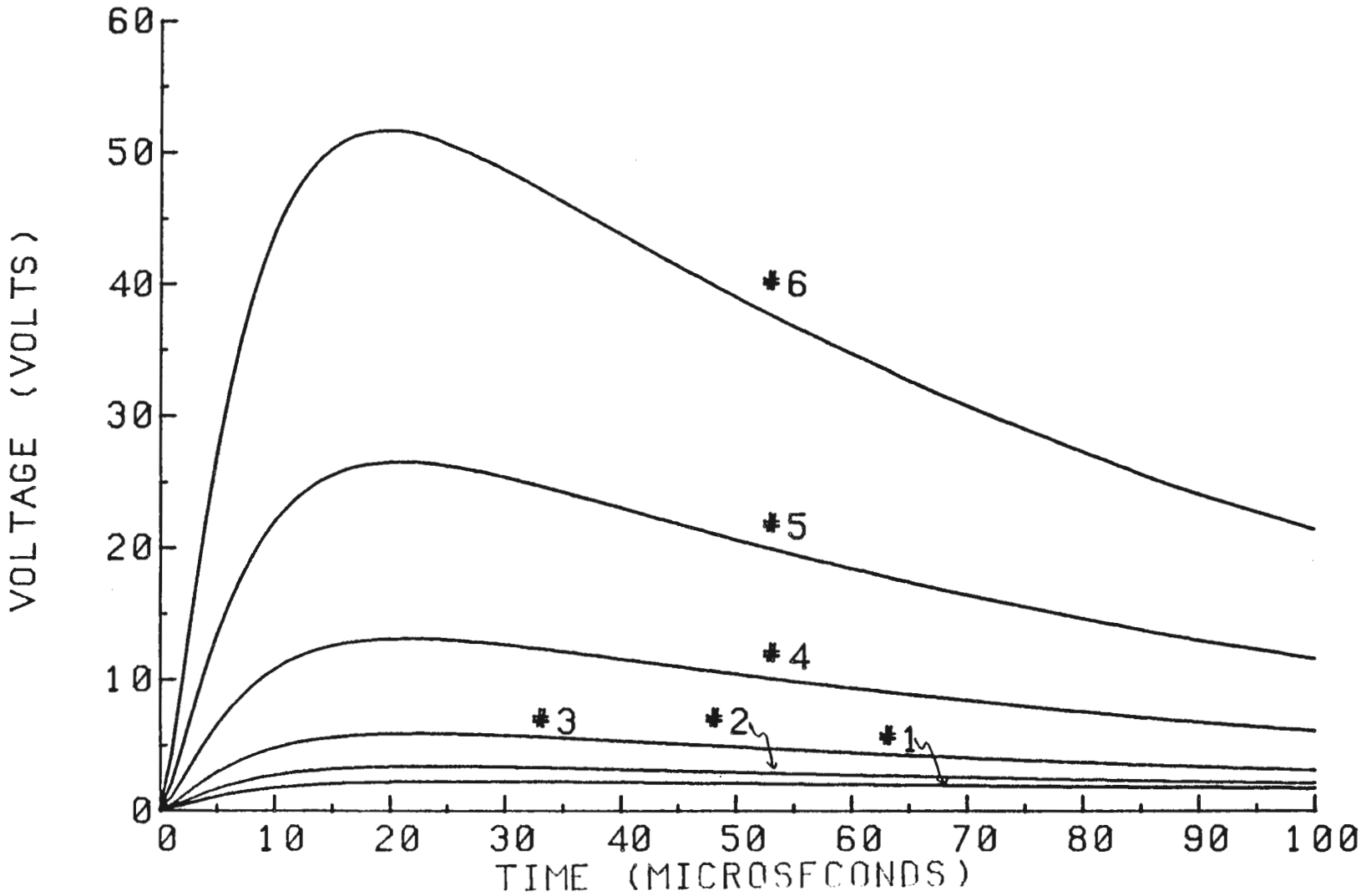



FIGURE 6.1-6. PIN VOLTAGE WAVEFORMS FOR SEVERAL SHIELD LOCATIONS

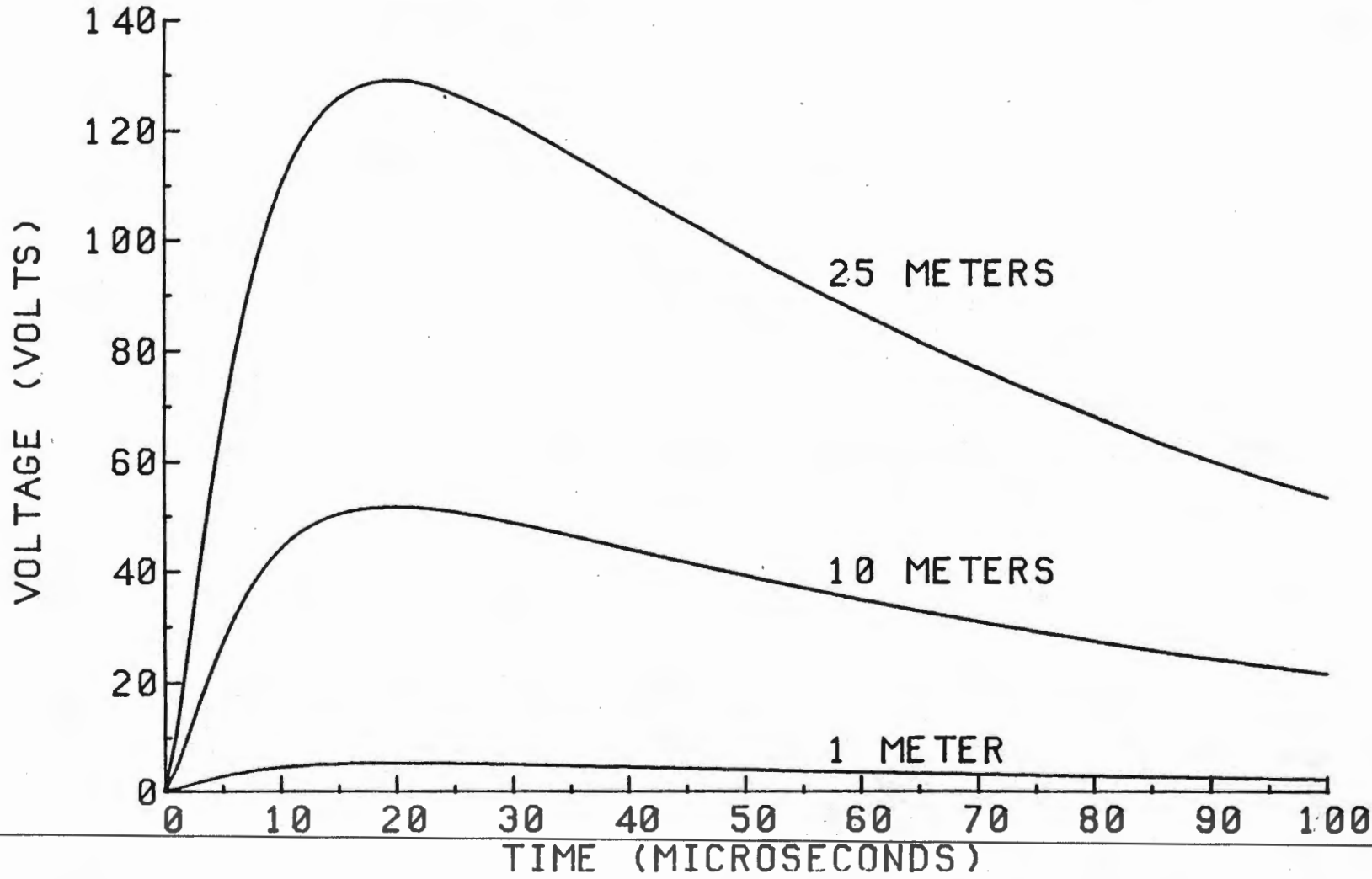



FIGURE 6.1-7. PIN VOLTAGE WAVEFORMS FOR SHIELD LOCATION #6 FOR VARIOUS LINE LENGTHS

## 6.2. Composite Wing Model

A dc resistance model of a composite wing used in NASA tests was constructed to determine the amount of current flowing through various sections of the wing. Results of this modeling study will predict energy flows into various structural members and fasteners. A dc model is appropriate for lightning studies since structural and fastener damage occurs because of the high energy levels. These high energy levels result primarily from large LF components of the lightning pulse.

An estimate of the dimensions and properties of the composite wing are listed in table 6.2-1. The spars were considered "I" beam sections 1/4 inch thick with the remaining values given in the table. The skin was taken as a product of average dimensions obtained from a NASA report, Pryzby and Plumer (1984). Both sides of the wing skin were considered in the total area. The braid running in the leading wing edge was assumed to be 1/2 inch braid which was taken to have 3 milliohm per meter resistance, as measured in previous studies. Fasteners were assumed to have a resistance of 10 milliohms each. These resistances are similar to predictions of various fastener resistances in Cooley (1985). The length of the wing is assumed to be 5.77 meters.

The structure resistance, $R_s$, is determined using the following equation.

$$R_s = L / (\sigma \times A)$$

where L is the length in meters, A is the area in square meters, and $\sigma$ is the conductivity in mhos (or siemens) per meter.

This simple resistance model predicts current level as percentages of total current in table 6.2-1. The predictions are obtained simply by resistive division between each item and the entire structure. These are compared to measured values for late time (80 microseconds) into the pulse. The agreement is good considering that dimensions, conductivities, number of fasteners, and fastener resistances are not accurately known.

TABLE 6.2-1. COMPOSITE WING RESISTANCE MODEL RESULTS VS. MEASUREMENTS

| ITEM | WIDTH (IN) | HEIGHT (IN) | AREA ($IN^2$) | CONDUCTIVITY (MHO/M) | STRUCTURE RESISTANCE (OHM) | TOTAL FASTENER RESISTANCE | STRUCTURE RESISTANCE W/FASTENERS | ITEM CURRENT (% OF TOTAL) | MEASURED CURRENT (3) (% OF TOTAL) |
|---|---|---|---|---|---|---|---|---|---|
| FRONT SPAR | 1.00 | 2.00 | 0.88 | 28000 | 0.363 | 0.005 (1) | 0.368 | 3.4 | 2% TO 5% |
| CENTER SPAR | 2.50 | 5.00 | 2.38 | 28000 | 0.134 | 0.005 (1) | 0.139 | 9.0 | 2% TO 5% |
| REAR SPAR | 1.50 | 3.00 | 1.38 | 28000 | 0.231 | 0.005 (1) | 0.236 | 5.3 | 2% TO 5% |
| SKIN | 77.00 | 0.0875 | 6.74 | 28000 | 0.047 | 0.001 (2) | 0.048 | 25.8 | 29% |
| BRAID | | | | | 0.017 | 0.005 (1) | 0.022 | 56.6 | 71% |

TOTAL RESISTANCE OF STRUCTURE = 0.012 OHM

FASTENER RESISTANCE ASSUMED = 0.01 OHM

(1) 2 FASTENERS ASSUMED
(2) 10 FASTENERS ASSUMED
(3) MEASURED DATA FROM PLUMER (1984)

# BIBLIOGRAPHY


AC 20-53 (Draft), Protection of Aircraft Electrical/Electronic Systems Against the Indirect Effects of Lightning, 8 June 1988.

Applied Computational Electromagnetics Conference Proceedings, Session V, Monterey, CA, March 1988.

Butters, W.G., et al., "Assessment of Lightning Simulation Test Techniques," Proceedings of IAGC on Lightning and Static Electricity, March 1982.

Clifford, D. W., K. E. Crouch, E. H. Schulte, "Lightning Simulation Testing," IEEE Transactions, EMC-24, V2, May 1982.

Coffey, E. L. and Lt. J. L. Hebert, "Implementation of GEMACS for Lightning Interaction Analysis," IAGC on Lightning and Static Electricity, June 1986.

Cooley, W. W., Determination of Electrical Properties of Grounding, Bonding, and Fastening Techniques for Composite Materials, Report No. DOT/FAA/CT-86/8, December 1985.

_____, and D. L. Shortess, Lightning Simulation Test Technique Evaluation, Report No. DOT/FAA/CT-87/38, November 1987.

Craft, W.L., Protection Optimization for Advanced Composite Structures, Grumman Aerospace Corporation, 1981.

Federal Aviation Administration - Florida Institute of Technology, Workshop on Grounding and Lightning Technology, March 1979.

Federal Aviation Administration - Georgia Institute of Technology, Workshop on Grounding and Lightning Protection, FAA-RD-78-83, May 1978.

Fisher, F. A. and J. A. Plumer, Lightning Protection of Aircraft, NASA Report RP-1008, 1977.

Geren, W. P., B. G. Melander, and D. L. Hall, "Lightning Current Redistribution," Applied Computational Electromagnetics Newsletter, Vol. 2, No. 1, May 1987.

IEEE Electromagnetic Compatibility Society, "Special Issue on Lightning and Its Interaction With Aircraft," IEEE Transactions on Electromagnetic Compatibility, Volume 24, No. 2, May 1982.

International Aerospace and Ground Conference on Lightning and Static Electricity, Dayton, OH, June 1986.

International Aerospace and Ground Conference on Lightning and Static Electricity, Fort Worth, Texas, June 1983.

International Aerospace and Ground Conference on Lightning and Static Electricity, Oklahoma City, OK, April 1988.

International Aerospace and Ground Conference on Lightning and Static Electricity, Orlando, FL, June 1984.

International Aerospace and Ground Conference on Lightning and Static Electricity, Oxford, FL, 1982.

Lightning Test Waveforms and Techniques for Aerospace Vehicles and Hardware, Report of Society of Automotive Engineers Committee AE4L, 20 June 1978.

Melander, B. G. and W. W. Cooley, Threat Environment Definition, D180-27423-1, The Boeing Company, February 1984.

MIL-B-5087B (ASG), Bonding, Electrical, and Lightning Protection, for Aerospace Systems, Amendment 2, 31 August 1970.

MIL-STD-1757, Lightning Qualification Test Techniques for Aerospace Vehicles and Hardware, 17 June 1980.

Mosher, J. and M. Wojtowicz, TCAP - Threshold Circuit Analysis Program Version 3.0 User's Guide (First Draft), TRW Report, August 1986.

Orange Book User's Manual - Lightning Interaction with Electronic Systems, SAE-AE4L Committee Report, October 1987.

PRESTO Digital Computer Code User's Guide - Volumes I to VI, DNA Report 3898F, October 1980.

Pryzby, J. E. and J. A. Plumer, Lightning Protection Guidelines and Test Data for Adhesively Bonded Aircraft Structures, NASA Contractor Report 3762, January 1984.

Riley, L. H., et al, "Lightning Tests of Pershing II," IAGC on Lightning and Static Electricity, June 1984.

Shortess, D. L., W. W. Cooley, and B. G. Melander, "Comparison of Four Lightning Simulation Tests on a Composite Test Bed Aircraft," IAGC on Lightning and Static Electricity, April 1988.

Simpson, M. M. and M. J. Katzer, "In-flight Aircraft Lightning Surface Current Model," IAGC on Lightning and Static Electricity, April 1988.

Sommer, D. L., Protection of Electrical Systems from Electromagnetic Hazards - Design Guide, Boeing Military Airplane Co., AFWAL-TR-81-2118, September 1981.

User's Manual for AC-20-53 - Protection of Aircraft Electrical/Electronic Systems Against the Indirect Effects of Lightning (Draft), September 1987.

Walen, D. B., "Lightning Tests on an All-Composite Helicopter," IAGC on Lightning and Static Electricity, April 1988.

Walko, L. C., A Test Technique for Measuring Lightning-Induced Voltages on Aircraft Electrical Circuits, NASA Report CR-2348, February 1974.

ACTION INTEGRAL. The action integral is a critical factor in the production of damage. It relates to the energy deposited or absorbed in a system. This energy cannot be defined without knowing the resistance of the system. The instantaneous power dissipated in a resistor is $I^2R$ and is expressed in watts. For the total energy expended, the power must be integrated over time to get the total joules, watt-seconds. By specifying the integral of $i(t)^2$ over the time interval involved, a useful quantity is defined for application to any resistance value. In the case of lightning, this quantity is defined as the action integral and is specified as $i(t)^2dt$ over the time the current flows.

ACTUAL TRANSIENT LEVEL. The actual transient level is the level of transients which actually appear at the system interfaces as a result of the external environment. This level may be less than or equal to the transient control level but should not be greater.

AIRCRAFT LIGHTNING INTERACTION. An encounter with lightning that produces sufficient current within or voltages along an aircraft skin or structure to pose a threat to the aircraft electrical/electronic systems, as a result of a direct lightning attachment.

ATTACHMENT POINT. A point of contact of the lightning flash with the aircraft.

CHARGE TRANSFER. The integral of the current over its entire duration, $i(t)dt$, in coulombs.

COMPONENT DAMAGE. Condition arising when the electrical characteristics of a circuit component are permanently altered beyond its specifications.

CORONA. A luminous discharge that occurs as a result of an electrical potential difference between the aircraft and the surrounding atmosphere.

CRITICAL. Functions whose failure would contribute to or cause a failure condition which would prevent the continued safe flight and landing of the aircraft.

DESIGN MARGIN. The difference between the equipment transient design levels and the transient control level.

DIRECT EFFECTS. Any physical damage to the aircraft or onboard systems due to the direct attachment of the lightning channel. This includes tearing, bending, burning, vaporization, or blasting of aircraft surfaces or structures, and damage to electrical/electronic systems.

EQUIPMENT TRANSIENT DESIGN LEVEL. The level of transients which the equipment is qualified to withstand.

EQUIPMENT TRANSIENT SUSCEPTIBILITY LEVEL. The transient level which will result in damage or upset to the system components. This level will be greater than the equipment transient design level.

EXTERNAL ENVIRONMENT. Characterization of the natural lightning environment with idealized waveforms for engineering purposes.

INDUCED VOLTAGES. A voltage produced around a closed path or circuit by changing magnetic or electric fields or structural IR voltages.

INDIRECT EFFECTS. Voltage and/or current transients induced by lightning in aircraft electrical wiring which can produce upset and/or damage to components within electrical/electronic systems.

INTERNAL ENVIRONMENT. The fields and structural IR potentials produced by the external environment, along with the voltages and currents induced by them.

LIGHTNING FLASH. The total lightning event in which charge is transferred from one charge center to another. It may occur within a cloud, between clouds, or between a cloud and the ground. It can consist of one or more strokes, plus intermediate or continuing currents.

LIGHTING LEADER STROKE. The leader forms an ionized path for charge to be channeled towards the opposite charge center. The stepped leader travels in a series of short, luminous steps prior to the first return stroke. The dart leader reionizes the return stroke path in one luminous step prior to each subsequent return stroke in the lightning strike.

LIGHTNING STRIKE. Any attachment of the lightning flash to the aircraft.

LIGHTNING STRIKE ZONES. Locations on the aircraft where the lightning flash will attach or where substantial amounts of electrical current may be conducted between attachment points. The location of these zones on any aircraft is dependent on the aircraft's geometry and operational factors and often varies from one aircraft to another.

LIGHTNING RETURN STROKE. A lightning current surge that occurs when the lightning leader makes contact with the ground or an opposite charge center.

MULTIPLE BURST. A randomly spaced series of bursts of short duration, low amplitude current pulses, with each pulse characterized by rapidly changing currents. These bursts may result from lightning leader progression or branching and may be accompanied by or superimposed on stroke or continuing currents. The multiple bursts appear to be most intense at the time of initial leader attachment to the aircraft.

MULTIPLE STRIKE. Two or more lightning strikes during a single flight.

MULTIPLE STROKE. Two or more return strokes occurring during a single lightning flash.

PEAK RATE OF RISE. The maximum instantaneous slope of the waveform as it rises to its maximum value. Mathematically, the peak rate of rise of a function, i(t), may be expressed as the maximum of d[i(t)]/dt.

RETURN STROKE. See lightning return stroke.

STRUCTURAL IR VOLTAGE. The portion of the induced voltage resulting from the product of the distributed lightning current, I, flowing through the resistance, R, of the aircraft skin or structure.

SWEPT STROKE. A series of successive attachments due to sweeping of the flash across the surface of the airplane by the motion of the airplane.

SYSTEM FUNCTIONAL UPSET. Impairment of system operation, whether permanent or momentary (e.g., a change of digital or analog state) which may or may not require manual reset.

TRANSIENT CONTROL LEVEL. The maximum allowable level of transients appearing at the systems interfaces as a result of the defined external environment.

TRIBOELECTRIC CHARGING. Static electricity produced on a structure from the effects of friction.

UPSET. See system functional upset.

## LIST OF ACRONYMS

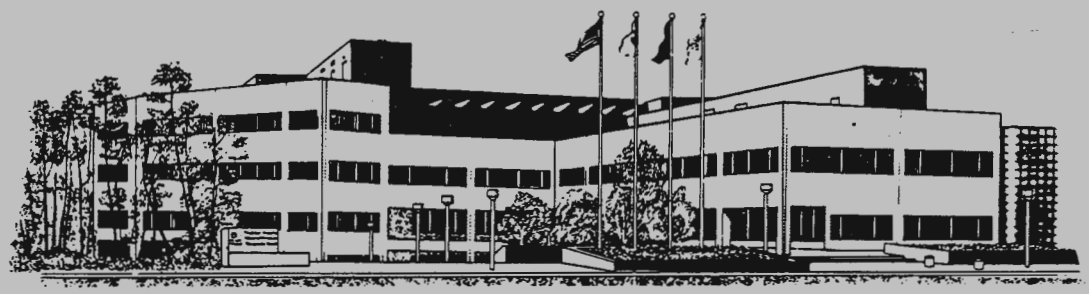
| | |
|---|---|
| ACAP | Advanced Composite Airframe Program |
| ACES | Applied Computational Electromagnetics Society |
| AEHP | Atmospheric Electricity Hazards Protection |
| AE4L | SAE Subcommittee (Lightning) |
| AFFDL | Air Force Flight Dynamics Laboratory |
| AFWAL | Air Force Wright Aeronautical Laboratories |
| ALCM | Air Launched Cruise Missile |
| B-dot | Derivative of the magnetic field with respect to time |
| CW | Continuous Wave |
| DNA | Defense Nuclear Agency |
| DOE | Department of Energy |
| E-dot | Derivative of the electric field with respect to time |
| EM | Electromagnetic |
| EMC | Electromagnetic Compatibility |
| EMI | Electromagnetic Interference |
| EMP | Electromagnetic Pulse |
| ETDL | Equipment Transient Design Level |
| FAA | Federal Aviation Administration |
| G/E | Graphite Epoxy |
| GEMACS | General Electromagnetic Model for the Analysis of Complex Systems |
| HF | High Frequency |
| I | Current |
| I-dot | Derivative of the current with respect to time |
| kA | Kiloampere |
| LF | Low Frequency |
| LPN | Lumped Parameter Network |
| LRU | Line Replaceable Unit |
| LTRI | Lightning and Transients Research Institute |
| NASA | National Aeronautics and Space Administration |
| NASC | Naval Air Systems Command |
| NEC | Numerical Electromagnetics Code |
| NSWC | Naval Surface Weapons Center |
| R | Resistance |
| RAE | Royal Aircraft Establishment |
| RF | Radio Frequency |
| RL | Resistance/Inductance |
| RLC | Resistance/Inductance/Capacitance |
| RLCM | Resistance/Inductance/Capacitance/Mutual |
| SAE | Society of Automotive Engineers |
| TCAP | Threshold Circuit Analysis Program |
| TCL | Transient Control Level |
| TTL | Transistor-Transistor Logic |
| TWTD | Thin Wire Time Domain |
| UHF | Ultra-high Frequency |
| UK | United Kingdom |
| VHF | Very High Frequency |

Department of Transportation
Federal Aviation Administration

DOT/FAA/CT-88/10

# HANDBOOK-VOLUME II
# DIGITAL SYSTEMS VALIDATION

## CHAPTER 14
## HIGH ENERGY RADIO FREQUENCY FIELDS
## (IMPACT ON DIGITAL SYSTEMS)



PREPARED BY:

COMPUTER RESOURCE MANAGEMENT, INC.
950 HERNDON PARKWAY, SUITE 360
HERNDON, VIRGINIA 22070

PREPARED FOR:

FEDERAL AVIATION ADMINISTRATION
TECHNICAL CENTER
ATLANTIC CITY INTERNATIONAL AIRPORT, NEW JERSEY 08405

NOTICE

This document is disseminated under the sponsorship of the U.S. Department of Transportation in the interest of information exchange. The United States Government assumes no liability for the contents or use thereof.

The United States Government does not endorse products or manufacturers. Trade or manufacturers' names appear herein solely because they are considered essential to the objective of this report.

TABLE OF CONTENTS


| Section | | Page |
|---|---|---|
| 1. | INTRODUCTION | 14-1 |
| | 1.1. Background | 14-1 |
| | 1.2. Objectives and Scope | 14-2 |
| 2. | ELECTROMAGNETIC ENVIRONMENT SURVEY AND ANALYSIS - UNITED STATES | 14-5 |
| | 2.1. Guidelines | 14-5 |
| | 2.2. Results | 14-5 |
| | 2.3. Airport Environment | 14-7 |
| | 2.4. Air-to-Air Environment | 14-10 |
| | 2.5. Ground Environment | 14-10 |
| | 2.6. Shipboard Environment | 14-14 |
| | 2.7. Composite of United States Emitters | 14-16 |
| | 2.8. Composite of United States Compared to NATO STANAG 3614AE | 14-16 |
| | 2.9. Takeoff and Landing Scenario | 14-16 |
| 3. | EME SURVEY AND ANALYSIS - WORLDWIDE | 14-21 |
| | 3.1. Guidelines | 14-21 |
| | 3.2. Results | 14-21 |
| 4. | COMMENTS | 14-25 |
| APPENDIX | | |
| | A - GUIDELINES FOR DEVELOPING MAXIMUM PEAK AND AVERAGE FIELD STRENGTH ENVELOPE GRAPHS FOR AIRCRAFT | 14-27 |
| BIBLIOGRAPHY | | 14-49 |
| LIST OF ACRONYMS | | 14-50 |

LIST OF ILLUSTRATIONS


| Figure | | Page |
|---|---|---|
| 1.2-1 | INDUCED RADIO FREQUENCY THREAT | 14-4 |
| 2.3-1 | PREDICTED MAXIMUM PEAK FIELD STRENGTH LEVELS FROM CIVILIAN AIRPORT FIXED EMITTERS | 14-8 |
| 2.3-2 | PREDICTED MAXIMUM AVERAGE FIELD STRENGTH LEVELS FROM CIVILIAN AIRPORT FIXED EMITTERS | 14-8 |
| 2.3-3 | PREDICTED MAXIMUM PEAK FIELD STRENGTH LEVELS FROM CIVILIAN AIRPORT MOBILE EMITTERS | 14-9 |
| 2.3-4 | PREDICTED MAXIMUM AVERAGE FIELD STRENGTH LEVELS FROM CIVILIAN AIRPORT MOBILE EMITTERS | 14-10 |
| 2.4-1 | PREDICTED MAXIMUM PEAK FIELD STRENGTH LEVELS FROM INTERCEPTOR AIRCRAFT EMITTERS | 14-11 |
| 2.4-2 | PREDICTED MAXIMUM AVERAGE FIELD STRENGTH LEVELS FROM INTERCEPTOR AIRCRAFT EMITTERS | 14-11 |
| 2.4-3 | PREDICTED MAXIMUM PEAK FIELD STRENGTH LEVELS FROM MILITARY NON-INTERCEPTOR AIRCRAFT | 14-12 |
| 2.4-4 | PREDICTED MAXIMUM AVERAGE FIELD STRENGTH LEVELS FROM MILITARY NON-INTERCEPTOR AIRCRAFT | 14-12 |
| 2.5-1 | PREDICTED MAXIMUM PEAK FIELD STRENGTH LEVELS FROM GROUND EMITTERS | 14-13 |
| 2.5-2 | PREDICTED MAXIMUM AVERAGE FIELD STRENGTH LEVELS FROM GROUND EMITTERS | 14-14 |
| 2.6-1 | PREDICTED MAXIMUM PEAK FIELD STRENGTH LEVELS FROM SHIPBOARD EMITTERS | 14-15 |
| 2.6-2 | PREDICTED MAXIMUM AVERAGE FIELD STRENGTH LEVELS FROM SHIPBOARD EMITTERS | 14-15 |
| 2.7-1 | PREDICTED MAXIMUM PEAK FIELD STRENGTH LEVELS COMPOSITE OF UNITED STATES EMITTERS | 14-17 |
| 2.7-2 | PREDICTED MAXIMUM AVERAGE FIELD STRENGTH LEVELS COMPOSITE OF UNITED STATES EMITTERS | 14-17 |
| 2.8-1 | PREDICTED MAXIMUM AVERAGE FIELD STRENGTH LEVELS UNITED STATES COMPOSITE VS NATO STANAG 3614AE | 14-18 |
| 2.9-1 | CRITERIA FOR SUSCEPTIBILITY THRESHOLD | 14-18 |
| 2.9-2 | PREDICTED MAXIMUM PEAK FIELD STRENGTH LEVELS COMPARISON UNITED STATES COMPOSITE WITH SIX AIRPORT COMPOSITE | 14-20 |
| 2.9-3 | PREDICTED MAXIMUM AVERAGE FIELD STRENGTH LEVELS COMPARISON UNITED STATES COMPOSITE WITH SIX AIRPORT COMPOSITE | 14-20 |
| 3.2-1 | PREDICTED MAXIMUM PEAK FIELD STRENGTH LEVELS COMPARISON OF UNITED STATES COMPOSITE WITH UNITED KINGDOM AND FRANCE COMPOSITE | 14-22 |
| 3.2-2 | PREDICTED MAXIMUM AVERAGE FIELD STRENGTH LEVELS COMPARISON OF UNITED STATES COMPOSITE WITH UNITED KINGDOM AND FRANCE COMPOSITE | 14-22 |
| 3.2-3 | PREDICTED MAXIMUM PEAK FIELD STRENGTH LEVELS WORLDWIDE WORST-CASE ENVIRONMENT | 14-23 |

LIST OF ILLUSTRATIONS (Continued)


| Figure | | Page |
|---|---|---|
| 3.2-4 | PREDICTED MAXIMUM AVERAGE FIELD STRENGTH LEVELS WORLDWIDE WORST-CASE ENVIRONMENT | 14-23 |
| 3.2-5 | PREDICTED MAXIMUM PEAK FIELD STRENGTH LEVELS COMPOSITE OF INTERNATIONAL ENVIRONMENTS | 14-24 |
| 3.2-6 | PREDICTED MAXIMUM AVERAGE FIELD STRENGTH LEVELS COMPOSITE OF INTERNATIONAL ENVIRONMENTS | 14-24 |
| A-1 | FAA CRITERIA FOR A ONE-ENGINE OUT SITUATION FOR A TWO-ENGINE AIRPLANE | 14-33 |
| A-2 | EXAMPLE OF ELECTRIC FIELD STRENGTH VERSUS FREQUENCY GRAPH ON A LOG-LOG SCALE | 14-43 |
| A-3 | EXAMPLE OF ENVELOPE OF FIELD STRENGTH VERSUS FREQUENCY GRAPH | 14-44 |
| A-4 | EXAMPLE OF FINAL OUTPUT | 14-45 |
| A-5 | EXAMPLE OF ENVELOPE GRAPH OF A SECOND ENVIRONMENT, TO BE COMBINED WITH FIGURE A-4 TO FORM A COMPOSITE GRAPH | 14-46 |
| A-6 | COMPOSITE OF TWO ENVIRONMENTS SHOWN IN FIGURES A-4 AND A-5 | 14-47 |


LIST OF TABLES


| Table | | Page |
|---|---|---|
| 2.2-1 | FIXED AIRPORT EMITTER | 14-6 |
| 2.3-1 | FIXED AIRPORT ENVIRONMENT | 14-7 |
| 2.3-2 | MOBILE AIRPORT ENVIRONMENT | 14-9 |
| 2.5-1 | GROUND EQUIPMENT | 14-13 |
| 2.6-1 | SHIPBOARD EQUIPMENT | 14-14 |
| A-1 | SUMMARY OF MINIMUM DISTANCES FOR THE WORST-CASE SCENARIO | 14-29 |
| A-2 | TYPICAL FIXED AIRPORT ENVIRONMENTAL EMITTERS | 14-30 |
| A-3 | NEAR-FIELD CORRECTION FACTOR FOR RECTANGULAR APERTURE ANTENNAS | 14-35 |
| A-4 | NEAR-FIELD CORRECTION FACTOR FOR CIRCULAR APERTURE ANTENNAS | 14-37 |
| A-5 | BEAMWIDTHS AND GAIN FACTORS OF RECTANGULAR ANTENNAS WITH VARIOUS ILLUMINATIONS | 14-40 |
| A-6 | BEAMWIDTHS AND GAIN FACTORS OF CIRCULAR ANTENNAS WITH VARIOUS ILLUMINATIONS | 14-40 |
| A-7 | SAMPLE ELECTRIC FIELD STRENGTH LEVELS USING VALUES IN FIGURE A-4 | 14-48 |

# 1. INTRODUCTION

## 1.1. Background

The introduction of a new generation of aircraft (fixed wing, rotary wing, and powered lift) advanced technologies, composite structures, and sophisticated high-integrity, integrated, software-based, digital flight control and avionic systems has confronted the aviation community with numerous flight safety issues and concerns. These new aircraft are making use of fly-by-wire flight control systems in which direct mechanical or hydraulic linkages are being replaced by solid-state digital electronic systems controlling engines, fuel systems, and electrohydraulic/electromechanical actuators. Previously, flight control and avionic systems utilized analog technology which was not responsive to transient disturbances unless they exceeded the analog device damage level. Unlike their analog predecessors, digital electronics are very susceptible to transient effects, as well as to discrete-frequency radiation. Digital device performance can be adversely affected (upset) before the device damage level is reached.

A number of European military aircraft fatal accidents have been attributed to High Energy Radio Frequency (HERF). A digital fly-by-wire military Toronado aircraft and crew were lost during a tactical training strafing attack in Germany. The loss was attributed to HERF when the aircraft flew through a high intensity Radio Frequency (RF) field. The civil/military aviation industry has very limited experience or data directed to accidents caused by electromagnetic transients and/or radiation. The present criteria, specifications, and procedures are being reevaluated. The HERF fields apparently upset the digital flight control system of the Toronado which was qualified to a very low Electromagnetic Environment (EME) standard.

While composite materials may offer significant advantages in strength, weight, and cost, they provide less electromagnetic shielding than aluminum. The use of solid-state digital technology in flight critical systems creates major challenges to prevent transient susceptibility and upset in both civil and military aircraft. Therefore, the Civil Aviation Authority (CAA), United Kingdom (U.K.) and the Federal Aviation Administration (FAA), United States (U.S.) voiced concern relative to emerging technology aircraft and systems.

The present criteria, specifications, and procedures appear to be inadequate for industry and the certification authorities for three reasons: (1) the changes in aircraft construction from aluminum to composites, (2) the increased criticality of software-based digital flight control and avionic systems, and (3) the increased levels of RF energy over a wide range of frequencies. The CAA and FAA therefore decided that the impact of each of these areas on the total aircraft RF susceptibility hardening and protection should be assessed.

## 1.2. Objectives and Scope

Currently, the U.S. civil aircraft manufacturers use the electromagnetic susceptibility guidelines as referenced in the Radio Technical Commission for Aeronautics (RTCA) document RTCA/DO-160B, section 20. Category Z equipment qualified to this standard is demonstrated to perform satisfactorily when subjected to a 1 Volt per meter (V/m) electric field intensity for frequencies between 35 MHz and 1215 MHz (a 2 V/m standard is imposed for the Very High Frequency (VHF) communication band from 118 to 136 MHz). A similar standard is used for frequencies between 15 kHz and 35 MHz but is specified in different terms, with a slightly higher standard for the High Frequency (HF) band between 2 and 30 MHz. The European Toronado aircraft was designed and protected according to a similar specification.

The U.S. military qualification standards for RF susceptibility are contained in MIL-STD-461B, section 19. This standard calls for an electric field strength 5 to 20 times greater than that called for in the RTCA document. It also extends the frequency range to at least 10,000 MHz and possibly to 40,000 MHz. This document implies that an electric field of at least 200 V/m may be encountered at specific frequencies which are specified by the procuring command. Testing methods are described in MIL-STD-462. The U.S. Navy, in qualifying aircraft for a carrier flight deck, has defined high electric field intensities greatly exceeding the values of MIL-STD-461B. A number of current military aircraft have encountered Electromagnetic Compatibility (EMC) problems, including the UH-60 helicopters. EME considerations for equipment design and procurement are described in MIL-STD-235-1A.

There have been no recognizable problems reported on civil transport airplanes to date. However, fly-by-wire technology is in use today on some Boeing Model 757-200 and 747-400 airplanes, on the Airbus Industrie A-320, Pratt and Whitney 4000° engines, and other U.S. civil aircraft (fixed and rotary wing). In addition, civil and military transmitter systems are continually being upgraded to operate at higher frequencies and to produce higher effective electric field intensities. The Spectrum Engineering Division of the FAA Technical Center and the Aircraft Engineering Division in the Office of Airworthiness initiated a research effort with the Department of Defense (DOD), Electromagnetic Compatibility Analysis Center (ECAC), Annapolis, Maryland, to survey the Contiguous United States (CONUS) and its possessions for civil and military ground, airborne, and shipborne HERF sources. Simultaneously, a solicitation for the CAA and its European counterparts to aggressively conduct a similar survey and analysis was requested. Following the surveys by ECAC, a validation phase (airborne measurements) of selected transmitters and one airport was conducted by Ohio University, Athens, Ohio.

United States research initiatives include:

- Conducting research into worldwide civil and military ground, airborne, and shipborne emitter radiation which may be a threat to emerging aircraft and systems.

- Sharing knowledge of each country's (i.e., United States, United Kingdom, and France) civil and military high powered transmitters.

- Developing the best approach to transfer this information to the airframe and systems manufacturers, RTCA, Society of Automotive Engineers (SAE), European Organization for Civil Aviation Electronics (EUROCAE), International Civil Aviation Organization (ICAO), etc.

- Developing an international susceptibility standard.

- Investigating possible regulatory actions (i.e., Notice of Proposed Rulemaking (NPRM), Advisory Circular (AC), User's Manual, etc.).

- Considering the airworthiness, certification, and operational issues and implications.

In addition to these initiatives, the Flight Safety Research Branch at the FAA Technical Center is participating in an Electromagnetic Radiation (EMR) measurement test being conducted by the U.S. Air Force Special Missions Operational Test and Evaluation Center (SMOTEC), Hurlburt Field, Florida. The primary purpose of this test is to collect EMR data at selected locations to enable safe air transport of munitions. Experience has shown that when an electromagnetic field impinges on a munition, it can fire the electroexplosive devices, destroy transistors, or cause electronic circuits to malfunction. Examples of the multitude of threats considered in this chapter are illustrated in figure 1.2-1.

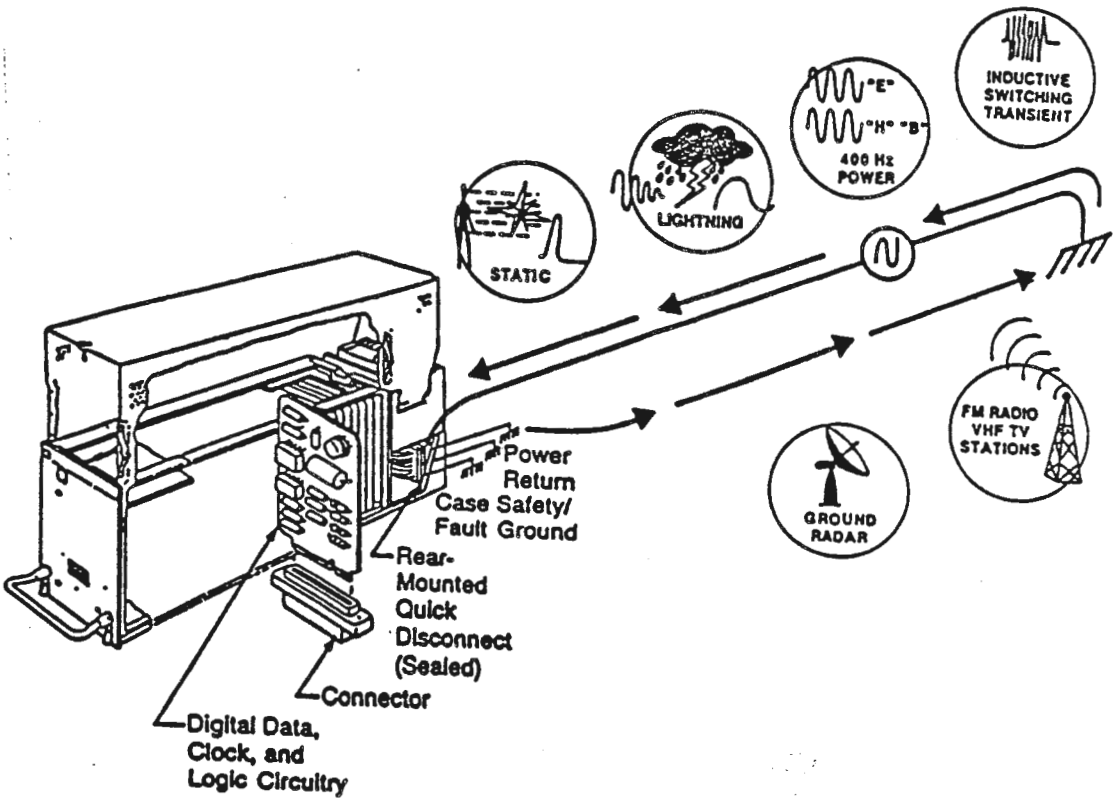



FIGURE 1.2-1. INDUCED RADIO FREQUENCY THREAT

## 2. ELECTROMAGNETIC ENVIRONMENT SURVEY AND ANALYSIS - UNITED STATES

### 2.1. Guidelines

In an effort to standardize prediction techniques, the FAA and ECAC documented the techniques used in the form of guidelines (DOD/ECAC, 1987) for predicting maximum EMR levels at aircraft exposed to high-powered emitters. Those guidelines were used in the calculation of maximum peak and average electric field strengths at the aircraft as a result of ground, airborne, and shipborne RF emitters. The guidelines for calculating peak and average field strengths are presented as step-by-step instructions starting with the data on the emitters in the EME and ending with an envelope graph representing maximum field strengths. The methods for calculating field strengths in the worst-case scenario and in the takeoff and landing scenario are defined and discussed. Simplifying assumptions are made and, where appropriate, justifications are shown. The minimum distances used in the U.S. environmental analysis are presented. The results obtained by following these guidelines will be peak and average field strengths, at worst case distances between emitters in the EME and aircraft. The total survey and environment analysis is described as the airport, air-to-air, ground, and shipborne environment.

Scenarios for the survey and analysis guidelines are as follows:

- A worst-case civil or military emitter, and for comparison, a takeoff or landing at specific airports.
- A worst-case considering the maximum EMR levels at an aircraft exposed to any ground, airborne, and shipborne emitter in the United States.
- The EME that would be experienced by an aircraft on takeoff and landing at six major U.S. airports.
- Integration and comparison of the worst-case emitters from the United Kingdom, France, and other countries as their information becomes available.
- Integration and comparison of the American, British, and French airport data.

### 2.2. Results

The results are presented in peak and average field strengths for the aircraft emitter sources surrounding airport, air-to-air, ground, and shipboard areas. The survey would review the emitters from 10 kHz to 40 GHz, identify worst-case conditions, and plot survey and study results in RF envelope histograms. To make the data manageable and unclassified, the granularity of the curve has been smoothed.

The following assumptions were made to simplify the task and the histograms:

- Specification of minimum distances.
- Continuous illumination.
- Mainbeam gain.
- Modulation independence.
- Constructive ground reflections at HF.
- Noncumulative field strengths.
- Near-field correction.

A typical analysis of a specific emitter considered the specifications of each, as indicated in table 2.2-1.

TABLE 2.2-1. FIXED AIRPORT EMITTER

| HEIGHT - FINDING RADAR | |
|---|---|
| Operating Frequency | 2700 MHz |
| Peak/Average Output Power | 3.5 MW/2.9 kW |
| Antenna Size | 8 ft x 30 ft |
| Antenna Gain | 39 dBi |
| Distance | 500 ft |
| **ELECTRIC - FIELD STRENGTH** | |
| Uncorrected Peak | 6050 V/m |
| Uncorrected Average | 175 V/m |
| Near Field | 700 ft |
| Near Field Correction Factor | 3.5 dB |
| Corrected Peak | 4100 V/m |
| Corrected Average | 120 V/m |

## 2.3. Airport Environment

The airport environment consists of fixed and mobile emitters located on the airport. The airport is defined by a five nautical mile (nmi) radius from the geographic center of the airport. Minimum distance calculations are for airport fixed emitters and mobile emitters. By regulations, the minimum distance between a fixed emitter and the taxiway or runway is 250 feet (ft). Table 2.3-1 identifies some fixed emitters at the airport.

TABLE 2.3-1. FIXED AIRPORT ENVIRONMENT

| |
|---|
| Marker Beacon |
| Localizer |
| VOR |
| Glide Slope |
| Ground Controlled Approach Radar |
| Distance Measuring Equipment |
| TACAN |
| Microwave Landing System |
| Airport Surface Detection System |
| Non-Directional Beacon |
| Airport Surveillance Radar |
| Air Route Surveillance Radar |
| Weather Radar |
| ATCRBS Interrogator |
| VHF and UHF Communications/Telemetry |

For fixed airport emitters above, predicted maximum peak and average field strengths were calculated and plotted as in figures 2.3-1 and 2.3-2.

The mobile emitter environment consists of all emitters that are not in a fixed location within a minimum distance of 50 ft. These include those emitters on ground support vehicles as well as those in aircraft. Aircraft emitters may operate while on the ground and taxiway, and prior to takeoff. Table 2.3-2 outlines those mobile emitters.

For mobile airport emitters above, predicted maximum peak and average field strengths were calculated and plotted as in figures 2.3-3 and 2.3-4.

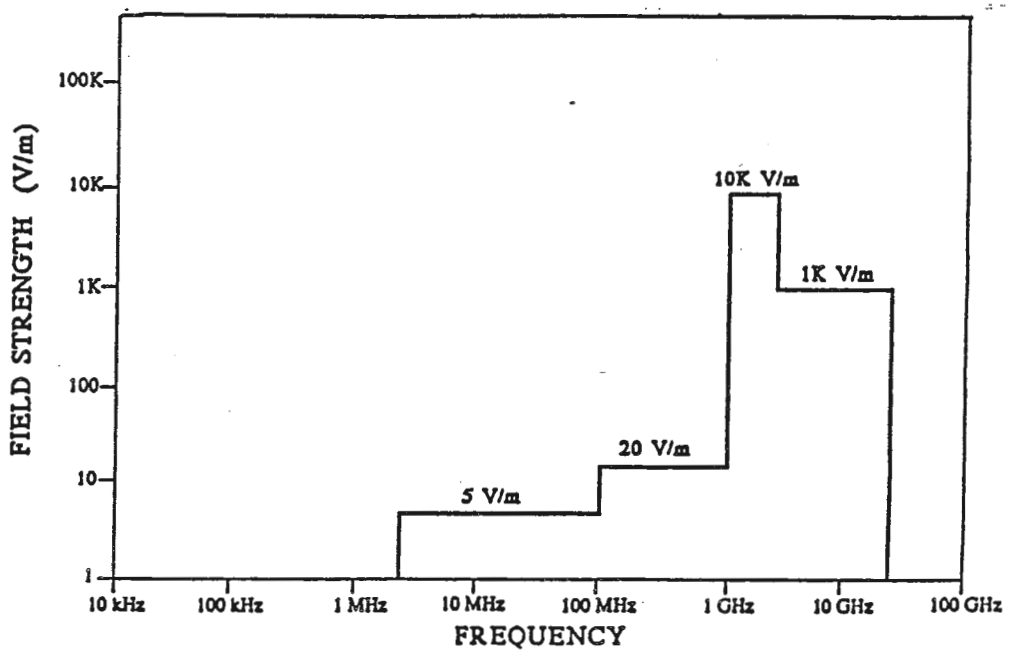



FIGURE 2.3-1. PREDICTED MAXIMUM PEAK FIELD STRENGTH LEVELS FROM CIVILIAN AIRPORT FIXED EMITTERS

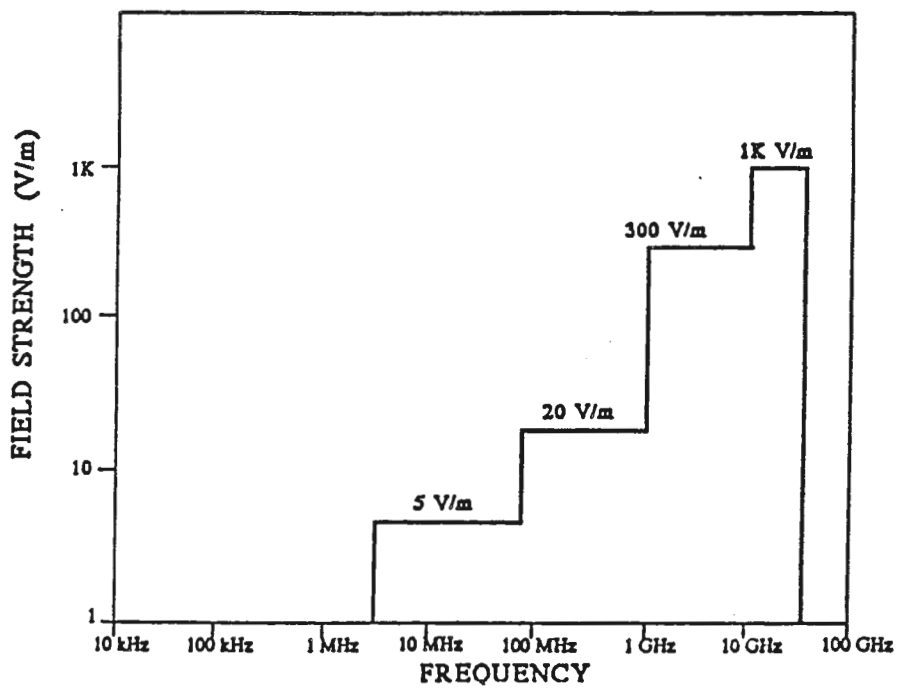



FIGURE 2.3-2. PREDICTED MAXIMUM AVERAGE FIELD STRENGTH LEVELS FROM CIVILIAN AIRPORT FIXED EMITTERS

TABLE 2.3-2. MOBILE AIRPORT ENVIRONMENT

| |
|---|
| HF/VHF/UHF Communications |
| Tacan |
| Doppler Navigational Radar |
| Radio Altimeter |
| Weather Radar |
| ATCRBS Beacon |

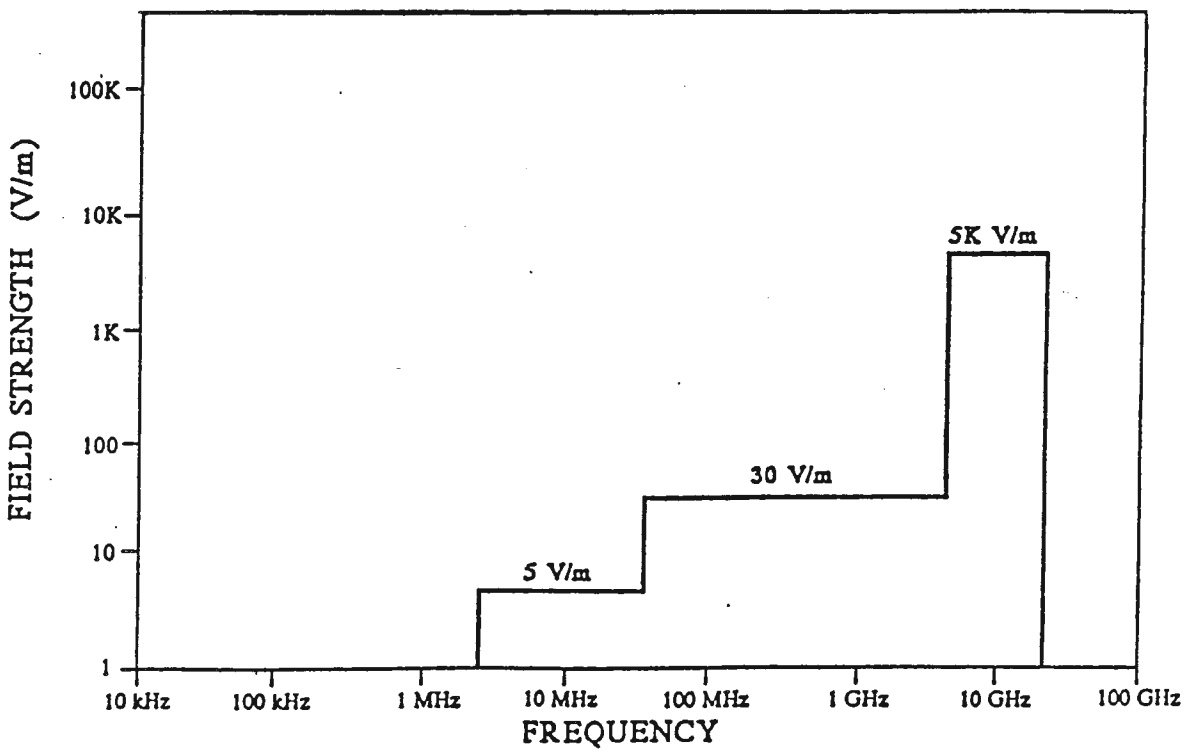



FIGURE 2.3-3. PREDICTED MAXIMUM PEAK FIELD STRENGTH LEVELS FROM CIVILIAN AIRPORT MOBILE EMITTERS

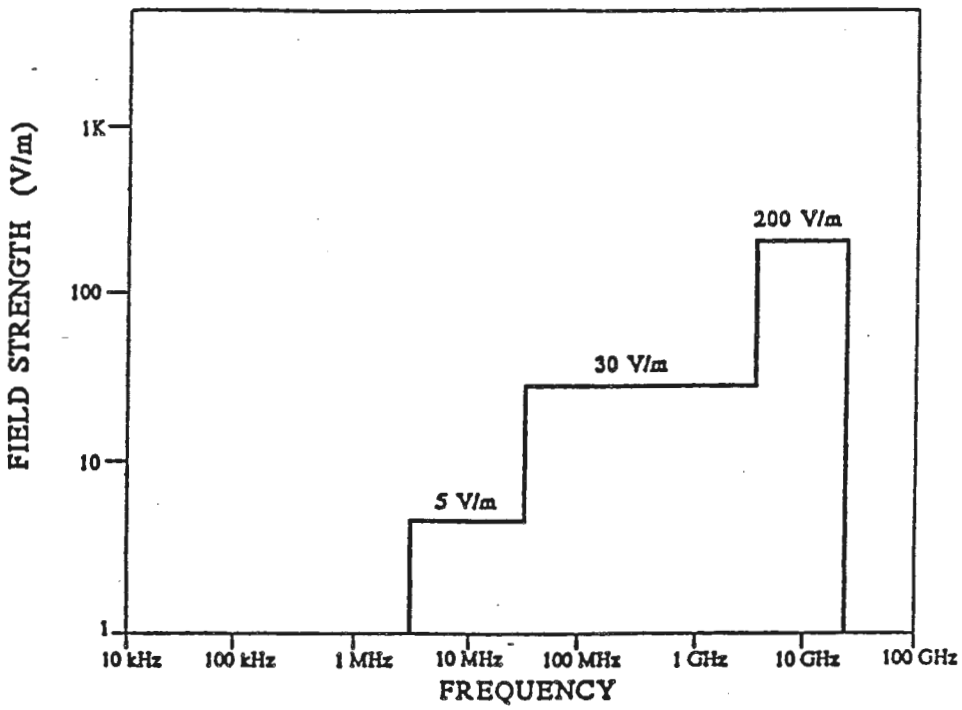



FIGURE 2.3-4. PREDICTED MAXIMUM AVERAGE FIELD STRENGTH LEVELS FROM CIVILIAN AIRPORT MOBILE EMITTERS

## 2.4. Air-to-Air Environment

In the air-to-air environment, normally, only military aircraft are considered. An interceptor aircraft may approach as close as 50 ft to an unidentified commercial aircraft while trying to establish its identity. Very high powered military Electronic Counter Measures (ECM) systems will be assumed as not operating at this close range. Non-interceptor aircraft, Airborne Warning and Control System (AWACS) or ECM aircraft would not intentionally approach closer than 500 ft but could be operating high-powered emitters. Predicted peak and average field strengths are calculated and plotted for the interceptor and noninterceptor aircraft in figures 2.4-1 through 2.4-4.

## 2.5. Ground Environment

The ground environment is the largest of the four environments since it includes all fixed and ground mobile emitters not located on an airport. These emitters are located at a distance greater than 5 miles from the center of the airport. The ICAO aircraft minimum obstruction clearance is 500 ft above ground level. The 500-ft minimum distance was used for field strength calculations; therefore, this places the aircraft within the near field of the emitting antenna. The ground equipment includes those types in table 2.5-1.

Predicted maximum peak and average field strengths are calculated and plotted for the ground environment equipment in figures 2.5-1 and 2.5-2.

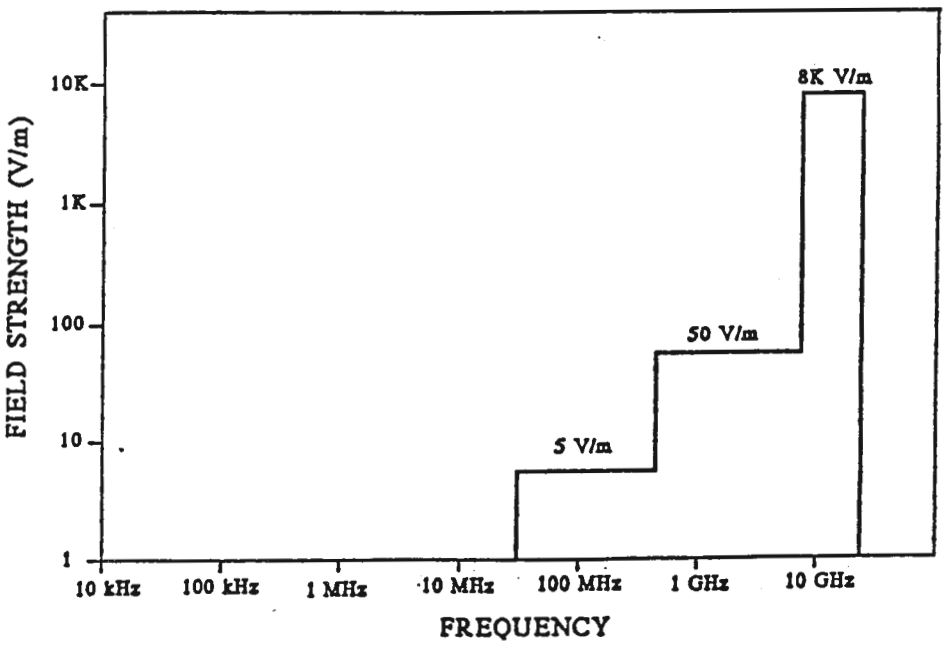



FIGURE 2.4-1. PREDICTED MAXIMUM PEAK FIELD STRENGTH LEVELS FROM INTERCEPTOR AIRCRAFT EMITTERS

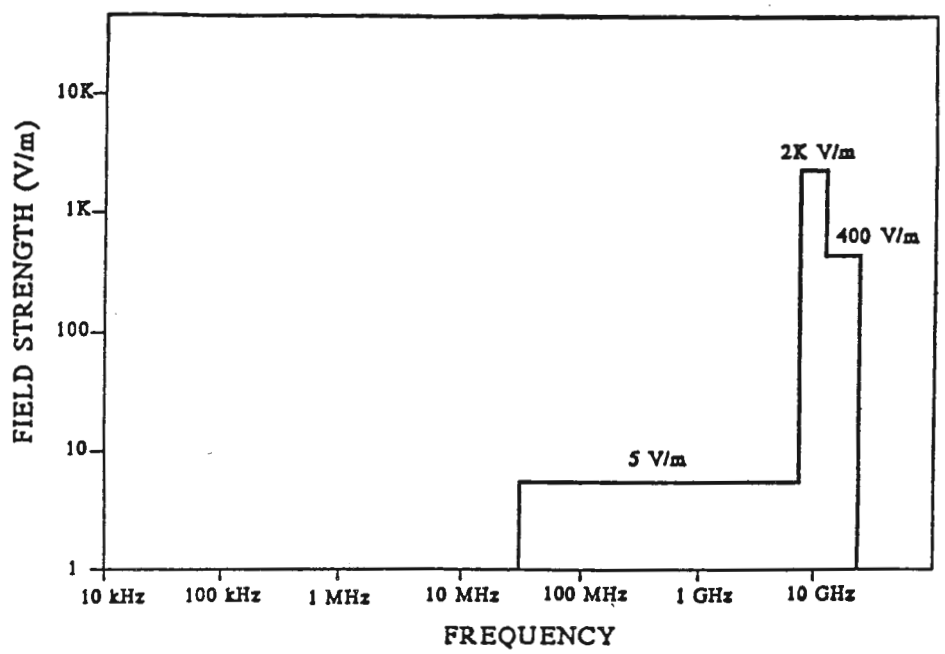



FIGURE 2.4-2. PREDICTED MAXIMUM AVERAGE FIELD STRENGTH LEVELS FROM INTERCEPTOR AIRCRAFT EMITTERS

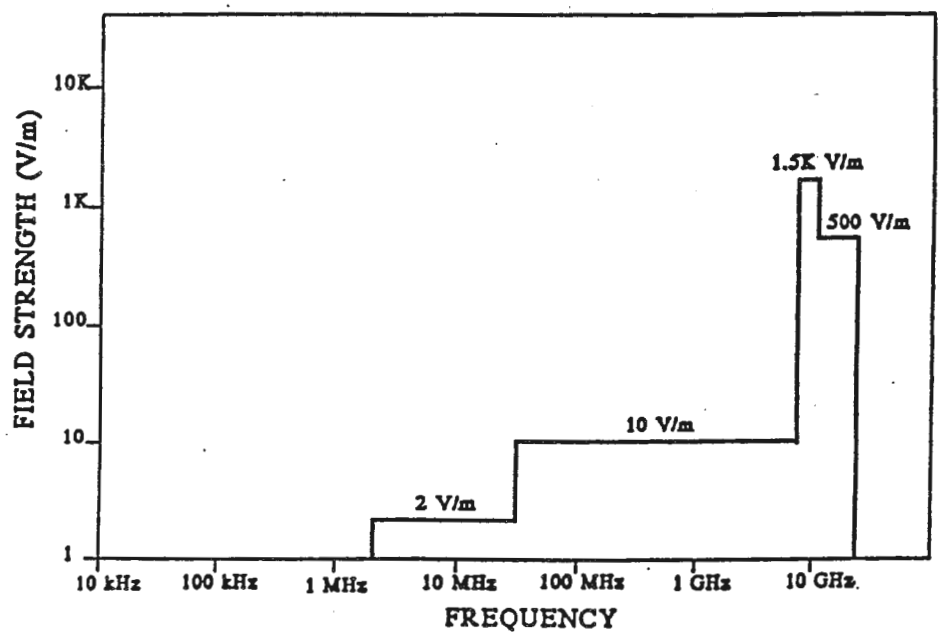



FIGURE 2.4-3. PREDICTED MAXIMUM PEAK FIELD STRENGTH LEVELS FROM MILITARY NON-INTERCEPTOR AIRCRAFT

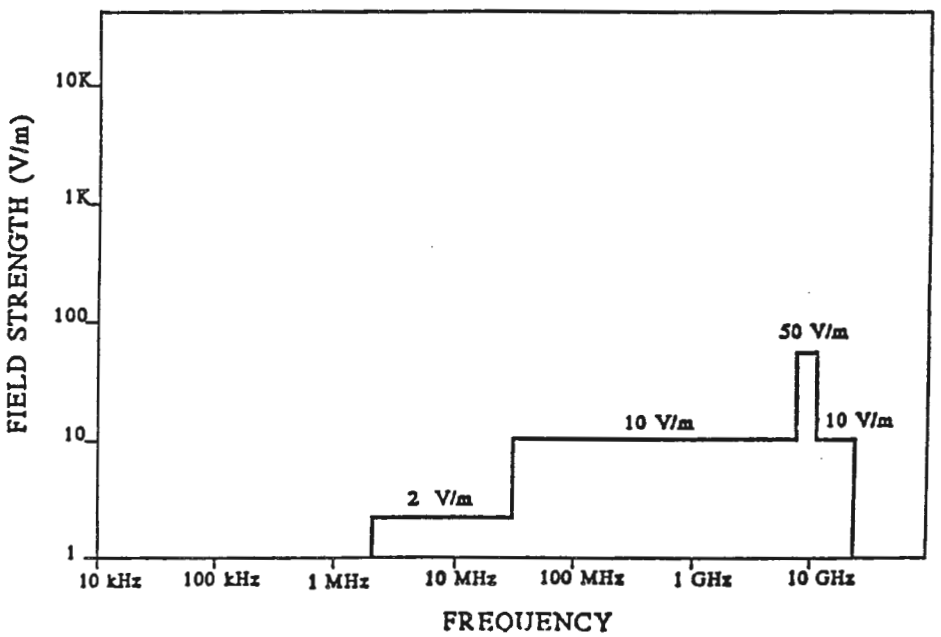



FIGURE 2.4-4. PREDICTED MAXIMUM AVERAGE FIELD STRENGTH LEVELS FROM MILITARY NON-INTERCEPTOR AIRCRAFT

TABLE 2.5-1. GROUND EQUIPMENT

| |
|---|
| Radars |
| Earth Terminals |
| Sounders |
| Troposcatter Communications |
| Commercial AM, FM, and TV Broadcast Transmitter |
| Test and Training Equipment |

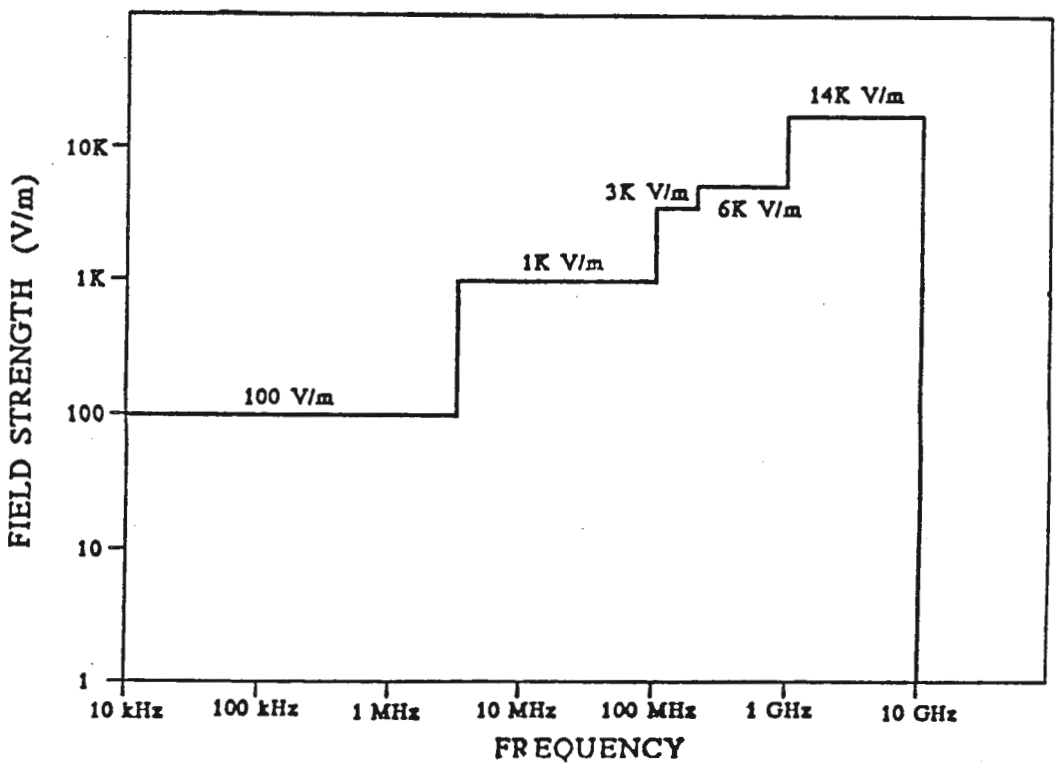



FIGURE 2.5-1. PREDICTED MAXIMUM PEAK FIELD STRENGTH LEVELS FROM GROUND EMITTERS

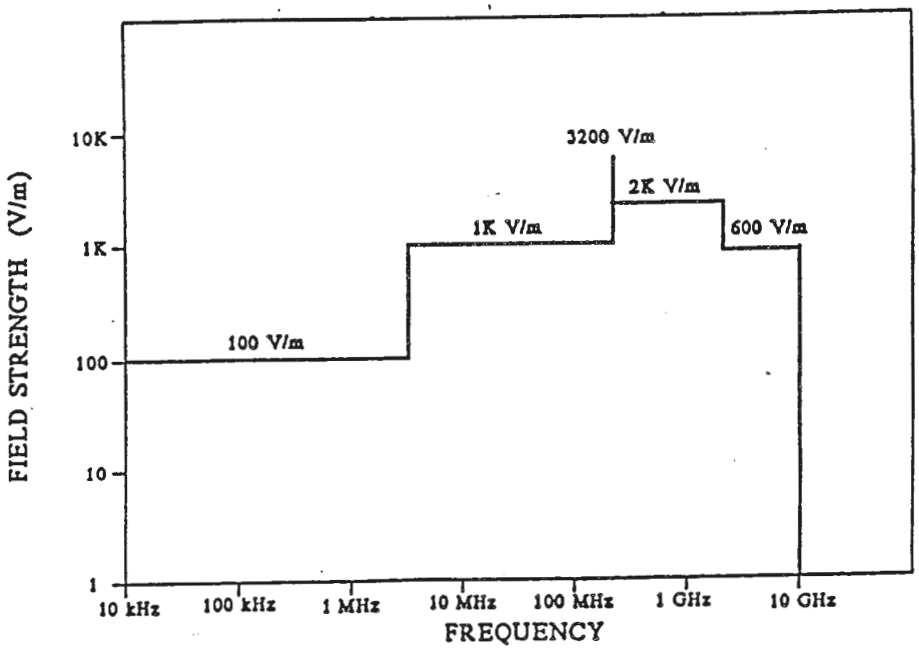



FIGURE 2.5-2. PREDICTED MAXIMUM AVERAGE FIELD STRENGTH LEVELS FROM GROUND EMITTERS

## 2.6. Shipboard Environment

A number of U.S. and international airports are located near harbors, shipping lanes, etc.; therefore, approach to landing and takeoff paths may cross over commercial and/or military ships, i.e., oil transports, cruise liners, aircraft carriers, and support vessels. Some of these ships, i.e., aircraft carriers, contain very high-powered emitters at high frequencies. A minimum distance of 1000 ft was considered for the field strength calculations. The shipboard equipment includes those types listed in table 2.6-1.

Predicted maximum peak and average field strengths are calculated and plotted for shipboard equipment in figures 2.6-1 and 2.6-2.

TABLE 2.6-1. SHIPBOARD EQUIPMENT

| |
|---|
| Search Radars |
| Track Radars |
| IFF/SIF |
| HF and UHF Communications |

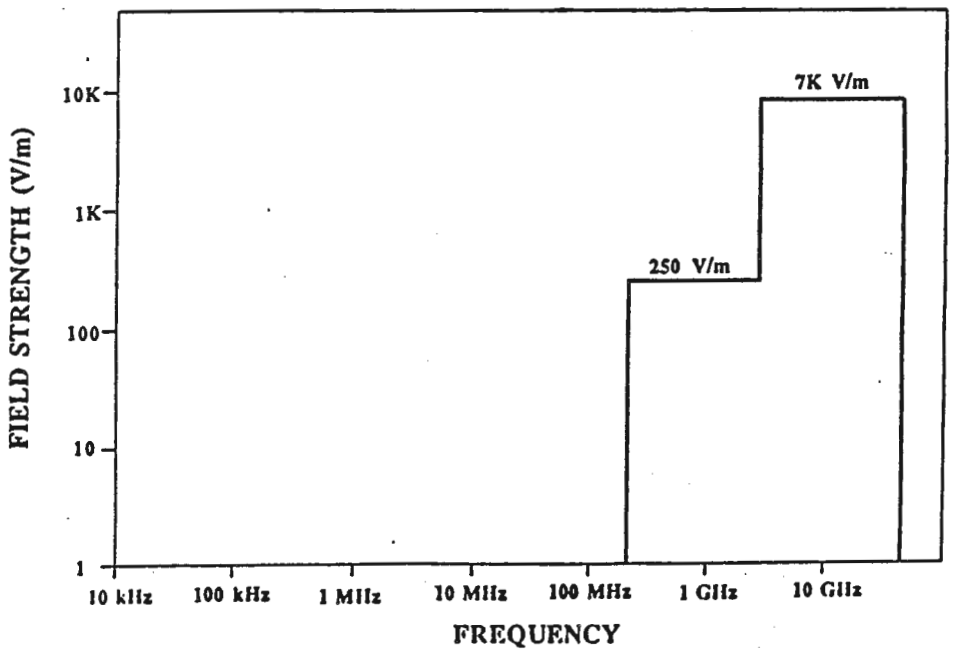



FIGURE 2.6-1. PREDICTED MAXIMUM PEAK FIELD STRENGTH LEVELS FROM SHIPBOARD EMITTERS

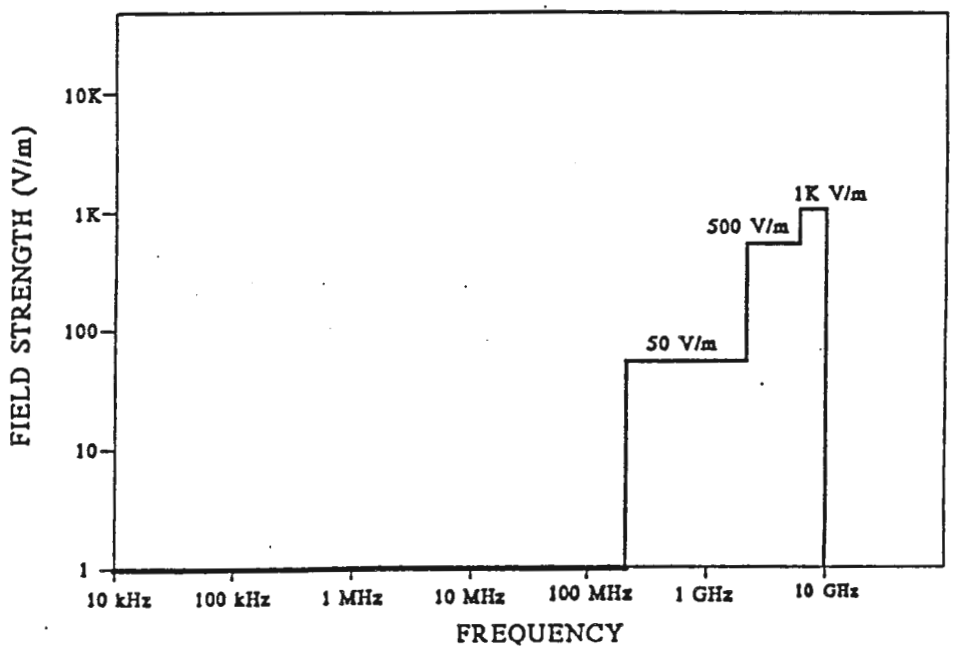



FIGURE 2.6-2. PREDICTED MAXIMUM AVERAGE FIELD STRENGTH LEVELS FROM SHIPBOARD EMITTERS

## 2.7. Composite of United States Emitters

The airport, air-to-air, ground, and shipboard peak and average field strengths were integrated into electric field strengths for the total U.S. environment. The U.S. environment includes both civil and military emitters. Predicted maximum peak and average field strengths are calculated and plotted for composite U.S. emitters in figures 2.7-1 and 2.7-2.

## 2.8. Composite of United States Compared to NATO STANAG 3614AE

Tracking of the total U.S. environment and the North Atlantic Treaty Organization (NATO) STANAG is remarkably close. The U.S. environment is not as granular as the NATO environments. In the early FAA and ECAC analysis, the charts were more granular, and the decision was made to present the data in a fashion as not to give enough information that would compromise the classified military emitters. Figures 2.7-1 and 2.7-2 are more granular than figure 2.8-1.

## 2.9. Takeoff and Landing Scenario

For minimum distance determination on and around airports, the worst-case condition is the engine out situation for a two-engine aircraft. The aircraft performance and FAA criteria dictates that the aircraft climb to and maintain a minimum altitude prior to initiating a go-around. Figure 2.9-1 shows the FAA criteria and the minimum altitudes that the aircraft must maintain.

The minimum distance between an aircraft and an emitter is dependent on the location of the emitter and the altitude of the aircraft. Where appropriate, the height of the emitting antenna above ground should be deducted from the aircraft minimum altitude to obtain the minimum distance from that specific emitter.

Based on the above criteria, minimum distances for field strength calculations can be estimated as follows:

- As expressed in the airport analysis above, a minimum distance of 250 ft is described near airport runways or taxiways.

- In an engine-out situation, the aircraft must be at an altitude of 35 ft at threshold (end of runway), climbing at a minimum gradient of 2.4 percent.

  This places the aircraft at an altitude of 300 ft at a distance of 1.84 nmi from the end of the runway.

- The aircraft will be at an altitude of 300 ft to a distance of 4.5 nmi from the end of the runway.

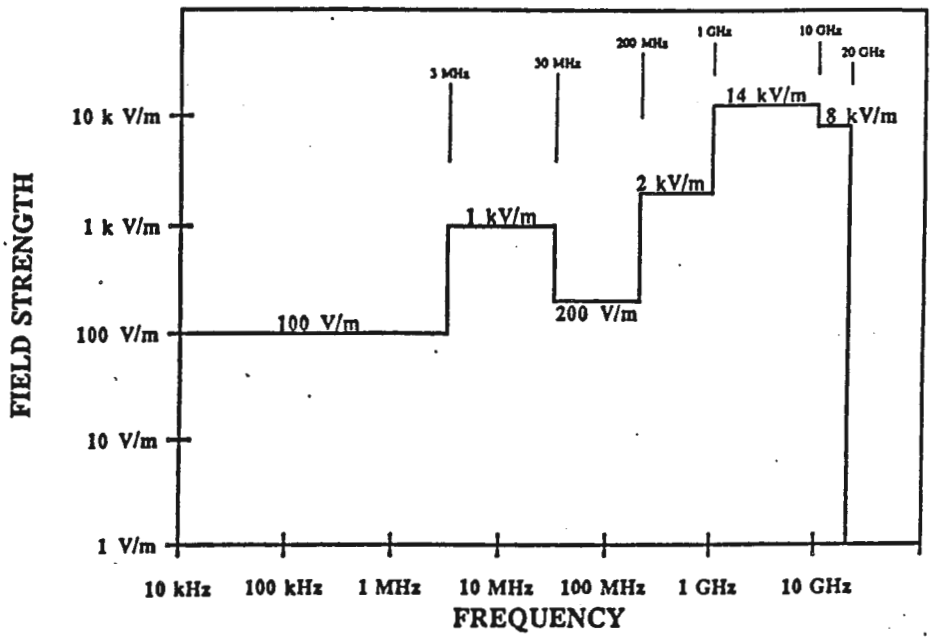



FIGURE 2.7-1. PREDICTED MAXIMUM PEAK FIELD STRENGTH LEVELS COMPOSITE OF UNITED STATES EMITTERS

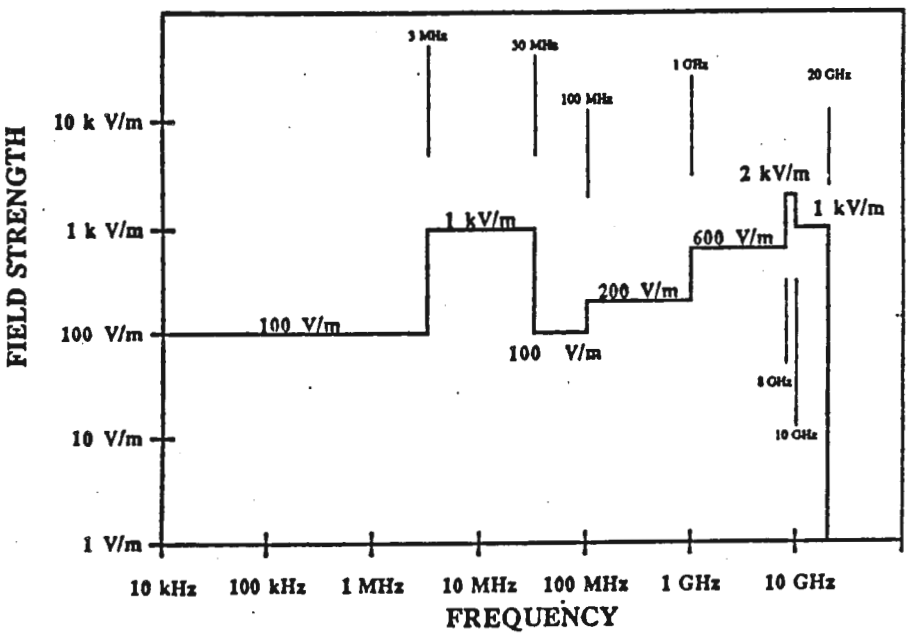



FIGURE 2.7-2. PREDICTED MAXIMUM AVERAGE FIELD STRENGTH LEVELS COMPOSITE OF UNITED STATES EMITTERS

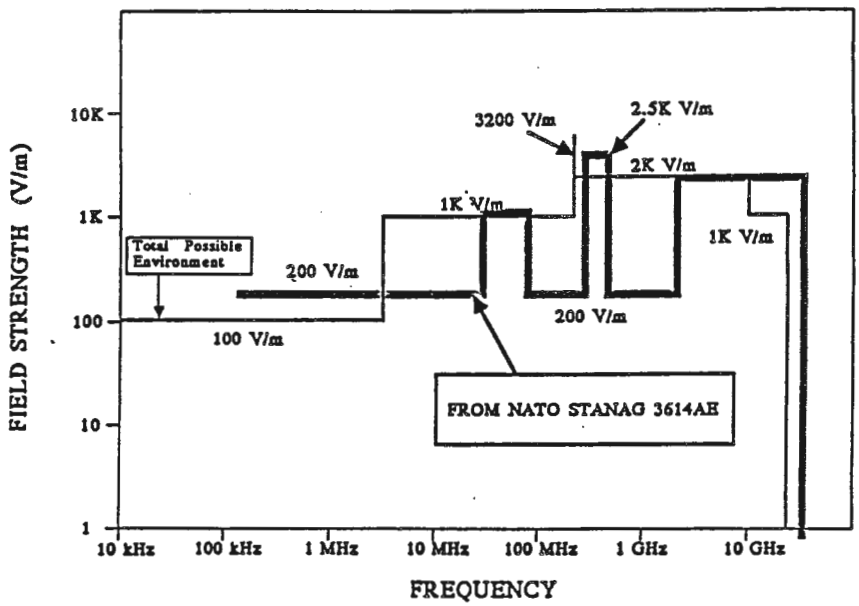



FIGURE 2.8-1. PREDICTED MAXIMUM AVERAGE FIELD STRENGTH LEVELS UNITED STATES COMPOSITE VS NATO STANAG 3614AE

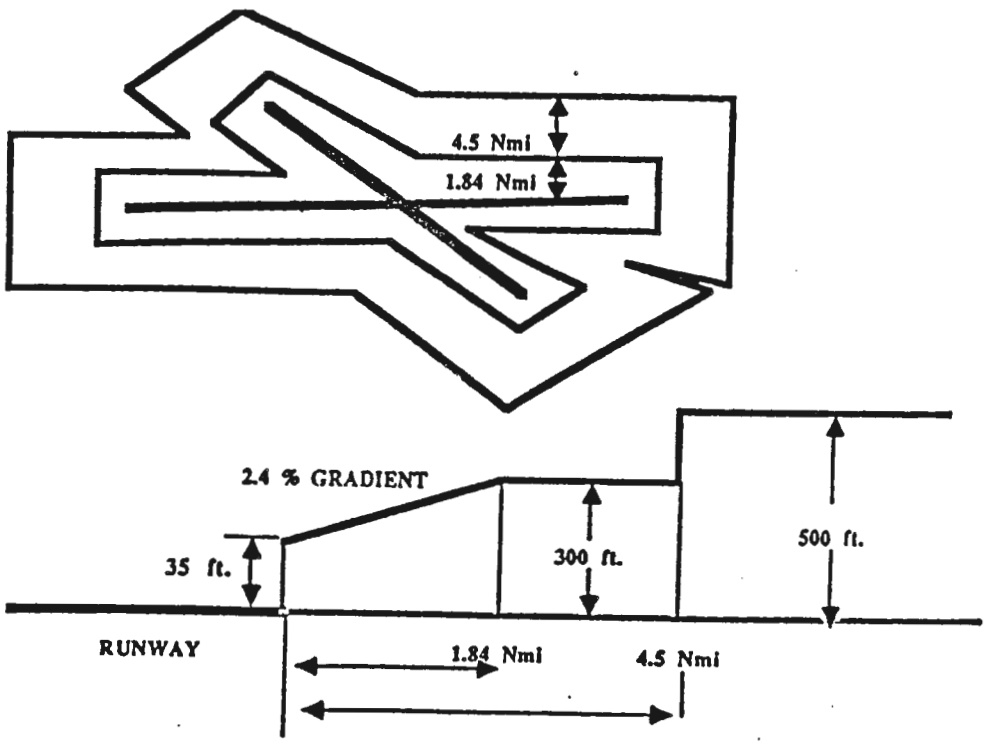



FIGURE 2.9-1. CRITERIA FOR SUSCEPTIBILITY THRESHOLD

- At 4.5 nmi, the minimum altitude is specified as 500 ft.

Note: In the shipboard environment, the minimum distance was considered to be 1000 ft, but the airport takeoff and landing scenario should be considered for those airports near the shipping lanes, which would reduce the minimum distance from 1000 ft to 300 ft. Recalculation of the peak and average field strengths in the shipboard environment is in progress.

The six airports that were examined in the United States are Baltimore, Honolulu, New York (Kennedy), Los Angeles, Seattle, and Denver (Stapleton). They were selected as being representative and typical of U.S. airports with certain unique local environments, industry, seaports, etc.

It is possible that the field strength levels may be higher at other airports than for the six airports selected, but there is a confidence level for those data being representative.

Peak and average field strengths were calculated for each airport, resulting in 12 graphs. Two composite graphs were developed from the 12 individual graphs. They are compared and overlaid in figures 2.9-2 and 2.9-3.

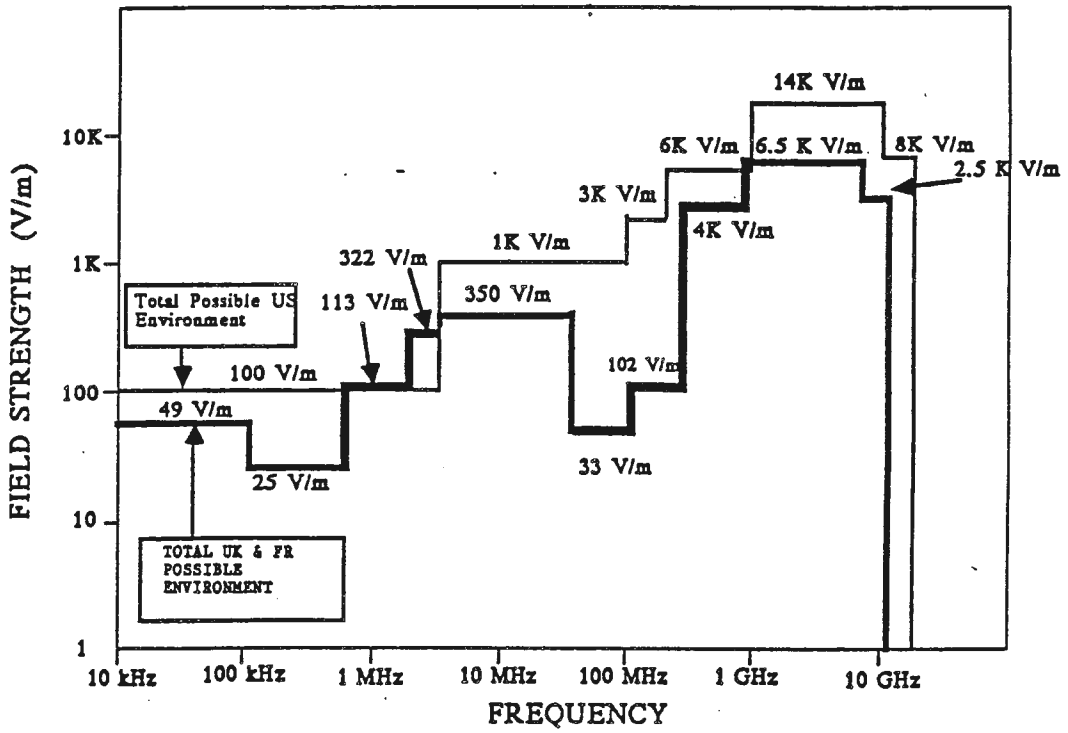



FIGURE 2.9-2. PREDICTED MAXIMUM PEAK FIELD STRENGTH LEVELS COMPARISON UNITED STATES COMPOSITE WITH SIX AIRPORT COMPOSITE

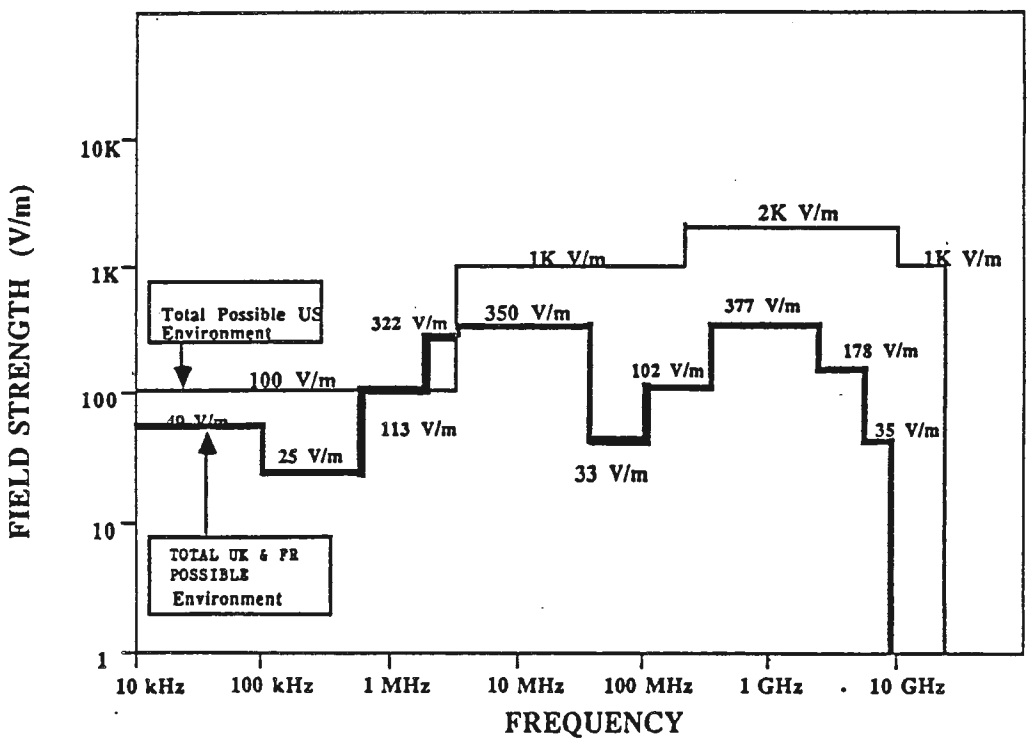



FIGURE 2.9-3. PREDICTED MAXIMUM AVERAGE FIELD STRENGTH LEVELS COMPARISON UNITED STATES COMPOSITE WITH SIX AIRPORT COMPOSITE

# 3. EME SURVEY AND ANALYSIS - WORLDWIDE

## 3.1. Guidelines

The CAA, U.K., initiated requests for other European countries to participate in this HERF survey and to study each country's own emitter environments. Initially, only the United Kingdom and France responded and aggressively supported this total effort. Their results are presented herein. In September 1987, 27 countries were invited to support and participate in an International Conference in Washington, D.C. Ten countries attended including the United States, United Kingdom, and France. Canada and Sweden were the only other countries in which preliminary surveys were conducted. Sweden's early results were presented at this conference by the Swedish representative.

In April of 1988, a similar international conference was conducted in Brighton, England. The same 27 countries were invited to participate and present their survey data. Australia and West Germany provided an overview of their preliminary data.

British, French, and Swedish technical teams used the FAA and ECAC format as guidelines for their survey and analysis. France, associated with its survey and analysis, conducted an extensive airborne (rotorcraft) measurement program for selected emitters. The U.S. measurement and validation effort was conducted by the FAA, ECAC, and Ohio University. The summary of these efforts will be described later in the validation study currently in progress.

## 3.2. Results

Results are presented in a comparison of the American worst-case total environment to the composite of the British and French survey data. Figures 3.2-1 and 3.2-2 show the predicted maximum peak and average field strength levels for this comparison.

Figures 3.2-3 and 3.2-4 show composite worst-case environment for all three countries.

Figures 3.2-5 and 3.2-6 show U.S. airports (Baltimore, Honolulu, Kennedy, Los Angeles, Seattle, Denver (Stapleton)) integrated with the British airports (Heathrow, Manchester, Glasgow) and the French airports (Orly, Toulouse, Lyon, Nice).

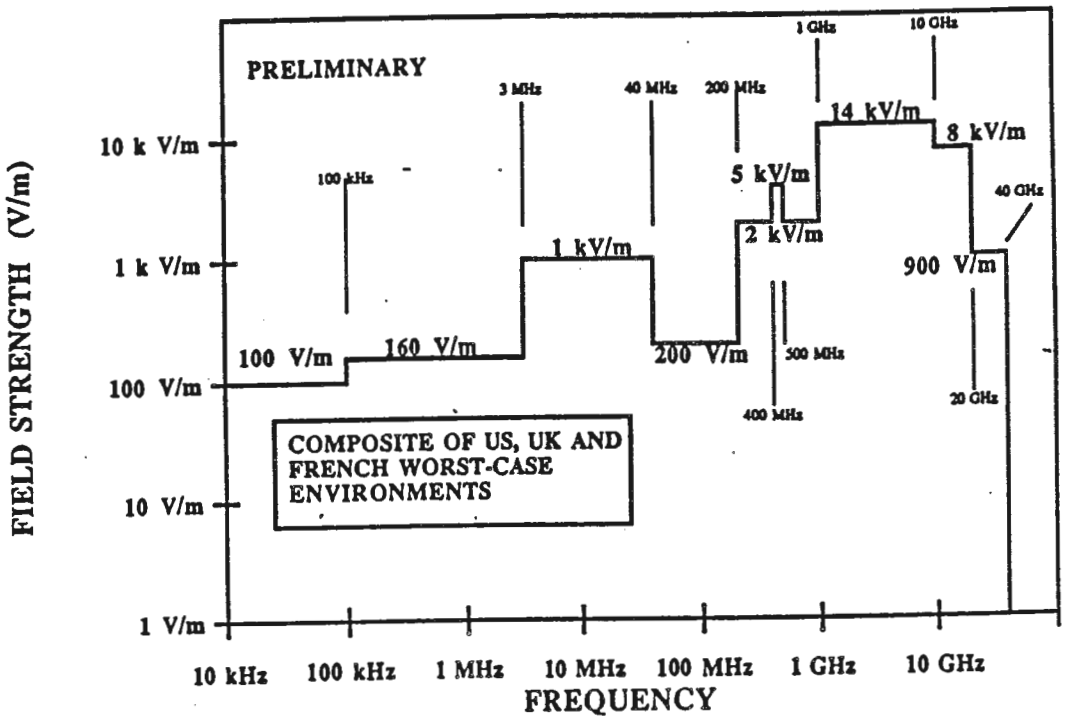



FIGURE 3.2-1. PREDICTED MAXIMUM PEAK FIELD STRENGTH LEVELS COMPARISON OF UNITED STATES COMPOSITE WITH UNITED KINGDOM AND FRANCE COMPOSITE

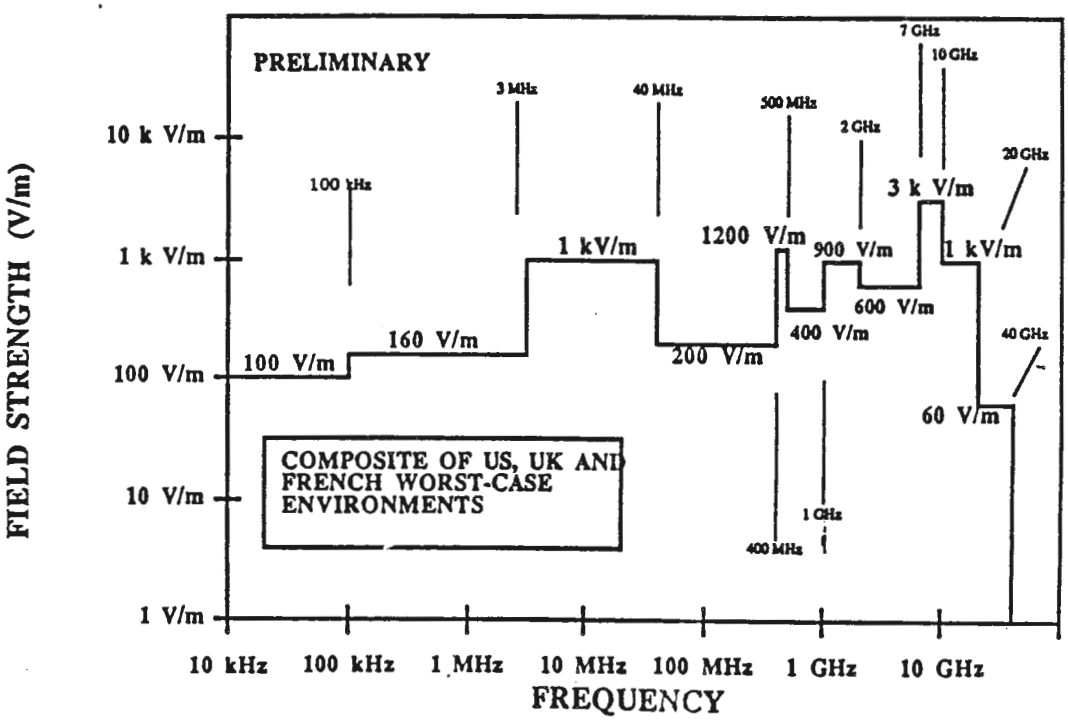



FIGURE 3.2-2. PREDICTED MAXIMUM AVERAGE FIELD STRENGTH LEVELS COMPARISON OF UNITED STATES COMPOSITE WITH UNITED KINGDOM AND FRANCE COMPOSITE

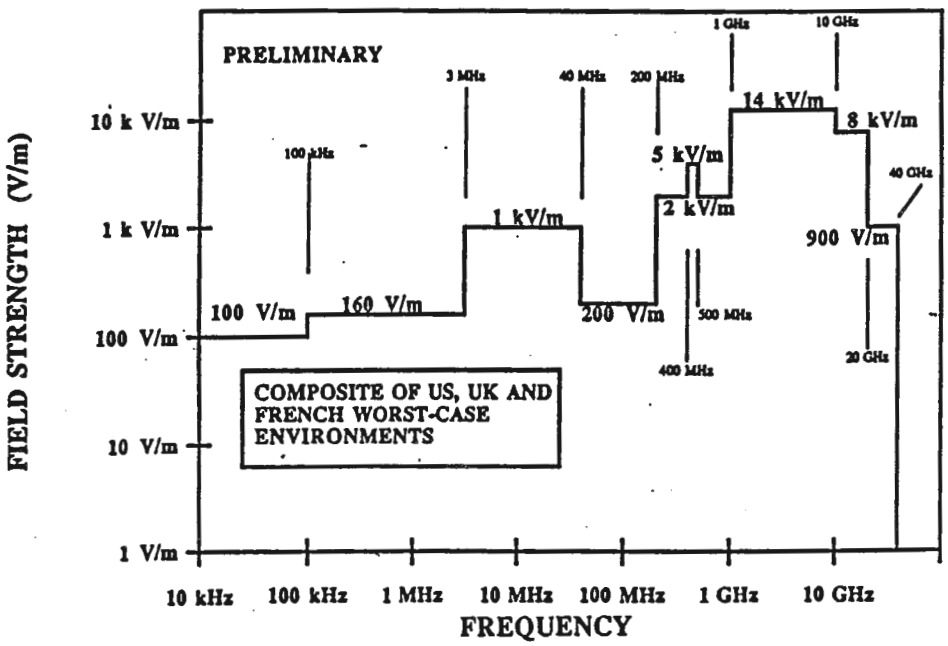



FIGURE 3.2-3. PREDICTED MAXIMUM PEAK FIELD STRENGTH LEVELS WORLDWIDE WORST-CASE ENVIRONMENT

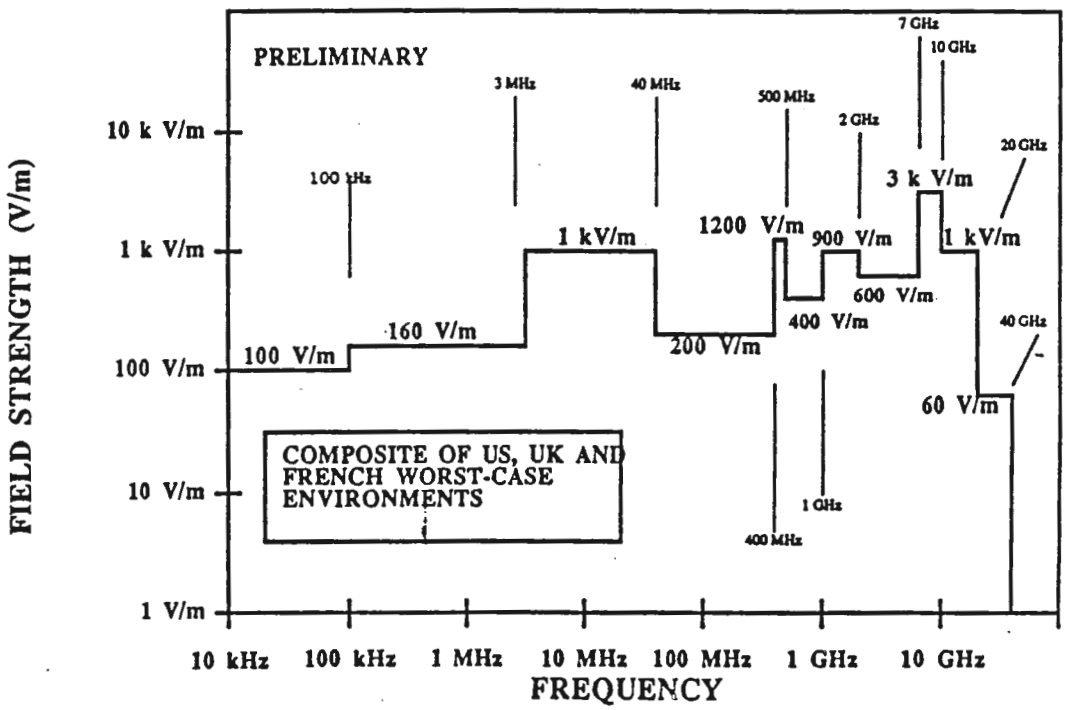



FIGURE 3.2-4. PREDICTED MAXIMUM AVERAGE FIELD STRENGTH LEVELS WORLDWIDE WORST-CASE ENVIRONMENT

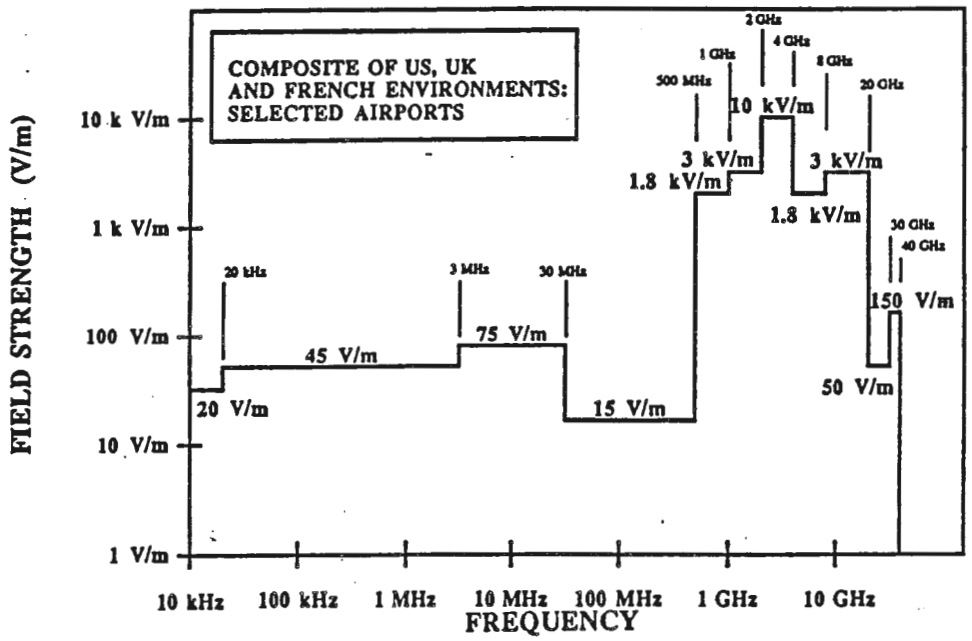



FIGURE 3.2-5. PREDICTED MAXIMUM PEAK FIELD STRENGTH LEVELS COMPOSITE OF INTERNATIONAL ENVIRONMENTS

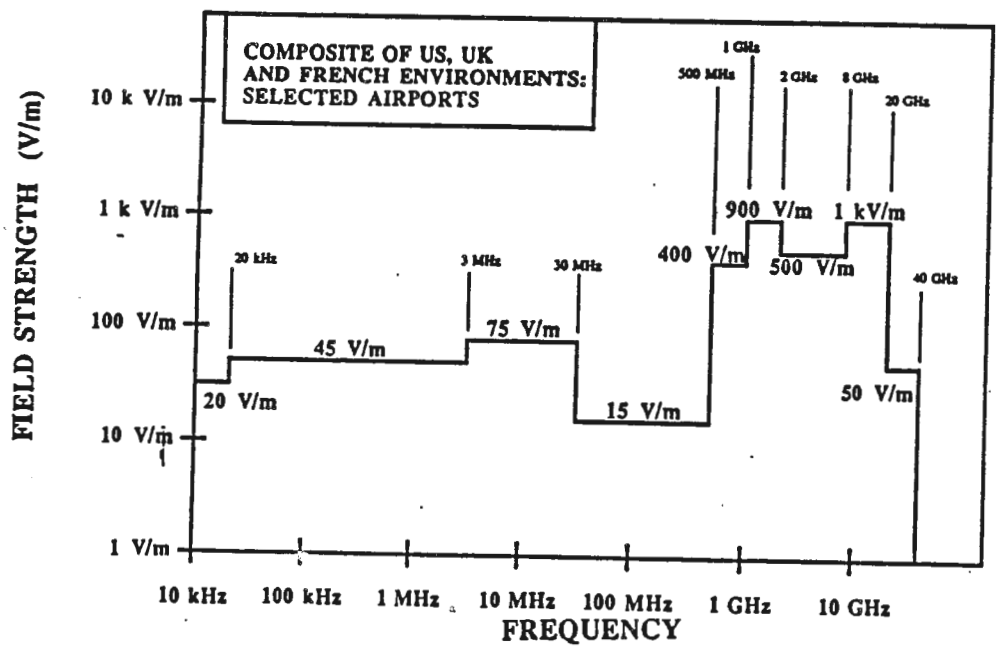



FIGURE 3.2-6. PREDICTED MAXIMUM AVERAGE FIELD STRENGTH LEVELS COMPOSITE OF INTERNATIONAL ENVIRONMENTS

## 4. COMMENTS

Based on the initial survey and analysis of our current environment, HERF fields are indeed a threat to state-of-the-art civil and military aircraft being placed into service throughout the world. Modern aircraft with composite construction, VLSI computers, and electronic devices for flight-critical functions have greatly magnified the threat of EMEs to safe flight operations.

The FAA, in concert with the DOD, U.K. CAA, French DGAC, and their European counterparts, are aggressively pursuing the research to meet this challenge. Following the survey and analysis, a validation phase of selected emitters will be conducted by Ohio University and ECAC. The guidelines established will promote uniformity of results from other members of the international aviation community so that all results can be combined and used to develop an international susceptibility standard.

RTCA was solicited by the FAA to establish a HERF subcommittee to update RTCA DO-160B, Section 20 to address the FAA EME as developed. A HERF Working Group was established under the RTCA Special Committee (SC-135) to review the equipment radiation susceptibility levels currently in DO-160B (which are inadequate for the FAA HERF environment) and then establish an industry-wide aircraft requirement for conducted and radiated susceptibility.

The FAA solicited the SAE-AE4 Committee for the development of procedures and guidance material, relative to HERF fields, which could be used during the aircraft certification process. Based on this request, SAE-AE4R, Aircraft Radiated Environment Subcommittee, was formed. The goal of the subcommittee was to draft an AC and attendant user's manual for guidance and information on HERF use in commercial aircraft.

SAE-AE4R subcommittee established three subcommittees: a Data Accuracy Subcommittee that will look at the exterior electromagnetic RF environment, a Design Approach Subcommittee that will develop design guidance for airframe manufacturers for the defined EME, and a Test and Analysis Methods subcommittee that will focus on validation techniques.

APPENDIX A

# GUIDELINES FOR DEVELOPING MAXIMUM PEAK AND AVERAGE FIELD STRENGTH ENVELOPE GRAPHS FOR AIRCRAFT

BY ALEXANDER GROSS
OF THE
DOD ELECTROMAGNETIC COMPATIBILITY ANALYSIS CENTER
ON BEHALF OF
THE DEPARTMENT OF TRANSPORTATION
FEDERAL AVIATION ADMINISTRATION

SEPTEMBER 1987

## APPENDIX A - GUIDELINES FOR DEVELOPING MAXIMUM PEAK AND AVERAGE FIELD STRENGTH ENVELOPE GRAPHS FOR AIRCRAFT

### BACKGROUND

The U.S. Federal Aviation Administration (FAA) is concerned with the safety and integrity of new technology transport-category airplanes with fully flight-critical and flight-essential systems. One of their concerns is the protection of these aircraft from electromagnetic radiation (EMR) produced by external high-energy ground and airborne radio-frequency (RF) emitters. The FAA is in the process of developing proposed requirements and standards to address this concern.

As part of this effort, the Electromagnetic Compatibility Analysis Center (ECAC) was requested to predict the maximum EMR levels at aircraft in the United States for two specified scenarios: a worst-case total U.S. scenario and, for comparison, a take-off and landing scenario at specific airports. The worst-case scenario considered the maximum EMR levels at an aircraft exposed to any emitter in the U.S. The second scenario considered the EMR that would be experienced by an aircraft on take-off and landing at six major U.S. airports.

Those results of this study will be used by the FAA in the development of U.S. requirements and standards relating to the effects of transmitted RF energy on flight-critical control and avionics systems aboard modern aircraft. Since aircraft are required to fly throughout the world, an international RF susceptibility standard is also desired. It is suggested that maximum EMR level predictions should be made by other members of the international aviation community to support an international standard. In an effort to standardize the prediction techniques, ECAC has been requested by the FAA to document the techniques that were used by ECAC in the form of guidelines for predicting maximum EMR levels at aircraft exposed to high-powered emitters.

### OBJECTIVE

The objective of this paper is to present those guidelines that were used by ECAC in the calculation of maximum peak and average electric field strengths at transport-category aircraft with fully electronic flight-critical and flight-essential systems as a result of ground and airborne RF emitters. These guidelines are presented to promote uniformity of results from other countries so that all inputs can be combined.

### APPROACH

The guidelines for calculating peak and average field strengths are presented as step-by-step instructions starting with the data on the emitters in the electromagnetic environment (EME) and ending with an envelope graph representing maximum field strengths. The methods for calculating field strengths in the worst-case in the worst-case scenario and in the take-off and landing scenario are defined and discussed. Equations used to estimate missing technical parameters, to calculate the near field of an antenna, and to calculate the field strengths are provided. Simplifying assumptions are made and, where

appropriate, justifications are shown. The minimum distances used in the U.S. environmental analysis are presented. The results obtained by following these guidelines will be peak and average field strengths, at worst-case distances between emitters in the EME and aircraft.

## PROCEDURES

### GENERAL ASSUMPTIONS

The calculation of maximum EMR is based on several assumptions. These are:

1. The minimum allowable separation distance between civil aircraft and an RF emitter is used to calculate the maximum electric field strength at the aircraft. The minimum distance is a worst-case distance that is determined by the scenario being considered and by the environment within that scenario.

2. The civil aircraft is being continuously illuminated by the mainbeam of the emitting antenna.

3. All calculations consider only free-space loss between the emitter and the aircraft. Above 30 MHz, no consideration is given to reflecting surfaces or obstacles that may exist in the vicinity of either the emitter or the aircraft. Below 30 MHz, only the ground reflected wave is considered and assumed to combine constructively with the direct wave, effectively doubling the field strength.

4. Field strength calculations for aircraft locations within the near field of the emitting antenna are made using the near-field correction factors discussed later in this paper.

5. The resulting field strength is independent of modulation or any other equipment-related factor.

6. The cumulative effect of two or more emitters illuminating the aircraft is not considered.

### DETERMINATION OF THE MINIMUM DISTANCES FOR THE WORST-CASE TOTAL ENVIRONMENT SCENARIO

The total environment can be separated into four parts: the airport environment, the aircraft-to-aircraft environment, the ground environment, and the shipboard environment. While on the ground, the aircraft will be exposed to emitters in the airport environment. While in flight, the aircraft will encounter emitters in the aircraft-to-aircraft, ground, and shipboard environments. The minimum distance for an aircraft in one of these environments is defined as the closest the aircraft can approach an emitter in that environment. Each environment and its minimum separation distance will be discussed in turn. The minimum distances are summarized in table A-1.

TABLE A-1. SUMMARY OF MINIMUM DISTANCES FOR THE WORST-CASE SCENARIO

| ENVIRONMENT | MINIMUM DISTANCE |
|---|---|
| FIXED EMITTERS ON AIRPORT | 250 FT |
| MOBILE EMITTERS ON AIRPORT | 50 FT |
| INTERCEPTOR AIRCRAFT EMITTERS | 50 FT |
| NON-INTERCEPTOR AIRCRAFT EMITTERS | 500 FT |
| GENERAL GROUND EMITTERS | 500 FT |
| SHIPBOARD EMITTERS | 300 FT |

The Airport Environment

The airport environment consists of fixed and mobile emitters located on an airport. For this discussion, the airport includes an area defined by a five-nautical-mile radius from the geographic center of the airport. The minimum distance at which field strengths are to be calculated differs for fixed and mobile emitters.

Fixed Emitters. The fixed airport environment consists of all fixed emitters located on an airport. In general, the minimum distance between a fixed emitter and the taxiway or runway is defined by regulations. This minimum distance is used as the minimum separation distance between the aircraft and the fixed airport environment.

Table A-2 lists the typical fixed emitters in the U.S. airport environment. Not all of these emitters are located on every airport, but they are located on some airport in the United States. Based on FAA regulations, fixed emitters are located no closer to an airport runway or taxiway than 250 ft. A minimum distance of 250 ft, therefore, was used in the calculation of the field strengths from the fixed emitters on a U.S. airport.

Mobile Emitters. The mobile airport environment consists of all emitters that are not in a fixed location. Emitters, such as commercial VHF radios on ground support vehicles, are part of the mobile environment. Emitters, such as HF and UHF communications, TACAN, Doppler Navigation Radars, radio altimeters, aircraft weather radars, and ATCRBS beacons, aboard other aircraft are also included in the mobile airport environment. These aircraft emitters may be operating while the aircraft is on the ground and waiting to take off.

A minimum distance of 50 ft was used in calculating field strengths from mobile emitters on the airports in the U.S. analysis.

TABLE A-2. TYPICAL FIXED AIRPORT ENVIRONMENTAL EMITTERS

| |
|---|
| MARKER BEACON |
| LOCALIZER |
| VOR |
| GLIDE SLOPE |
| GROUND CONTROLLED APPROACH RADAR |
| DISTANCE MEASURING EQUIPMENT |
| TACAN |
| MICROWAVE LANDING SYSTEM |
| AIRPORT SURVEILLANCE RADAR |
| AIR ROUTE SURVEILLANCE RADAR |
| WEATHER RADAR |
| ATCRBS INTERROGATOR |
| VHF AND UHF COMMUNICATION AND TELEMETRY |

Aircraft-to-Aircraft Environment

This environment consists of emitters onboard other aircraft at distances much closer than normal in-flight separation. Note that the emitters on other civil aircraft were included in the mobile airport environment. Therefore, only military aircraft need to be considered in this environment.

The military aircraft fall into two categories: interceptor and noninterceptor aircraft. An interceptor aircraft may approach as close as 50 ft to an unidentified commercial aircraft while attempting to establish the identity of, and provide instructions to, the aircraft. It can be assumed that very high powered military emitters such as electronic countermeasure (ECM) systems will not be operating at this close range. Noninterceptor aircraft would not intentionally approach closer than 500 ft from another aircraft but may be operating high powered emitters such as early warning radars and ECM systems. In the U.S. study, ECAC used these distances when calculating electric field strengths.

Ground Environment

Of the four environments, the ground environment is the largest since it includes all fixed and ground mobile emitters that are not located on airports. Emitters such as sounders, troposcatter communications, commercial broadcast, radar astronomy, land mobile, test and training equipment, radars and transmitting earth terminals are part of the ground environment.

The International Civil Aviation Organization (ICAO) aircraft minimum obstruction clearance is 500 ft above ground level. Therefore, the minimum distance

at which field strengths are calculated is 500 ft. Note that for most high power emitters, this places the aircraft within the near field of the emitting antenna.

Shipboard Environment

This environment includes emitters on commercial and military ships that may be located in harbors near airports. Some of the high-powered emitters that are included are: search and track radars, fire control radars, satellite communications, HF communications, and electronic countermeasure systems. A minimum distance of 1000 ft was used in the U.S. analysis based on the locations of U.S. harbors.

DETERMINATION OF THE MINIMUM DISTANCES FOR THE TAKEOFF AND LANDING SCENARIO

The takeoff and landing scenario is concerned with the environment on and around airports, where the aircraft is expected to be near the ground. The worst-case condition is the one-engine out situation for a two engine aircraft. In this situation, the aircraft must climb to a minimum altitude and must maintain that altitude. Figure A-1 shows the FAA criteria and the minimum altitudes that the aircraft must maintain. Based on this criteria, the minimum distances for field-strength calculations are as follows:

1. While on the airport runways or taxiways, the minimum distance is 250 ft, as discussed under the airport environment section, above.

2. The aircraft must be at an altitude of 35 ft at threshold (end of runway), climbing at a minimum gradient of 2.4%. This places the aircraft at an altitude of 300 ft at a distance of 1.84 nautical miles from the end of the runway.

3. The aircraft will be at an altitude of 300 ft to a distance of 4.5 nmi from the end of the runway.

4. At 4.5 nmi, the minimum altitude is specified as 500 ft.

5. The minimum distance between an aircraft and an emitter is dependent on the location of the emitter and the altitude at that location as given in 1 through 4 above. Where appropriate, the height of the emitting antenna above ground should be deducted from the minimum aircraft altitude to obtain the minimum distance from that specific emitter.

CALCULATION OF FIELD STRENGTHS

The conventional measure of the intensity of a radiated electromagnetic wave is a measure of the intensity of the electric field in volts per meter. (Note that the distances in the above discussion were in English units since this is common for the aviation industry. The discussion of electric field strengths will be in metric units with volts per meter being the unit used in international regulations.) The field intensity of a radiated wave in free space falls off in direct proportion to the distance from the transmitting antenna. For practical purposes free space propagation is realized if the following conditions are

fulfilled: No large obstacles intervene between the antenna and the aircraft along an optical line of sight; no alternate transmission path can be followed by a substantial fraction of the radiated energy (this is the same as Assumption 3 in the Introduction section); the intervening atmosphere has a constant index of refraction, so that no bending of the wave occurs; and the intervening atmosphere does not absorb energy from the wave at the frequency used. For this analysis, it is assumed that these conditions are fulfilled between an emitter in the environment and an aircraft.

The emitter antenna will be either an aperture, phased array or linear antenna. Phased array antennas are assumed to have similar radiating properties to an aperture antenna of the same physical dimensions. Elliptical antennas are approximated by rectangular antennas for these calculations.

Calculation of Field Strengths in the Far Field

Aperture Antennas. The power density obeys the inverse square law starting at a certain distance from the antenna where the antenna appears to be a point source. This is where the "Fraunhofer Region," or far field, begins. The distance to the far-field boundary for aperture antennas can be designated as:

$$D_f = 2\ L^2/\lambda \qquad (1)$$

where:

$D_f$ - distance to the far-field boundary, meters

$L$ - Maximum dimension of antenna, meters

$\lambda$ - wavelength, meters

In the Fraunhofer Region, the radiation is in a diverging beam shape, where the intensity is maximum at the beam center, and decreases away from the beam center as the angle of divergence increases. If the antenna is made larger or the antenna illumination is made more uniform, the radiated beam is made narrower and the beam center power density is made higher. The power density at the beam center in the far field is given by the equation:

$$P_D = \frac{PG}{4\pi d^2} \qquad (2)$$

where:

$P_D$ - power density, watts/meter$^2$

$P$ - transmitter power, watts

$G$ - transmitter antenna gain, unitless

$d$ - distance, meters

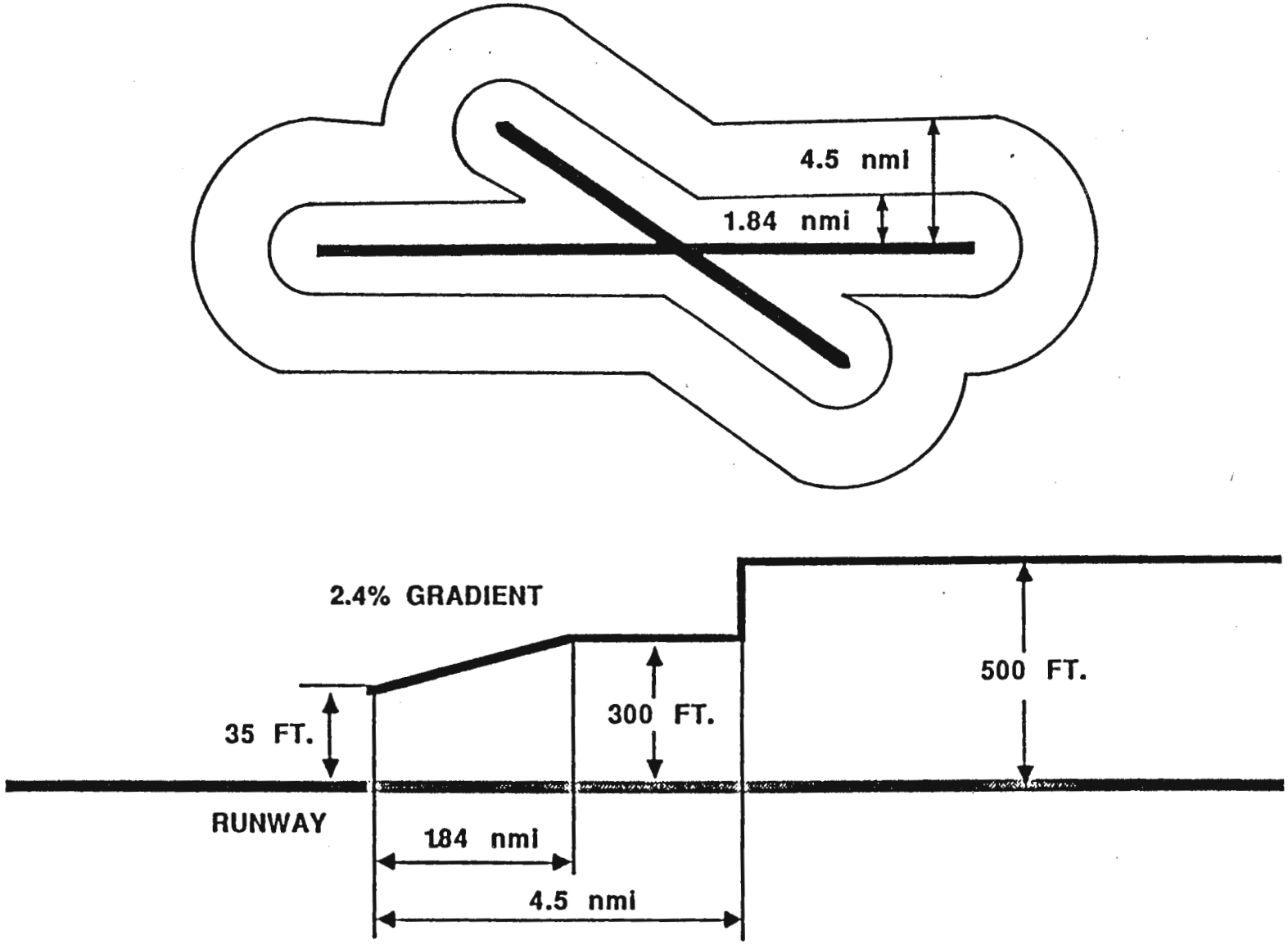



FIGURE A-1. FAA CRITERIA FOR A ONE-ENGINE OUT SITUATION FOR A TWO-ENGINE AIRPLANE

Power density is converted to field strength (FS) using the impedance of free space in Equation 3:

$$FS = \sqrt{P_D(377)} \tag{3}$$

Linear Antennas. Equations 2 and 3 can be used to calculate the electric field strength at given distances from a linear emitting antenna. The far-field boundary is designated as $4\lambda$.

Calculation of Field Strengths in the Near Field

Power densities in the near field are calculated using the far-field equation (Equation 2) and a near-field gain reduction factor ($\chi$), or:

$$P_D = \frac{PG\chi}{4\pi d^2} \tag{4}$$

where all values are previously defined. Power density is converted to field strength using Equation 3.

The near field is the region within the far-field boundary, as defined by Equation 1 for aperture antennas and by $4\lambda$ for linear antennas. It should be noted that for the aircraft environment scenarios described above the aircraft is within the near field of most emitters using aperture antennas and in the far field for all of the emitters using linear antennas.

Near field gain reduction factors were calculated for the ECAC study using US Department of Defense approved techniques. A summary of these techniques for aperture antennas is presented below. No near field gain reduction techniques are required for linear antennas.

Near-field Gain Reduction for Rectangular Aperture Antennas. For rectangular aperture antennas the vertical and horizontal axis may not have the same illumination taper. Therefore, the gain reduction for each axis must be determined, then summed (in dB) to determine the total gain reduction.

Using the antenna axial illuminations, the main beam axis near-field gain reduction for a selected distance, d, from the antenna may be determined in the following manner:

1. Normalize the selected distance:

$$\Delta_h = d/(L_h^2/\lambda)$$

and

$$\Delta_v = d/(L_v^2/\lambda) \tag{5}$$

where:

$\Delta_v$, $\Delta_h$ - normalized distance associated with vertical and horizontal antenna dimensions, respectively.

2. From table A-3, find the near-field gain reduction factors ($\chi_h$ and $\chi_v$) using the axis illuminations and the $\Delta$ values. The total near-field gain reduction is the sum of the gain reductions, in dB, converted to absolute form:

$$\chi = 10^{-[(\chi_h + \chi_v)/10]} \qquad (6)$$

TABLE A-3. NEAR-FIELD CORRECTION FACTOR FOR RECTANGULAR APERTURE ANTENNAS

| FOR UNIFORM ILLUMINATION | | $\chi$, dB FOR ALL OTHER ILLUMINATIONS | | | | |
|---|---|---|---|---|---|---|
| $\Delta$ | $\chi$, dB | $\Delta$ | cos | $\cos^2$ | $\cos^3$ | $\cos^4$ |
| 0.016 | 17.50 | 0.010 | ---- | 14.0 | 12.6 | 11.5 |
| 0.020 | 17.25 | 0.015 | 14.0 | 12.4 | 11.0 | 9.8 |
| 0.021 | 15.95 | 0.020 | 13.2 | 11.2 | 9.7 | 8.6 |
| 0.023 | 16.75 | 0.025 | 12.2 | 10.2 | 8.8 | 7.8 |
| 0.026 | 14.75 | 0.030 | 11.4 | 9.5 | 8.0 | 7.0 |
| 0.032 | 15.65 | 0.035 | 10.8 | 8.8 | 7.4 | 6.4 |
| 0.037 | 13.50 | 0.040 | 10.2 | 8.2 | 6.8 | 5.8 |
| 0.042 | 15.00 | 0.050 | 9.2 | 7.2 | 5.8 | 4.8 |
| 0.056 | 11.95 | 0.060 | 8.4 | 6.6 | 5.0 | 4.0 |
| 0.065 | 13.10 | 0.070 | 7.6 | 5.8 | 4.4 | 3.4 |
| 0.093 | 9.0 | 0.080 | 7.0 | 5.3 | 3.8 | 2.8 |
| 0.14 | 11.0 | 0.090 | 6.6 | 4.7 | 3.4 | 2.4 |
| 0.4 | 2.2 | 0.10 | 6.1 | 4.2 | 3.0 | 2.1 |
| 1.0 | 0.3 | 0.15 | 4.3 | 2.6 | 1.6 | 1.2 |
| | | 0.20 | 3.0 | 1.6 | 0.9 | 0.6 |
| | | 0.25 | 2.0 | 1.1 | 0.6 | 0.4 |
| | | 0.30 | 1.5 | 0.8 | 0.4 | 0.3 |
| | | 0.35 | 1.1 | 0.6 | 0.2 | 0.2 |
| | | 0.40 | 0.8 | 0.4 | 0.2 | 0.2 |
| | | 0.50 | 0.5 | 0.2 | 0 | 0 |
| | | 0.60 | 0.4 | 0.2 | 0 | 0 |
| | | 0.70 | 0.2 | 0.1 | 0 | 0 |
| | | 0.80 | 0.2 | 0 | 0 | 0 |
| | | 0.9 | 0.1 | 0 | 0 | 0 |
| | | 1.0 | 0 | 0 | 0 | 0 |

NOTE: If the value of $\Delta$ falls between two points on the chart, then the value of $\chi$ should be interpolated.

3. Multiply the power density at distance, d, obtained using Equation 2, by $\chi$ as shown in Equation 4. Use Equation 3 to obtain the field strength from $P_D$.

Near-field Gain Reduction for Circular Aperture Antennas. The method employed with the near-field gain reduction factor for circular antennas is different than that for rectangular antennas. The axial power density at a given distance, d, from a circular aperture antenna is determined as follows:

1. Calculate the reference power density by substituting $2L^2/\lambda$ for distance in Equation 2. Or:

$$P_{D_0} = \frac{PG\lambda^2}{16\pi L^4} \tag{7}$$

where:

$P_{D_0}$ - reference power density, $W/m^2$

2. Convert distance, d, to normalized distance $\Delta = d\lambda/2L^2$.

3. Using the antenna illumination, obtain the value of $\chi$, the near-field gain reduction value, from table A-4, and multiply it by the reference power density to obtain the corrected power density. That is:

$$P_D = \chi P_{D_0} \tag{8}$$

4. Convert to field strength using Equation 3.

DEFINING AND DETERMINING THE VALUES OF NECESSARY PARAMETERS

The methodology for field strength calculations, discussed above, is dependent on the availability of technical information on the emitters. Specifically, the calculations require the following technical parameters: transmitter power, wavelength, antenna gain, antenna dimensions, and antenna beamwidths. These parameters are discussed below along with techniques for estimating parameter values if they are not available.

Peak Power

The peak power is used in calculating the peak field strength from the emitter. It is usually recorded as the maximum rated power at the output of the emitter before the antenna connections. It does not include system losses or losses due to equipment aging. If the peak power is not available, then no calculations can be made.

TABLE A-4. NEAR-FIELD CORRECTION FACTOR FOR CIRCULAR APERTURE ANTENNAS

| Δ | $\chi$ FOR ALL ILLUMINATIONS[1,2] UNIFORM | $(1-r^2)$ | $(1-r^2)^2$ | $(1-r^2)^3$ |
|---|---|---|---|---|
| 0.01 | 26.0 | 27.7 | 59.0 | 103.5 |
| 0.02 | 26.0 | 28.1 | 59.2 | 102.5 |
| 0.03 | 26.0 | 30.1 | 59.4 | 102.0 |
| 0.04 | 26.0 | 31.9 | 59.7 | 102.0 |
| 0.05 | 26.0 | 33.5 | 60.7 | 101.0 |
| 0.06 | 26.0 | 35.0 | 62.6 | 98.0 |
| 0.07 | 26.0 | 37.0 | 63.7 | 92.0 |
| 0.08 | 26.0 | 39.0 | 63.5 | 84.0 |
| 0.09 | 26.0 | 40.0 | 61.0 | 77.0 |
| 0.1 | 26.0 | 40.7 | 56.0 | 69.0 |
| 0.15 | 24.5 | 29.0 | 32.0 | 35.0 |
| 0.2 | 18.0 | 19.5 | 20.1 | 22.0 |
| 0.3 | 9.5 | 10.5 | 10.5 | 11.0 |
| 0.4 | 5.5 | 6.0 | 6.0 | 6.0 |
| 0.5 | 3.5 | 3.9 | 3.9 | 3.5 |

NOTES:

1. If the value of Δ falls between two points on the chart, then the value of $\chi$ should be interpolated.

2. For $\Delta > 0.5$, calculate field strengths using the far-field equation.

Average Power

The average field strength from the emitter is calculated from the average output power of the emitter. This is the maximum rated average power before the antenna connections. If this value is not available for a particular emitter, it can be calculated from the peak power and other parameters.

The average power is the product of the peak power and duty cycle. The duty cycle is a unitless number, less than or equal to 1.0, which is the product of the pulsewidth (PW) in seconds and the pulse repetition frequency (PRF) in Hertz. Emitter documentation often lists multiple values of PW and PRF or the PW and PRF as ranges. The duty cycle may be the product of some PW and some PRF in the range(s) but not necessarily the maximum value(s). If the precise duty cycle cannot be determined, then the maximum values may be used to obtain a conservative estimate of the duty cycle. For non-pulsed systems, the duty cycle is unity and the peak and average powers are the same.

Maximum Mainbeam Antenna Gain

As mentioned above, it is assumed that the aircraft is within the mainbeam of the emitting antenna for the purpose of calculating field strength. The mainbeam gain is the maximum possible gain of the antenna. If the gain is not available, it can be calculated from the area of the antenna aperture, the beamwidths, and the antenna illumination. The following procedures will show how to derive the antenna gain from one or more of these factors.

Calculating Gain of Aperture Antennas if Antenna Dimensions Are Known. If the dimensions of the antenna are known, then the gain can be calculated by application of some simplifying assumptions. Equation 9 is the basic formula for calculating gain:

$$G = \frac{4\pi^2 A\rho}{\lambda^2} \qquad (9)$$

where:

G - gain, unitless

A - area of antenna aperture, square meters

$\rho$ - antenna efficiency, unitless

F - Antenna gain factor, unitless

$\lambda$ - wavelength, meters

The antenna efficiency is a rating of how well the antenna emits the input power. Antenna efficiency is usually between 0.5 and 0.9, with a typical value of 0.55. If no other information is available, the typical value of 0.55 will be used for $\rho$. The product of antenna aperture and efficiency is known as the effective aperture, $A_e$, of the antenna. The antenna factor, F, is dependent on the type of aperture illumination used. If the aperture illumination is not known, uniform aperture illumination will be used because it will produce a conservative value of gain., For uniform illumination, F = 1.

Calculating Gain if Antenna Beamwidths Are Known. An approximation of gain from the known values of the horizontal and vertical beamwidths is:

$$G \approx \frac{20{,}000}{\theta_h \theta_v}$$

where:

$\theta_h, \theta_v$ - horizontal and vertical beamwidths, degrees

This assumes that the illumination is uniform and the antenna efficiency is 0.55.

Calculating Gain if Antenna Beamwidths and Dimensions Are Known. If the antenna dimensions and beamwidths are known, a more precise value for gain can be calculated. Assume an antenna efficiency of 0.55. Calculate the antenna aperture area. Then determine the gain factor from table A-5 for rectangular apertures and table A-6 for circular apertures. Substitute these values into Equation 9.

To determine the gain factor for rectangular apertures:

1. Use table A-5. Note that the horizontal illumination may be different than the vertical illumination, with a different gain factor for each dimension.

2. Observe between what values $\theta_h$ falls after substituting the appropriate values of $L_h$ and wavelength.

3. Find the gain factor, $F_h$, from the table.

4. Repeat steps 2 and 3 for $\theta_v$ and $L_v$ to obtain $F_v$.

5. Obtain the gain factor for the antenna from the product of the horizontal and vertical gain factors. That is, $F = F_h F_v$.

For circular apertures:

1. Use table A-6.

2. Observe the range of values of $\theta$ after substituting the appropriate values for diameter and wavelength.

3. Find the gain factor, F, from the table.

Calculating Gain if Antenna Illumination is Known. By itself, antenna illumination can only determine the gain factor (F). Either beamwidth or antenna dimension (for each axis, if the aperture is rectangular) is also required to calculate gain using the above procedure. Only one of the parameters is required because the other parameter can be derived using tables A-5 and A-6.

If all three parameters (i.e., illumination, beamwidths and dimensions) are known, then:

1. The Gain Factor(s) can be obtained using table A-5 or A-6.

2. Calculate $\theta$ using the formulas in the last column of the selected table and compare with the actual beamwidth(s). If the corresponding information does not agree, further checking on the available technical parameters may be required.

TABLE A-5. BEAMWIDTHS AND GAIN FACTORS OF RECTANGULAR ANTENNAS WITH VARIOUS ILLUMINATIONS

| ILLUMINATIONS | GAIN FACTOR, F | $\theta$, DEGREES |
|---|---|---|
| Uniform | 1.00 | 50.4λ/L to 68.7λ/L |
| Cosine | 0.81 | 68.7λ/L to 83.0λ/L |
| Cosine Squared | 0.667 | 83.0λ/L to 95.0λ/L |
| Cosine Cubed | 0.575 | 95.0λ/L to 110λ/L |
| Cosine Fourth | 0.515 | 110λ/L to 116λ/L |

TABLE A-6. BEAMWIDTHS AND GAIN FACTORS OF CIRCULAR ANTENNAS WITH VARIOUS ILLUMINATIONS

| ILLUMINATIONS | GAIN FACTOR, F | $\theta$, DEGREES |
|---|---|---|
| Uniform | 1.00 | 58.5λ/L to 72.8λ/L |
| $(1-r^2)$ | 0.75 | 72.8λ/L to 84.2λ/L |
| $(1-r^2)^2$ | 0.56 | 84.2λ/L to 94.5λ/L |
| $(1-r^2)^3$ | 0.44 | 94.5λ/L to 103.5λ/L |
| Cosine Fourth | 0.515 | 110λ/L to 116λ/L |

Median Gain. The median gain is sometimes used instead of mainbeam gain when calculating field strengths if antenna rotation is a factor. The median gain is the gain averaged over the entire rotation. Usually, median gain is used if an object if in the mainbeam of the antenna for too short a period of time. Yet, an aircraft may be in the mainbeam of the antenna for sufficient time to degrade or damage sensitive flight-critical systems aboard the aircraft. For example, an aircraft approaching for a landing at a typical velocity of 170 nautical miles per hour in the same direction that an antenna for an air search radar is rotating at a typical rate of 10 rotations per minute will be in the mainbeam of the antenna for one second. Damage to electronic equipment can occur during exposures of a few milliseconds. It is, therefore, recommended that mainbeam gain be used in field strength calculations instead of median gain.

Beamwidths and Antenna Dimensions

Beamwidths. In and of itself, beamwidths are not directly necessary for the calculation of field strengths, yet beamwidths serve to check or derive a value for gain, and to derive a value for antenna dimension.

Antenna Dimensions. The antenna length and width for rectangular apertures ($L_h$ and $L_v$), or diameter (L) for circular apertures is required in the calculation of field strength in the near field. Given that gain is known, the antenna dimension can be calculated based on specific assumptions. Fewer assumptions are required if more information about the antenna is known.

If only the gain is known: assume F = 1 and $\rho$ = 0.55. Solve Equation 9 for A, the antenna aperture area. Now assume a circular aperture and derive the diameter. The diameter is the dimension used in field strength calculations.

If the horizontal and vertical beamwidths are also knows: assume uniform illumination for each axis. For rectangular aperture antennas, derive the value of $L_h$ and $L_v$ using table A-5. For circular aperture antennas, derive a value of L from table A-6. Substitute the appropriate values of F and A into Equation 9 and derive the efficiency, $\rho$. If the efficiency is not between 0.5 and 0.9, repeat these steps with a different assumption for illumination (one or both axis for rectangular apertures) until the efficiency condition is met.

If beamwidths and illuminations are known: Derive the value(s) of L from table A-5 or A-6. Substitute the appropriate values of F and A into Equation 9 and derive the efficiency. If the efficiency is not between 0.5 and 0.9, check the values of the parameters for correctness.

Peak and Average Field Strength Predictions

Using the methods described above, the electric field strengths can be calculated from the data collected on the emitters in a particular environment. The distance at which the field strengths are to be calculated is dependent on the environment in question. A separate set of results for each of the environments discussed above is recommended. Using peak power, the maximum peak field strengths can be calculated for the emitters. Average transmitted power

is used to calculate the maximum average electric field strength. The resultant electric field strength predictions can be tabulated in preparation for producing envelope graphs.

PRESENTATION OF RESULTS

The most convenient method of presenting the maximum electric field strength predictions is in the form of envelope graphs. First, the results of the analysis are plotted on log-log paper. The abscissa is frequency from 10 kHz to 100 GHz. The ordinate is field strength in volts per meter (V/m) with a range from 1 V/m to some power of 10 that is greater than the greatest value to be plotted on the graph. The resultant graph is a histogram. An example of a histogram is given in figure A-2. (Note that the value plotted in figures A-2 through A-6 and table A-7 are random and are merely used for illustration purposes.)

The "envelope" of the graph is a discontinuous line that identifies the level at which all of the values on the graph are beneath. The data in figure A-2 with an envelope over it is shown in figure A-3. Figure A-4 is just the envelope of the values. As can be seen, the values that produced the electric field strength envelope cannot be determined from figure A-4. The values of the levels in the envelope graphs are shown in table A-7. Note that such tables are required for the establishment of an international standard.

The results of both the total environment analysis and the takeoff and landing analysis is a composite of several environments. The composite graph merely follows the maximum value of the component envelope graphs. For example, the envelope graph of figure A-4 is combined with the envelope graph of figure A-5 to produce the composite graph is figure A-6. For the total environment, as many as 12 envelope graphs are produced and combined into two composite graphs, one for peak and one for average field strength. For the takeoff and landing environment, two graphs for the environments around each of the airports are produced. In the U.S. case, six airports were examined resulting in 12 graphs. The two composite graphs that are produced for the takeoff and landing scenario are comprised of these 12 individual graphs.

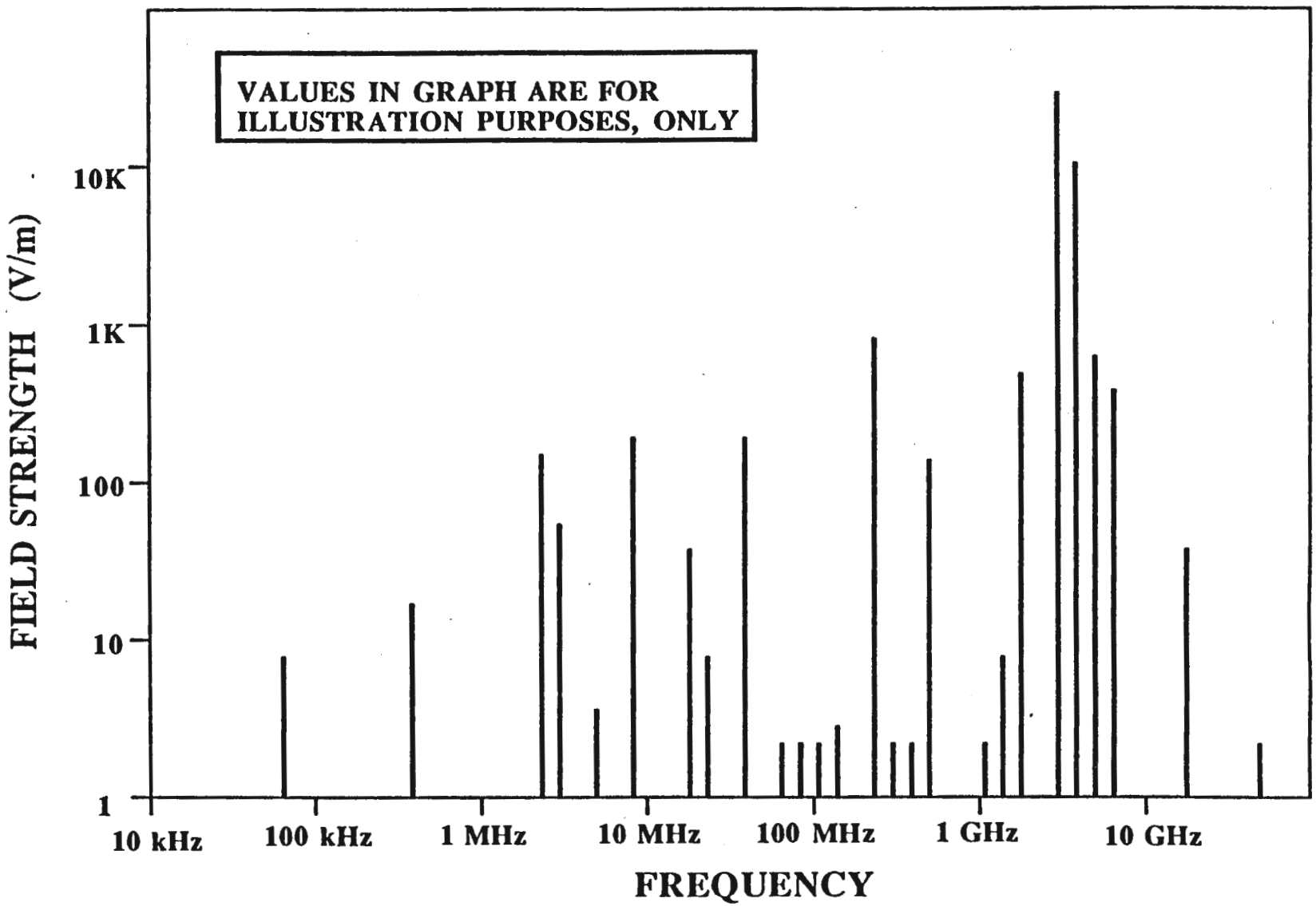



FIGURE A-2. EXAMPLE OF ELECTRIC FIELD STRENGTH VERSUS FREQUENCY GRAPH ON A LOG-LOG SCALE

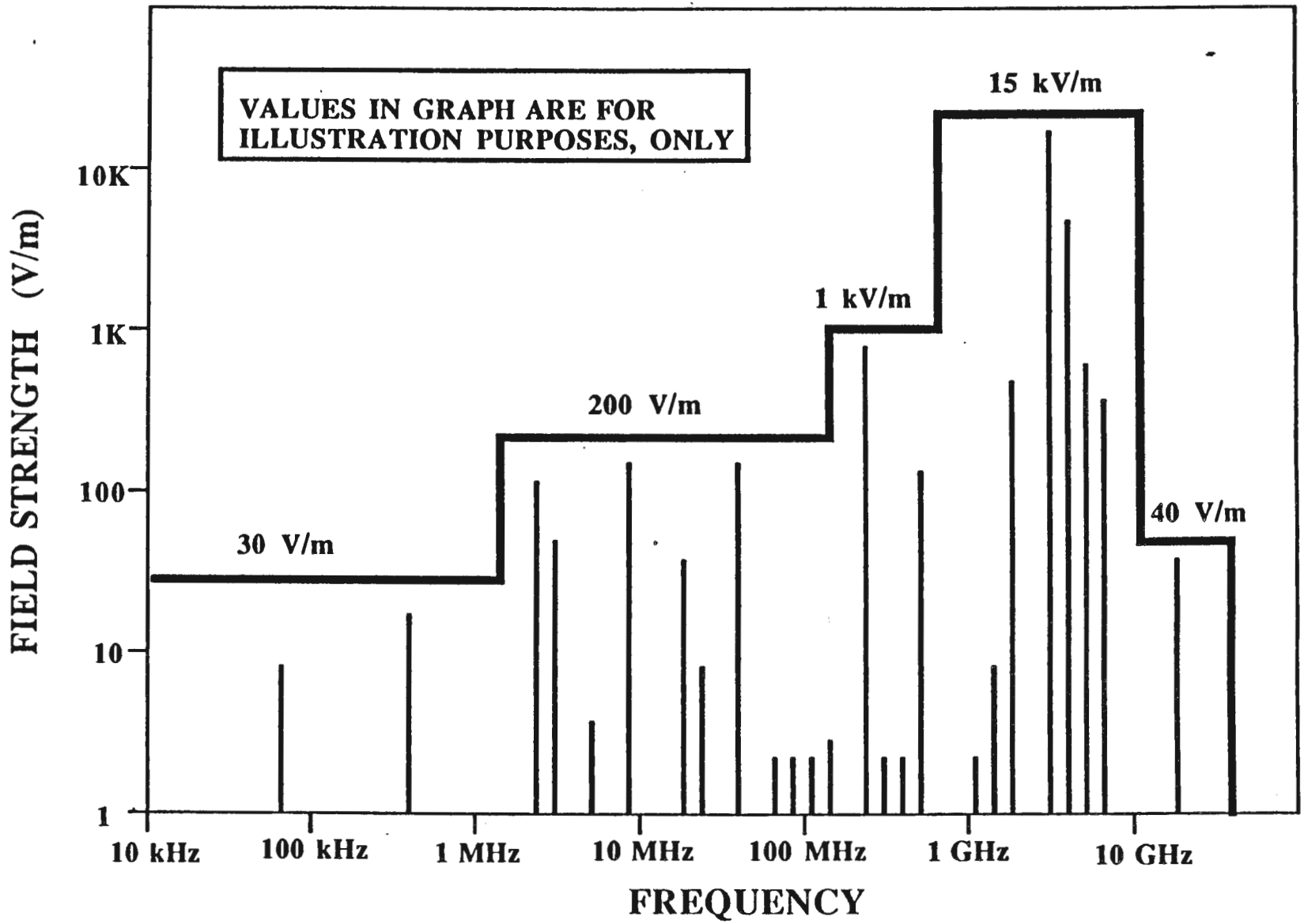



FIGURE A-3. EXAMPLE OF ENVELOPE OF FIELD STRENGTH VERSUS FREQUENCY GRAPH

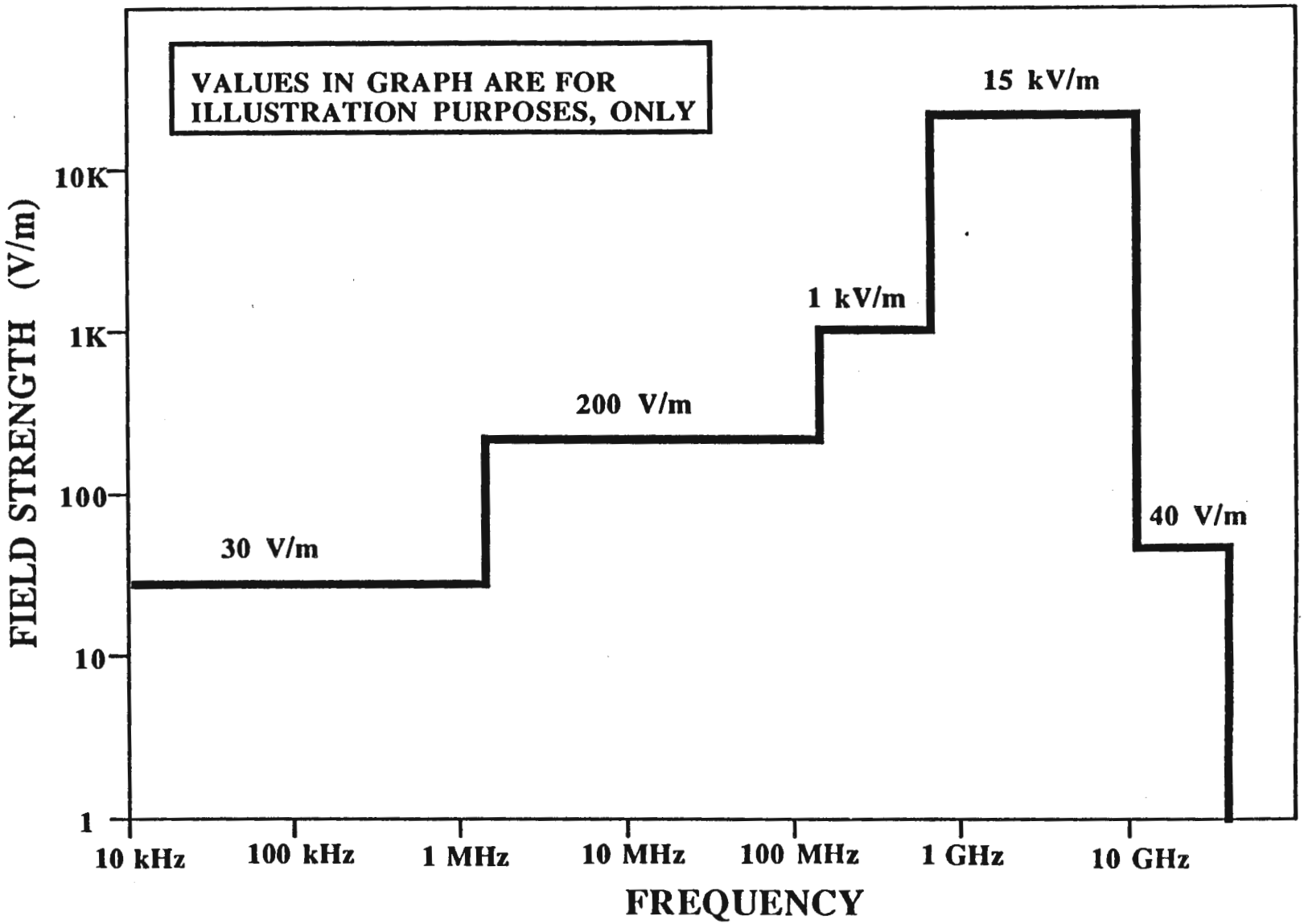



FIGURE A-4. EXAMPLE OF FINAL OUTPUT

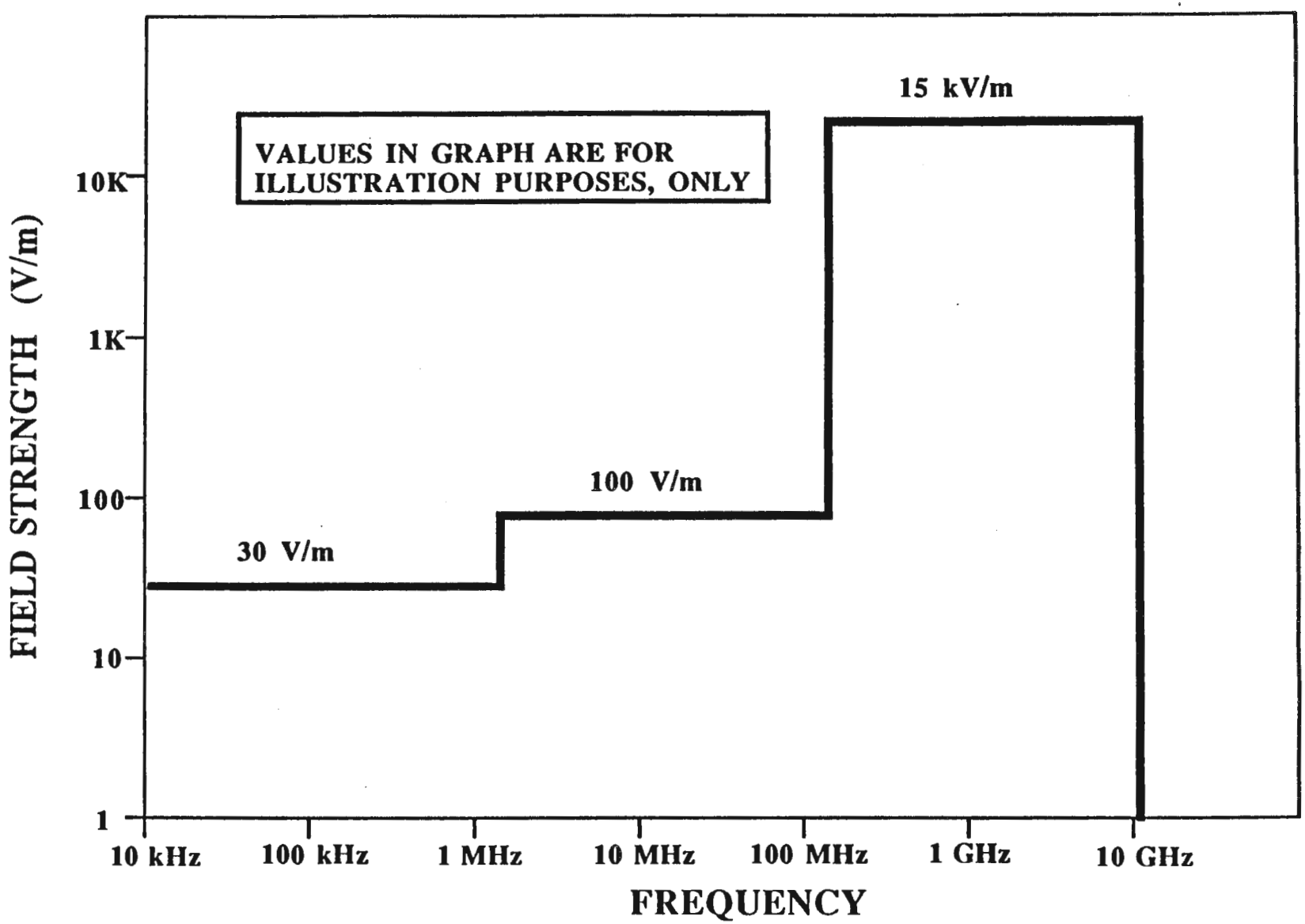



FIGURE A-5. EXAMPLE OF ENVELOPE GRAPH OF A SECOND ENVIRONMENT, TO BE COMBINED WITH FIGURE A-4 TO FORM A COMPOSITE GRAPH

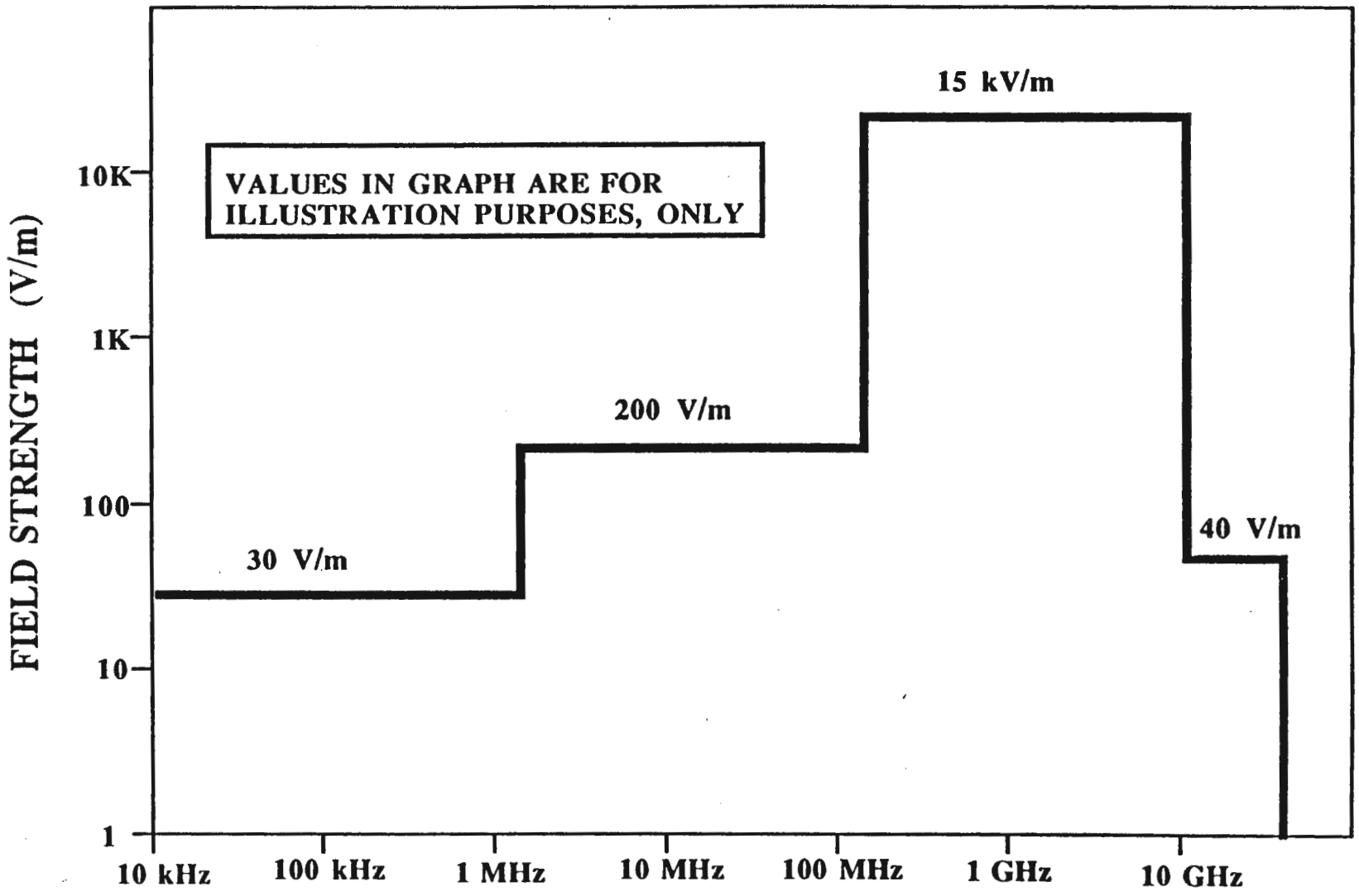



FIGURE A-6. COMPOSITE OF TWO ENVIRONMENTS SHOWN IN FIGURES A-4 AND A-5

TABLE A-7. SAMPLE ELECTRIC FIELD STRENGTH LEVELS USING VALUES IN FIGURE A-4

| FREQUENCY | FIELD STRENGTH (V/m) |
|---|---|
| 10 kHz - 2 MHz | 30 |
| 2 MHz - 100 MHz | 200 |
| 100 MHz - 800 MHz | 1000 |
| 800 MHz - 10 GHz | 15,000 |
| 10 GHz - 40 GHz | 40 |

NOTE: Values in table are for illustrative purposes, only.

## SUMMARY

The intention of this chapter was to provide the guidelines that were used to calculate the maximum electromagnetic radiation level that would be experienced by an aircraft flying in the United States. These guidelines are presented in an effort to promote uniformity of results from other members of the international aviation community in order that all results can be combined and used to develop an international RF susceptibility standard. Two aircraft scenarios were discussed: a worst-case scenario and a takeoff and landing scenario. The logic used by the United States to develop minimum distance criteria for each scenario was presented. The techniques used to calculate the far field and near field electric field strengths were described. Emitter parameters and methods that can be used to estimate values for parameters that are not available were discussed. Finally, how to present the results in the form of envelope graphs (useful because the results can be displayed in a simple chart without identifying the emitters involved) and field strength tables was shown.

## BIBLIOGRAPHY

DOD/ECAC, Guidelines for Developing Maximum Peak and Average Field Strength Envelope Graphs for Aircraft, Sept. 1987.

MAC Test 15-117-87, Electromagnetic Radiation Measurement, 7 June 1988.

MIL-STD-235-1A, Electromagnetic (Radiated) Environment Considerations for Design and Procurement of Electrical and Electronic Equipment.

MIL-STD-461B, Electromagnetic Emission and Susceptibility Requirements for the Control of Electromagnetic Interference.

MIL-STD-462, Measurement of Electromagnetic Interference Characteristics.

RTCA DO-160B, Environmental Conditions and Test Procedures for Airborne Equipment.

## LIST OF ACRONYMS

| | |
|---|---|
| AC | Advisory Circular |
| AM | Amplitude Modulation |
| ATCRBS | Air Traffic Control Radar Beacon System |
| AWACS | Airborne Warning and Control System |
| CAA | Civil Aviation Authority |
| CONUS | Contiguous United States |
| dBi | Decibels with respect to one milliampere |
| DGAC | Directorate Generale Aviation Civile |
| DOD | Department of Defense |
| ECAC | Electromagnetic Compatibility Analysis Center |
| ECM | Electronic Counter Measures |
| EMC | Electromagnetic Compatibility |
| EME | Electromagnetic Environment |
| EMR | Electromagnetic Radiation |
| EUROCAE | European Organization for Civil Aviation Electronics |
| FAA | Federal Aviation Administration |
| FM | Frequency Modulation |
| ft | feet |
| HF | High Frequency |
| HERF | High Energy Radio Frequency |
| ICAO | International Civil Aviation Organization |
| IFF | Identify Friend or Foe |
| kHz | kilohertz |
| MHz | megahertz |
| NATO | North Atlantic Treaty Organization |
| nmi | nautical mile |
| NPRM | Notice of Proposed Rulemaking |
| RF | Radio Frequency |
| RTCA | Radio Technical Commission for Aeronautics |
| SAE | Society of Automotive Engineers |
| SIF | Selective Identification Facility |
| SMOTEC | Special Missions Operation Test and Evaluation Center |
| STANAG | Standardization Agreement (NATO) |
| TACAN | Tactical Air Navigation |
| TV | Television |
| UHF | Ultra High Frequency |
| U.K. | United Kingdom |
| U.S. | United States |
| VHF | Very High Frequency |
| VLSI | Very Large Scale Integration |
| V/m | Volt/meter |
| VOR | VHF Omnidirectional Range |

Department of Transportation
Federal Aviation Administration

DOT/FAA/CT-88/10

# HANDBOOK-VOLUME II
# DIGITAL SYSTEMS VALIDATION

## CHAPTER 15
## ELECTROMECHANICAL ACTUATOR SYSTEMS
## (ELECTRICAL SYSTEMS CERTIFICATION ISSUES)

PREPARED BY:

COMPUTER RESOURCE MANAGEMENT, INC.
950 HERNDON PARKWAY, SUITE 360
HERNDON, VIRGINIA 22070

PREPARED FOR:

FEDERAL AVIATION ADMINISTRATION
TECHNICAL CENTER
ATLANTIC CITY INTERNATIONAL AIRPORT, NEW JERSEY 08405

NOTICE

This document is disseminated under the sponsorship of the U.S. Department of Transportation in the interest of information exchange. The United States Government assumes no liability for the contents or use thereof.

The United States Government does not endorse products or manufacturers. Trade or manufacturers' names appear herein solely because they are considered essential to the objective of this report.

TABLE OF CONTENTS


| Section | | Page |
|---|---|---|
| 1.0 | Electromechanical Actuation Systems Technology | 15-1 |
| 1.1 | Introduction and Scope | 15-1 |
| 1.2 | Emerging Technology | 15-1 |
| 1.2.1 | Electromechanical Actuation Systems for V-22 | 15-2 |
| 1.2.2 | All-Electric Airplane | 15-2 |
| 1.3 | Technology Needs for an Aerospace Plane | 15-3 |
| 1.4 | Electromechanical Actuation Systems Trends | 15-3 |
| 2.0 | Smart Servos and Actuators | 15-5 |
| 2.1 | Electrohydraulic Actuators | 15-5 |
| 2.1.1 | Reliability and Redundancy of Electrohydraulic Actuators | 15-7 |
| 2.1.2 | Detailed Examples of Current Technology | 15-15 |
| 2.1.3 | Detailed Examples of Advanced Technology and Research | 15-20 |
| 2.2 | All-Electric Actuators | 15-35 |
| 2.2.1 | Actuation Concepts | 15-35 |
| 2.2.2 | Active Isolation | 15-38 |
| 2.2.3 | Gearbox Technology | 15-38 |
| 2.2.4 | Reliability | 15-38 |
| 2.3 | Actuation Reliability and Redundancy Technology | 15-39 |
| 2.3.1 | Failure Detection | 15-41 |
| 2.3.2 | Fault Isolation | 15-42 |
| 2.3.3 | Failure Management | 15-43 |
| 3.0 | Digital Engine Control Reliability and Redundancy | 15-45 |
| 3.1 | Reliability Models for FADEC Systems | 15-45 |
| 3.1.1 | Detailed State Model For FADEC | 15-45 |

TABLE OF CONTENTS


| Section | | Page |
|---|---|---|
| 3.1.2 | Markov Model | 15-48 |
| 3.1.3 | Transition Probabilities | 15-49 |
| 3.1.4 | Effect of Maintenance | 15-53 |
| 3.1.5 | Markov Models for System Safety and Availability | 15-56 |
| 3.2 | Analytical Redundancy Design for Improved Engine Reliability | 15-58 |
| 3.2.1 | Analytical Redundancy Engine Control Concept | 15-60 |
| 3.2.2 | Component Tracking Filter | 15-61 |
| 3.2.3 | Failure Detection And Isolation | 15-62 |
| 3.2.4 | Failure Management | 15-63 |
| 3.2.5 | Test Results | 15-65 |
| 4.0 | Primary and Secondary Electrical Power | 15-69 |
| 4.1 | Aircraft Power Systems | 15-69 |
| 4.1.1 | Power System Reliability Requirements | 15-69 |
| 4.1.2 | Electrical Power System Elements | 15-70 |
| 4.1.3 | Technology Trends | 15-72 |
| 4.2 | Power Systems Detailed Examples | 15-75 |
| 4.2.1 | Fault Protection | 15-75 |
| 4.2.2 | High Voltage dc Generation and Distribution | 15-75 |
| BIBLIOGRAPHY | | 15-81 |
| ACRONYMS AND ABBREVIATIONS | | 15-85 |

LIST OF ILLUSTRATIONS


| Figure | | Page |
|---|---|---|
| 1.1-1 | ELECTROMECHANICAL ACTUATION ELEMENTS | 15-1 |
| 1.2-1 | V-22 ACTUATION SYSTEMS | 15-3 |
| 2.1-1 | SINGLE POWER RAM HYDRAULIC ACTUATOR | 15-5 |
| 2.1-2 | DUAL TANDEM POWER RAM ACTUATOR | 15-6 |
| 2.1-3 | TYPICAL OPERATIONAL DUAL TANDEM ACTUATOR | 15-7 |
| 2.1-4 | OPERATION OF ELECTROHYDRAULIC SERVOVALVE | 15-8 |
| 2.1-5 | DIRECT DRIVE DUAL TANDEM ACTUATOR | 15-9 |
| 2.1-6 | F-16 INTEGRATED SERVO ACTUATOR | 15-16 |
| 2.1-7 | F-18 STABILATOR ACTUATOR | 15-18 |
| 2.1-8 | V-22 SWASHPLATE ACTUATOR | 15-20 |
| 2.1-9 | SINGLE FAIL OPERATE DIRECT DRIVE ACTUATOR | 15-22 |
| 2.1-10 | DYNAMIC CONTROLS' F-16 DIRECT DRIVE ACTUATOR | 15-23 |
| 2.1-11 | MOVING COIL TORQUE MOTOR | 15-24 |
| 2.1-12 | BELL-4VALVE ACTUATOR | 15-26 |
| 2.1-13 | BOEING/MOOG DISAC ACTUATOR | 15-27 |
| 2.1-14 | DYNAMIC CONTROLS' ALL DIGITAL SERVOVALVE ACTUATOR | 15-29 |
| 2.1-15 | SOLENOID POPPET VALVE | 15-30 |
| 2.1-16 | PIEZOELECTRIC SERVOVALVE | 15-32 |
| 2.1-17 | DYNAMIC CONTROLS FLUTTER DAMPING ACTUATOR | 15-34 |
| 2.2-1 | ELECTRIC ACTUATOR CONCEPTS | 15-36 |
| 2.2-2 | SINGLE MOTOR HINGELINE ACTUATOR | 15-37 |
| 2.2-3 | DUAL MOTOR HINGELINE ACTUATOR | 15-37 |
| 2.3-1 | FAULT DIAGNOSIS AND FAILURE MANAGEMENT | 15-40 |
| 3.1-1 | DUAL REDUNDANT ENGINE CONTROLLER | 15-47 |
| 3.1-2. | STATE TRANSITIONS FOR ENGINE CONTROLLER OUT OF STATE 1 | 15-51 |
| 3.1-3 | NUMBER OF SHUTDOWNS PER MILLION HOURS OPERATION - DUPLEX | 15-55 |
| 3.1.4 | SENSITIVITY OF INFLIGHT SHUTDOWN TO COVERAGE | 15-55 |
| 3.1-5 | NUMBER OF SHUTDOWNS PER MILLION HOURS OPERATION - TRIPLEX | 15-59 |
| 3.2-1 | SCHEMATIC OF ANALYTICAL REDUNDANCY ENGINE CONTROLLER | 15-60 |
| 3.2-2 | COMPONENT TRACKING FILTER | 15-63 |
| 3.2-3 | STEADY STATE ERROR DETECTION - FAN SPEED SENSOR | 15-64 |
| 3.2-4 | DYNAMIC ERROR DETECTION - FAN SPEED SENSOR | 15-64 |
| 3.2-5 | CORRECTIVE ACTION - FAN SPEED SENSOR | 15-68 |
| 3.2-6 | CORRECTIVE ACTION - FUEL FLOW ACTUATOR SENSOR | 15-68 |
| 4.1-1 | ISOLATED CHANNEL POWER SYSTEM | 15-71 |
| 4.1-2 | RECONFIGURABLE POWER SYSTEM | 15-72 |
| 4.1-3 | INTERRUPTIBLE POWER SUPPLY METHODS | 15-74 |
| 4.2-1 | SOLID STATE POWER CONTROLLER SWITCHING | 15-76 |
| 4.2-2 | ADVANCED AIRCRAFT ELECTRICAL SYSTEM DIAGRAM | 15-77 |

LISTS OF TABLES


| Table | | Page |
|---|---|---|
| 2.1-1 | FAILURE DETECTION AND MANAGEMENT ON SIX SELECTED ACTUATORS | 15-12 |
| 2.2-1 | CANDIDATE ACTUATOR FAILURE MODES COMPARISON | 15-39 |
| 3.1-1 | COMPONENTS REQUIRED FOR ENGINE CONTROL MODES | 15-48 |
| 3.1-2 | MARKOV MODEL STATE DEFINITIONS | 15-50 |
| 3.1-2 | STATE TRANSITIONS FOR ENGINE CONTROLLER OUT OF STATE 1 | 15-51 |
| 3.1-3 | PARAMETERS FOR TRANSITIONS OUT OF STATE 2 | 15-52 |
| 3.1-4 | BASE LINE COMPONENT FAILURE RATES | 15-54 |
| 3.2-1 | ACTUATOR FAILURE ACCOMMODATION TEST RESULTS | 15-66 |
| 3.2-2 | SOFT FAILURE TEST SUMMARY | 15-67 |
| 4.2-1 | HVDC VS CONVENTIONAL POWER | 15-78 |

# 1.0 Electromechanical Actuation Systems Technology

## 1.1 Introduction and Scope

This chapter covers the technological description of electromechanical actuation systems (EMAS) including technology trends in these devices and certification issues involving actuators, electric power, and digital engine controls. The scope of this chapter, illustrated in figure 1.1-1, covers actuators, engine controls, and electrical power. The present electromechanical actuation technology is reviewed in section 2; particularly actuator devices for use on advanced technology aircraft. Technology and issues related to digital engine controls are included in section 3. Electric power systems technology is discussed in section 4 with emphasis on the technology trends of future generation aircraft.

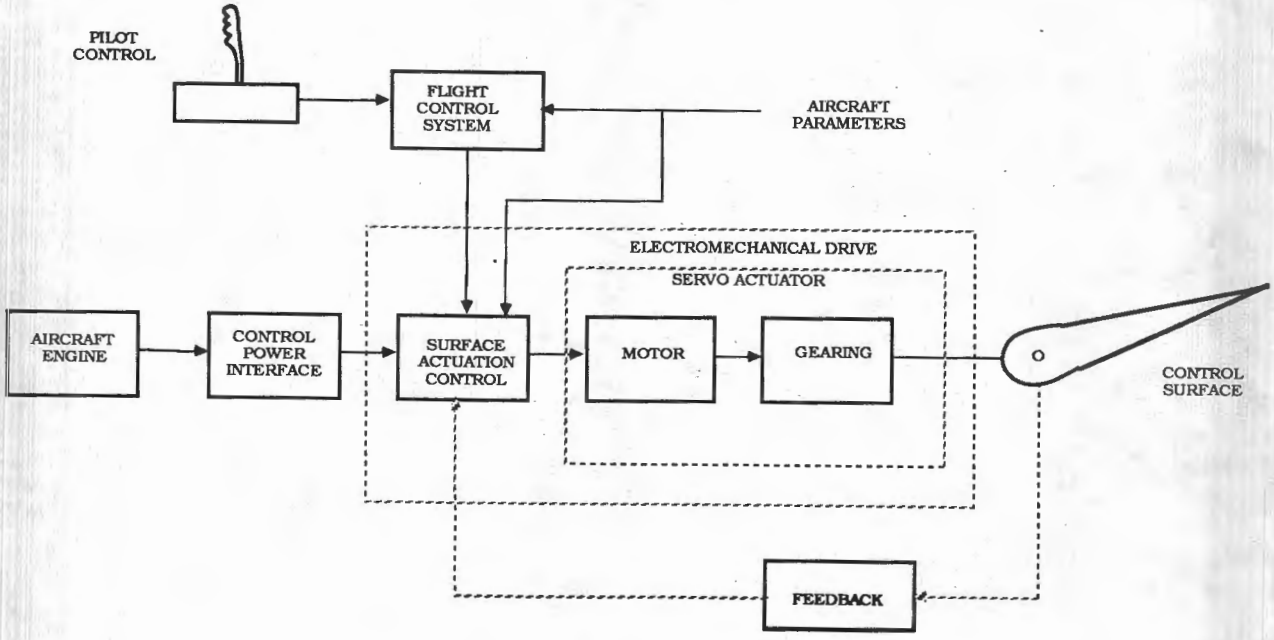



FIGURE 1.1-1. ELECTROMECHANICAL ACTUATION ELEMENTS

## 1.2 Emerging Technology

Areas of emerging advanced technology aircraft include the V-22 tilt rotor, all weather aircraft. This aircraft was originally intended for use by the military, but could find application in commercial operations, perhaps as a local interurban feeder to large airport hubs like the Los Angles, Dallas-Ft.

Worth, New York, Chicago, and Washington, D.C. areas. Many other potential application areas could be found within the U.S. and internationally.

Another important area of future application includes faster commercial transports, made either by the U.S. or international consortiums. There is renewed interest by Aerospatiale in developing the British/French Concorde into a larger faster model. The National Aeronautics and Space Administration (NASA) has initiated research on an aerospace plane that has captured the interest of U.S. industry and has achieved limited congressional funding.

Any of these applications will require systems having higher reliability than today's EMAS while operating in far more severe thermal, shock, and vibration environments.

### 1.2.1 Electromechanical Actuation Systems for V-22

One view is that the V-22 aircraft is a flying actuator (refer to figure 1.2-1 from McManus 1985). Actuation systems are required for thrust control, flight control and stability augmentation, and configuration change-over from vertical to horizontal flight.

This aircraft may tolerate failures for a very short time during configuration changes or even during normal flight for some systems. Requirements for reliability and redundancy for commercial service should be more severe than for more conventional configurations of transport aircraft, due to flights over populated areas and the unique variable configuration.

### 1.2.2 All-Electric Airplane

The use of electric power rather than hydraulic power for actuator systems has several advantages (Leonard 1983, AiResearch 1976). An industry trend toward high voltage (270 volt dc) distribution, proposed and investigated by the Naval Air Development Center and several major airframe companies, may make it possible to operate actuators and control systems without hydraulics (Perkins and Marek 1977). This trend is based on the advent of permanent magnetic motors providing high power in a small space. The maintenance and repair of such all-electric systems would be simplified. For example, no hydraulic system bleeds or pressurization would be needed when maintaining or repairing an actuator.

Since electric power is more easily distributed, redundancy could be implemented by having several actuators rather than a single one driving a torque tube connected to several surfaces.

Three types of all-electric actuators that have been developed and evaluated include linear, rotary, and rotary-hingeline. Although electromagnetic actuation systems could eventually surpass electrohydraulic reliability, the results to date indicate that electrical actuator reliability has not been much improved over hydraulic actuators.

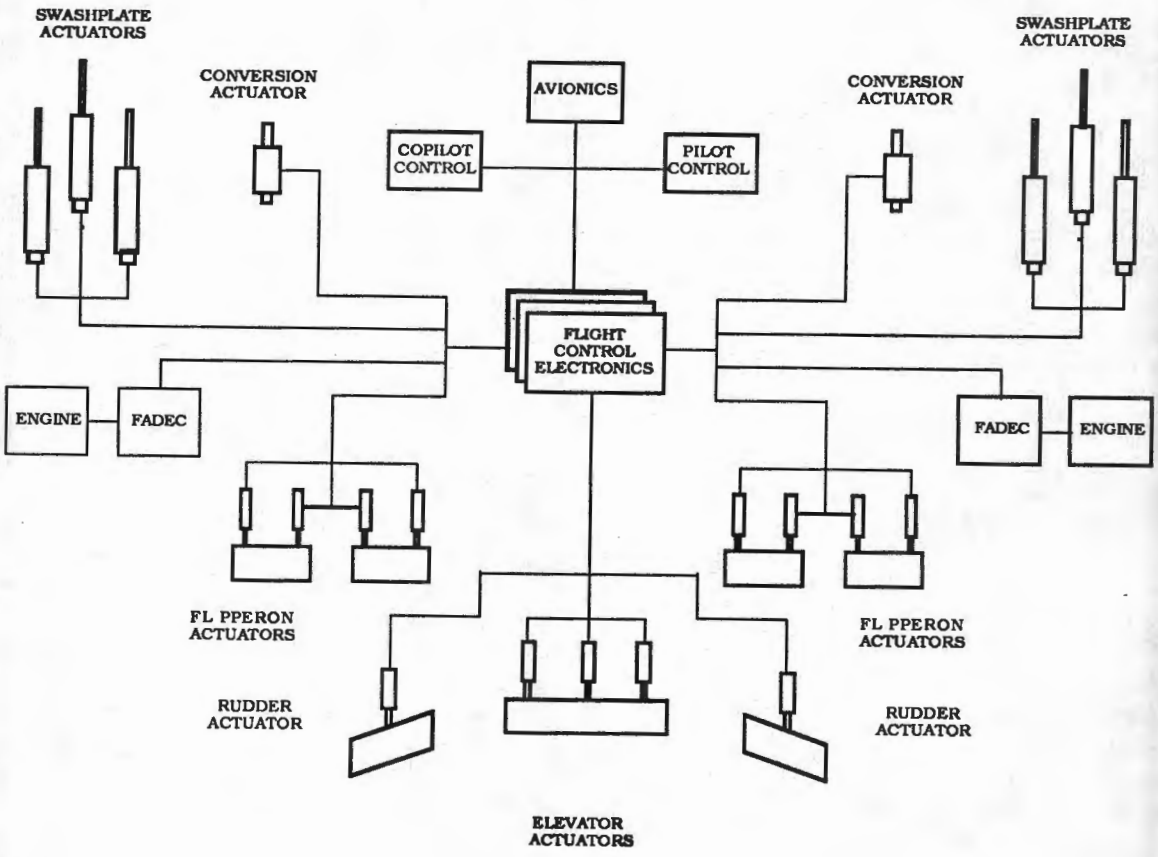



FIGURE 1.2-1. V-22 ACTUATION SYSTEMS

## 1.3 Technology Needs for an Aerospace Plane

An aerospace plane, described in Jorgensen 1983, is capable of flying intercontinental distances in commuter aircraft times. This imposes the most severe requirements for reliability and redundancy. Safe flight and immediate landing following a critical system failure are not possible from low earth orbit altitudes. In addition, a principal economic advantage of this aircraft lies in having flight operations with a round trip in a single crew day. This requires a low factor of unplanned maintenance. A two-hour turn-around is anticipated for the aircraft.

Flight operation at low earth orbit or suborbital altitudes requires flight critical systems having two orders of magnitude greater reliability than for present aircraft. Achieving these levels of reliability makes fault detection, isolation, and reconfiguration a necessity.

## 1.4 Electromechanical Actuation Systems Trends

In support of aircraft technology trends, such as those highlighted in this section, actuation system technology needs more reliable devices. One way of

accomplishing increased reliability is by using simpler devices. Current research and development is attempting to increase reliability as well as find ways for effectively applying computer technology. This is particularly true for the direct drive hydraulic actuators and electric actuators described in section 2. Military system applications have higher operating pressures for hydraulic systems and thus smaller and lighter actuator systems. This should also affect commercial actuation systems. The trend toward high voltage electric distribution is aided by a shift toward 270 volt dc power as an answer for providing an uninterruptable source of dc power for onboard computers. Newer engine control applications require additional computers on the engines. These additional computers provide automatic engine optimization (fuel versus power) during flight.

In addition to the development of the electromechanical actuation devices, research efforts by NASA Langley are also underway. NASA research will develop and demonstrate the means for controlling sensitive computer devices that must operate in severe aircraft electrical environments (i.e., electrical noise from onboard systems and from external sources of noise such as Electromagnetic Interference (EMI), lightning, and High-Intensity Radiated Fields (HIRF) environments). A multiyear advanced technology demonstration program is planned by NASA Langley to develop and demonstrate the technology for application of fiber optic devices as needed for the control of onboard computer devices. The program emphasizes Fly-By-Light and Power-By-Wire (FBL/PBW).

Fiber optic technology holds the promise for assuring that sensitive digital data is immune to electrical interference from external sources. The NASA program will include development and flight test of fiber optic devices to provide interconnects for digital control signals. This will be in the form of a fiber optic network data bus. Present computer devices may be interfaced to the network by bus interface devices that convert the electrical signals to optical signals. The electrical power to move surfaces and actuators will likely be provided by electrical means as for present systems.

## 2.0 Smart Servos and Actuators

A wide range of devices are currently available to operate the control surfaces on advanced technology aircraft. Improvements in hydraulic control devices and control configurations for fault detection and isolation greatly improve the reliability and performance of these devices. A realm long held by hydraulic actuators is being invaded by all-electric devices for control and actuation. This is due to the discovery of rare earth permanent magnetic materials. These materials allow operation of high power electric motors with small size and weight, and low standby power requirements.

### 2.1 Electrohydraulic Actuators

Hydraulic actuators operate via pressure applied to a power ram piston that converts the pressure to a linear or rotary motion. The positioning of a control surface is accomplished by directing the force into a mechanical linkage to the surface. The force is altered by directing pressure onto either side of a piston.

A simplex actuator, shown in figure 2.1-1, relies on only one hydraulic fluid and a single power ram piston. This configuration is widely used for ground based equipment and for some non-critical aircraft applications. It is not reliable enough for aircraft control in flight critical systems. While other parts of the actuator may be fault tolerant, any failure of the power ram or hydraulic system will disable the actuator. Therefore, for flight critical applications, power rams have two pistons in tandem, as shown in figure 2.1-2. Each piston is supplied from a separate hydraulic system. These actuators are designated as dual tandem actuators.

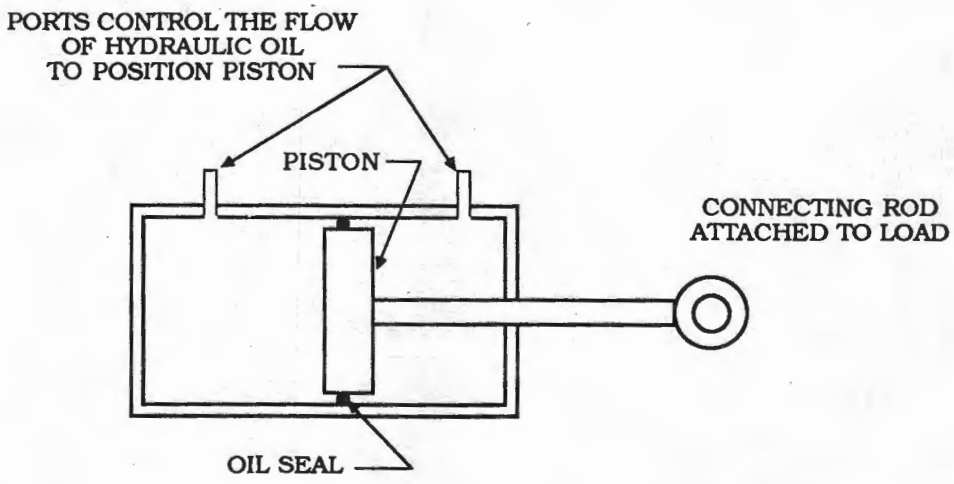



FIGURE 2.1-1. SINGLE POWER RAM HYDRAULIC ACTUATOR

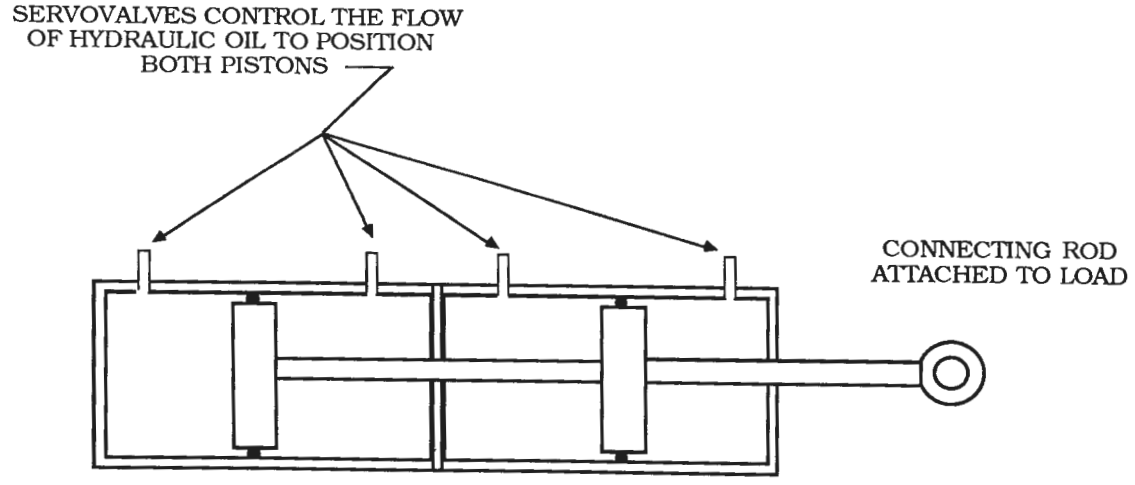



FIGURE 2.1-2. DUAL TANDEM POWER RAM ACTUATOR

A typical operational configuration for controlling a dual tandem actuator is illustrated in figure 2.1-3. These actuators use several stages to convert and amplify an electrical signal or mechanical input into useful motion. While current dual tandem actuators may differ from this diagram in some manner, the diagram conveys an understanding of the operation.

The first stage normally consists of three or four jet pipe or flapper valves that convert the input into a differential pressure for driving the second stage servovalve spool. The first two stages are often combined into a single unit called a two-stage electrohydraulic servovalve. With these devices, the spool position is controlled by feedback (normally mechanical) of the spool position to the first stage. The schematic and operation of a two-stage electrohydraulic control valve is shown in figure 2.1-4.

The second stage supplies hydraulic fluid to modulator pistons or servo ram pistons which in turn position the dual main control valve. The second stage may alternatively position the main control valve mechanically. Considerable force can be developed by these actuators (tens of thousands of pounds for large actuators). Because of limited power gain, the force to move the main control valve is also large. This two-stage control of the hydraulic flow to the main ram piston is the normal mode of operation. Closed loop control of the power ram position is often used. This control may be implemented by mechanical, analog, or digital processes. For mechanical feedback, the linear variable differential transformers (LVDTs) are replaced by mechanical linkages.

One experimental class of actuators consists of systems for eliminating the main control valve. These systems use more powerful 2-4 stage electrohydraulic control valves, or even electric motors, to operate the main control valve directly, as shown in figure 2.1-5.

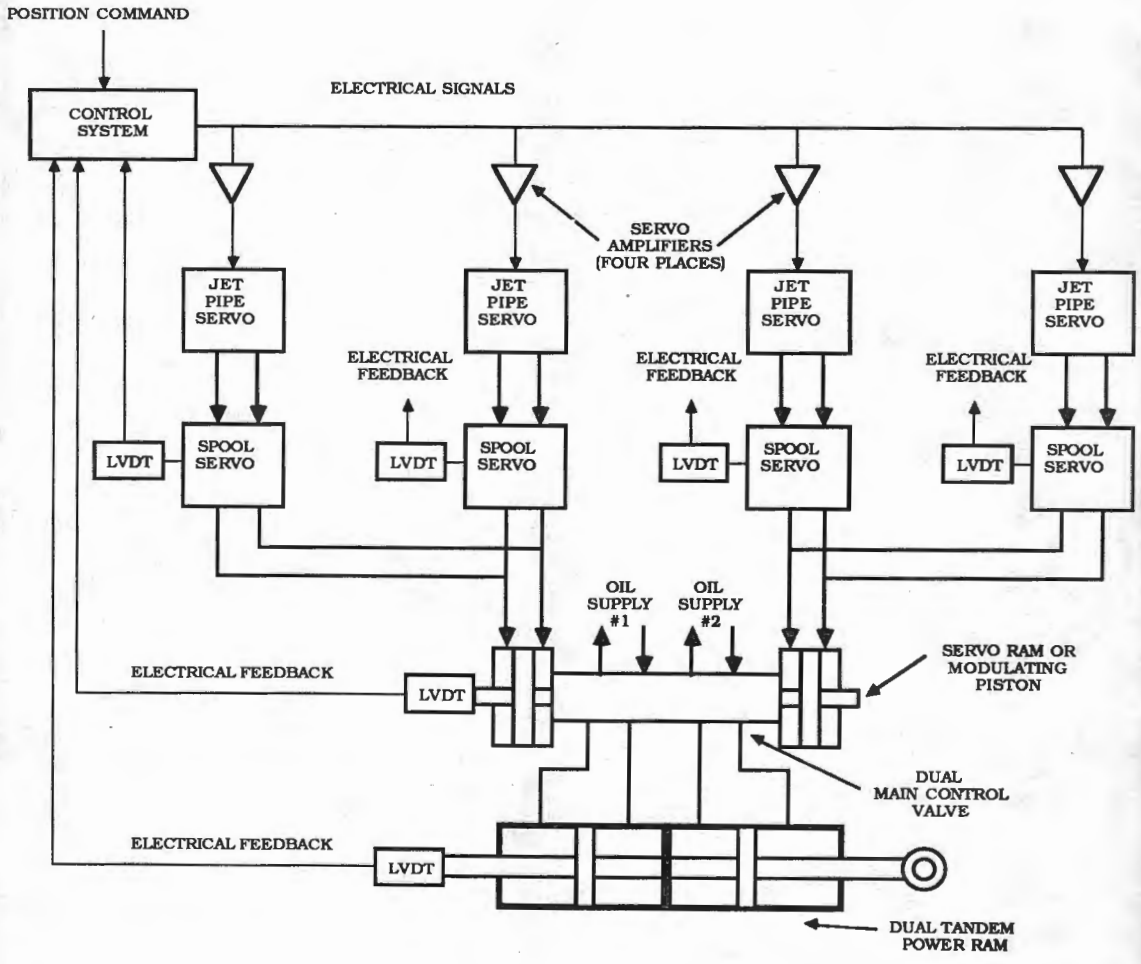
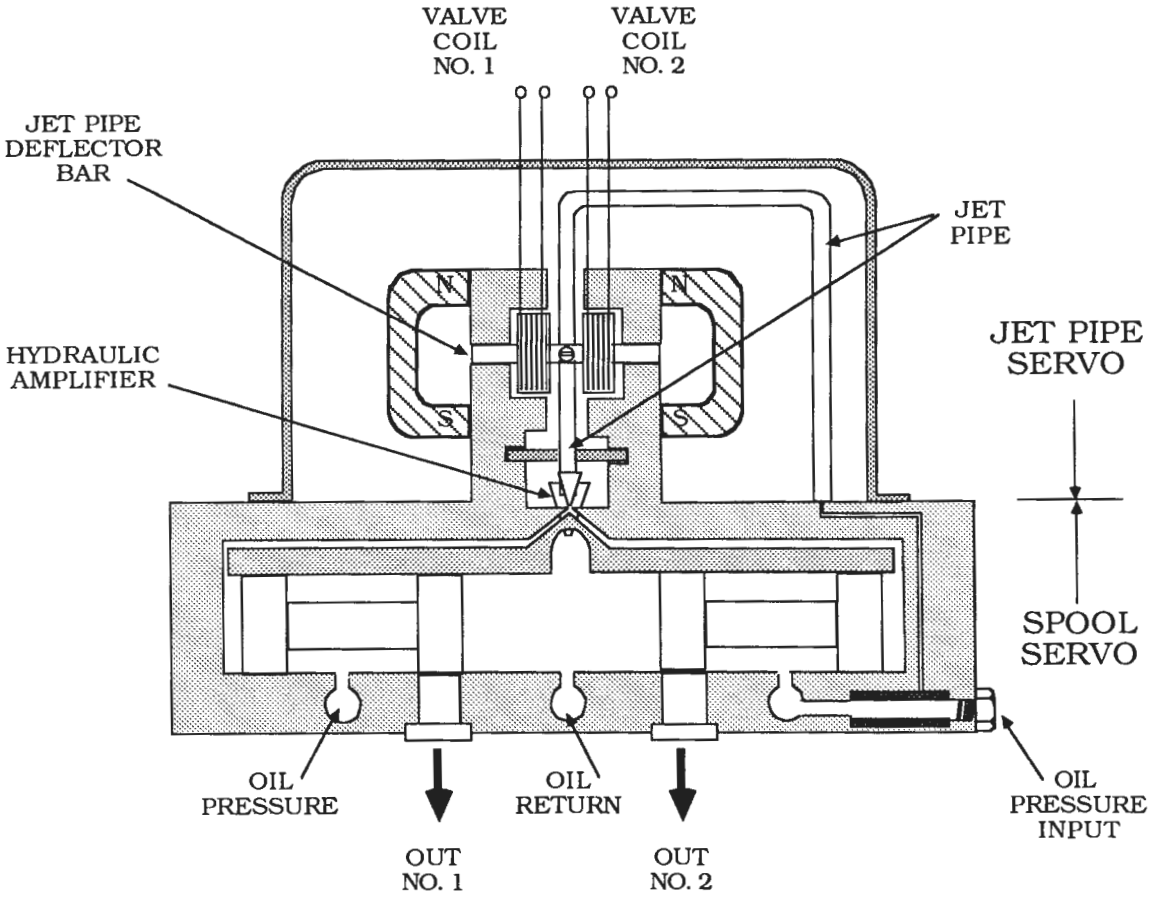



FIGURE 2.1-3. TYPICAL OPERATIONAL DUAL TANDEM ACTUATOR

### 2.1.1 Reliability and Redundancy of Electrohydraulic Actuators

Fault detection and isolation systems vary widely among actuator systems. These variations are the result of design decisions and design philosophy. Each example described in detail in the following sections represents a system at a different stage of technology. Current technology includes the in-place operational systems; research includes systems under development and laboratory prototypes. In these examples, the designers have the goal of increasing reliability of the actuation systems. Several fighter aircraft actuation systems are discussed. These systems are for application where extreme reliability is required because these systems must operate without failures in single thread fly-by-wire flight control systems.




The principal of operation is as follows:

Hydraulic fluid flows out of the jet pipe to the first stage of the hydraulic amplifier.

The first stage consists of the transmitter orifice on the end of the jet pipe and the two receiver oirifices below the transmitter orifices.

When the jet pipe is in position shown, the two receivers orifices have equal pressure and the piston will remain balanced between the two output ports.

By deflecting the jet to the right or left, the pressure will increase on the between the right and left orifices. This results in motion of the second stage spool and therefore a hydraulic flow to the actuator.

The jet piston is controlled by the force of generated by the electric currents flowing in the coils #1 and #2 and the magnets on the left and right and the mechanical feedback spring

FIGURE 2.1-4. OPERATION OF ELECTROHYDRAULIC SERVOVALVE

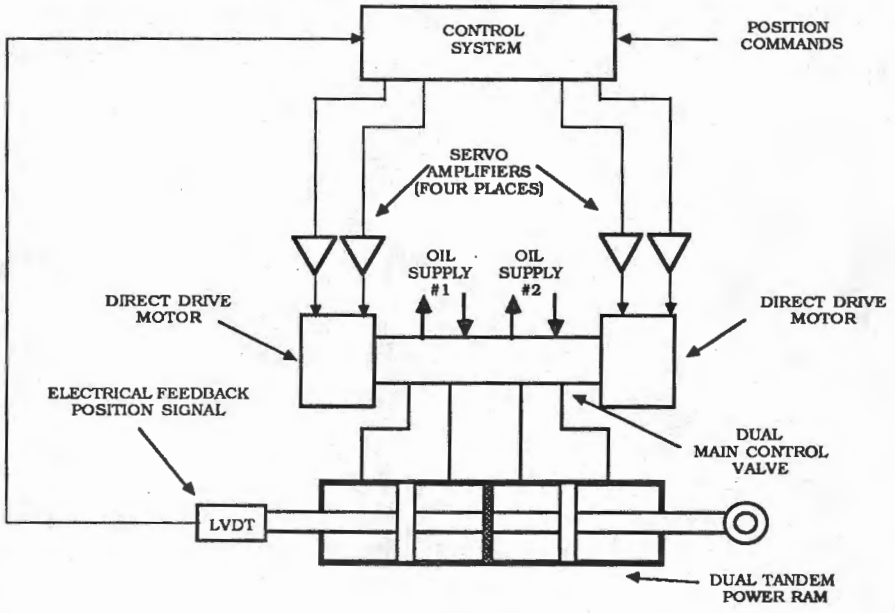



FIGURE 2.1-5. DIRECT DRIVE DUAL TANDEM ACTUATOR

The detection of component failure (and reconfiguration) must be considered in advanced fly-by-wire systems. One consequence of component failure may be the loss of the aircraft.

The most reliable hydroelectric actuators are built with the dual-tandem concept described in the previous section. This class of actuators uses more stages for amplification. Consequently, smaller first stage servovalves can be used, reducing the hydraulic power loss of the actuator. A major benefit of the main control valve is that it isolates the effect of load from the servovalves and produces a dynamic response having considerable stiffness. The loss of an electrohydraulic servovalve results in degraded performance (reduced response bandwidth) rather than total failure.

Current research is examining the removal of the main control valve in the dual tandem design. It is hoped that this modification will lead to a more reliable actuator having fewer parts. Although removal of the main control valve simplifies the design, more powerful two-stage servovalves are required. This design results in higher power loss and higher failure transients. In addition, less power is available for shearing metal chips that may jam the main ram piston.

A breakdown of failures for actuation systems was given in Kenmir 1972. This paper is relatively old, but indicates why the research trends are toward direct drive actuators. The following data indicates the percentage of operational failures due to actuator component parts:

- Tandem ram, control valves, bypass valves, and stabilization system 32%
- Servovalves and solenoid valves 45%
- Secondary Actuators 3%
- Transducers and connectors 20%

Increasing the system reliability by means of a dual channel will double the number of parts and thus double the total number of failures in a given time interval. The bulk of this increase is in servovalves and solenoid valves. Eliminating these parts would allow a significant increase in system reliability while having about the same number of failures requiring maintenance as for a single channel.

One approach to simplifying the design uses electric motors to control the main servovalve. Powerful dc motors, such as those presently used on direct drive experimental actuators, are needed to operate control valves with normal hydraulic pressures. The power loss of a standard two-stage design would be too large for a conventional servovalve to move the main control valve. Electric motors using rare-earth materials have the drive capability needed in a small size, weight, and power. At higher pressures, i.e., much greater than the normal 3000 pounds per square inch (psi), leakage around the piston becomes a significant problem. The dynamic performance is slower for this class of actuator compared to the direct drive or the conventional two-stage actuators.

The material in sections 2.1.2 and 2.1.3 is largely drawn from Baker and Bonnice 1988, and Jenney et al. 1989. These sections describe details of several selected actuators that include the following:

- F-16 Integrated Actuator
- F-18 Stabilator Actuator
- V-22 Swashplate Actuator
- Dynamic Controls F-16 Direct Drive Actuator
- Bell-4Valve actuator
- Boeing/Moog Digital Integrated Servo Actuator Controller (DISAC)
- Dynamic Controls All Digital Actuator
- Dynamic Controls F-15 Flutter Suppression Actuator

The first three, F-16, F-18, and V-22, are in use on recently developed military aircraft. They represent the best actuators currently in use. The rest of the listed actuators are experimental designs and indicate a variety of approaches for improving the performance or reliability of current designs. The Boeing Moog DISAC, Bell-4Valve, and F-16 direct drive actuators have eliminated the

main control valve. These designs are intended to improve reliability and performance. The Boeing Moog actuator is an experimental prototype developed to test microprocessor-based control, fault diagnosis, and failure management. The Bell-4Valve was developed for a flight test program. The F-16 direct drive actuator is a proof-of-concept prototype. The last two are unique conceptual prototypes built to demonstrate technology and evaluate new approaches for extended range of performance; as prototypes they have little built-in fault detection and isolation.

Table 2.1-1 shows a summary of the Fault Detection and Failure Management (FDFM) systems for the first six of these actuators (Baker and Bonnice 1988). For these systems the FDFM system is able to detect and reconfigure the actuators for servovalve, electrical, and sensor failures. No reconfiguration capability exists to address failures of the main control valve and power ram. The most likely common mode failure for the main control valve and power ram is jamming. The upper stages that drive these components are designed with sufficient shearing force to shear any metal chips that may jam the operation. An automatic reconfiguration capability exists for direct drive electrical motors. Since the most common failure modes are open or short circuits in the motor coils, the actuators are designed with several coils on a single solenoid and will continue operation with several coil failures. The shearing force and the dynamic response may be degraded due to a failure of one of the two motor coils.

Also not addressed in the systems examined are failures of the FDFM system components, including the fault detection and reconfiguration logic. This is especially true for logic circuits implemented hydraulically, or in analog circuitry where there is no redundancy. If the hydraulic or analog circuitry fails, the result could indicate a failure in error, or detect no failure at all. The latter case allows the actuator to continue operation but creates the situation where a subsequent failure may not be acted upon and a flight critical situation may arise without warning. In the case of false detection (isolation of a good component), the operation of the actuator may be shut off or degraded because of the failure in detection logic. Unless unnecessary loss of the actuator is critical to safe flight operations, false alarms are preferable upon failure detection. Digital implementations of the logic are more tolerant of faults since the logic is distributed to more than one processor.

In many cases, failures of fault reconfiguration components, such as bypass and solenoid valves and switches, are impossible to detect. This is because they are on-off devices and are not completely testable in flight. For example, if a valve or switch that is normally off fails in that position, the failure will not be detectable until the device is turned on. Failures that cannot be detected because they do not affect the present system operation are termed latent failures. Only when a component is commanded to turn on or off, and fails to act, will the failure be detectable. Since only one of the actuators investigated provided backup or redundant components for these devices (a backup coil for the DISAC actuator bypass valves), they are apparently reliable enough that fault detection is not required. Preflight testing of these devices would reduce the likelihood of flying with a failed reconfiguration device of this type.

TABLE 2.1-1. FAILURE DETECTION AND MANAGEMENT ON SIX SELECTED ACTUATORS

| Component Failures | Actuators | | | | | |
|---|---|---|---|---|---|---|
| | F-16 | F-18 stabilator | V-22 swashplate | Bell-4Valve | DISAC | Dynamic Controls direct drive |
| Servo amplifier and servovalve or direct drive motor coil | Direct comparison of servo amplifier current implemented in analog logic. Switches to standby amplifier and coil. | Servo amplifier current is compared to the output of a digital model. Eliminates failed electrical channel. | Servo amplifier current is compared to the output of a digital model. Hydraulic flow to the corresponding servovalve is shutoff. | Servo amplifier current compared to the output of a simple analog model. The corresponding EHSV is disengaged. | Not described in reference. | Cross-strapping design eliminates the need for explicit FDFM. |
| Servovalve or direct drive motor | Hydraulic comparison of output pressures (i.e., different pressure) from the first-stage servovalves. Switches to active standby servovalve if one of the primary servovalves has failed. | Hydraulic pressure output of paired first stage servovalves is compared. Hydraulic supply to the pair of servovalves with a discrepancy is shutdown. | Measured position of the second stage spool is compared to a digital model. Hydraulic pressure to the failed servovalve is shutoff. | Direct comparison of measured spool position implemented in analog logic. Control system accommodates small failures; hydraulic flow shutoff when failure detected. | Measured position of the second stage spool is compared to slow and fast models of the EHSV. Bypass valve actuated. | None |
| LVDT | None? | Position of the servo ram is compared to a model. Electric channel disabled. | Self-test. Hydraulic flow to the corresponding servovalve is shutoff. | Detected by servovalve detection test. | Self-test. Use redundant LVDT by switching microprocessor control? | Comparison of the two error signals to each motor. Command to the corresponding motor removed. |
| Other | Other failures detected by comparing the actual system output to the measured output. Solenoids are activated to mechanically center the power ram. | Loss of hydraulic power activates a bypass/damping valve. | | | Position sensor on the bypass valves allows preflight testing to detected bypass valve failure. | |

Some failures are handled using the capability within some actuators and do not require an explicit FDFM system. For example, the Bell-4Valve and the F-18 actuators can accommodate hardover servovalve failures with closed loop control. For these actuators, the control system will cause the other three servovalves to move in opposition to the failed servovalve. The result is a force fight while maintaining an acceptable performance. Similarly, the cross strapped amplifier design on the Dynamic Controls direct drive actuator for the F-16 simply offsets the effect of one amplifier failure with an opposite current in the other amplifier. Force fighting is characteristic of many such passive failure reconfiguration schemes. The advantage of this approach is that no logic or reconfiguration devices are required. The disadvantage is that excess capacity, greater than for normal operation, is required demanding more power than necessary. In addition, force fighting will degrade performance, while increasing mechanical wear and fatigue, thus increasing the likelihood of subsequent failures. Unless the fault detection thresholds account for the effects of force fighting, false alarms will occur.

Fault detection and isolation logic is implemented in three basic ways: hydromechanically (F-16); in analog circuitry (F-16, Bell-4Valve, and Dynamic Controls direct drive); and in digital software (F-18, V-2, and DISAC). For digital implementations, the logic can be in the central Flight Control Computer (FCC) or a local microprocessor. The trend in technology is from hydromechanical to digital logic. The disadvantages of hydromechanical logic are additional cost, power, size, weight, and hydraulic complexity. More complex single thread systems are generally less reliable or at least require increased maintenance. In addition, hydromechanical logic can only be used for direct redundancy failure detection and must be simple in order to be more reliable than the devices monitored. Furthermore, this logic is inflexible because any modifications require a major effort.

Analog logic overcomes most of these objections. However, analog logic is primarily limited to direct redundancy comparisons and self-test; only a limited analytical redundancy capability is possible. Digital implementation of FDFM offers the best potential. Digital systems can use all of the detection approaches including self-test, direct redundancy, and analytical redundancy. In addition, digital implementations can fine tune the logic during the operations and maintenance phase, as may be needed. Benefits of a local microprocessor, rather than placing all FDFM in a central FCC, include reducing the overhead of the FCC, less cabling to the FCC, distributed processing, and better digital control response time. However, the local environment on the microprocessor may be a serious limitation for achieving high reliability.

While digital implementations offer the best performance, the F-18 and V-22 actuator systems had constraints imposed on them by the design of the fault tolerant computer system. Interfacing with the quadruplex FCC on the F-18 required separate electrical sensors and actuators for each of the four channels. Each of the four redundant components or sensors interfaced with one FCC. No crosstalking between systems was allowed. This approach eliminates using a direct comparison of sensor outputs for actuator fault detection. For this system, the response to channel faults is to eliminate the entire electrical channel if a single sensor or component fault is detected. The V-22 system is similar to the F-18 except a triplex FCC is used. The DISAC actuator also has

limited the communication between channels. These architectures with a "brick wall" between channels benefit from a simpler FDFM system at the expense of additional components and increased failure rate. Reliable communication between channels may be possible with increased complexity and increased risk in achieving reliability goals.

The performance of FDFM systems is determined by the performance of each individual detection test. Self-test, used to detect failures on the DISAC actuator and LVDT failures, is only able to detect certain specific failures. Direct and analytical redundancy approaches should result in better detection performance due to the ease and accuracy with which component responses can be modeled. In practice (Jenney 1977), the thresholds set for fly-by-wire systems are often 30 to 50 percent of the maximum level for hardover failure. As a result only failures near hardover conditions are detected. The remaining failures are compensated for by the control system. Setting lower levels results in an unacceptably high false alarm rate. False alarms are apparently more important to the designer than concern over missed failures (Biafore and Grieszmer 1983, and Weinstein 1987).

Following are three areas of potential improvements pointed out in Baker and Bonnice 1988:

- Reduced false alarm rate.

  The detection of failures in flight often cannot be duplicated on the ground. The two most common results during maintenance are "error condition could not be duplicated under test" and "component tests ok".

- More efficient FDFM design.

  Several practices tend to increase the need for maintenance and decrease reliability. The first practice is simply adding a single sensor to a component for fault diagnosis. Then a faulty component is indistinguishable from a sensor failure. The result is that good actuator components may be removed as failed. However, if analytical redundancy or some other method was used to detect sensor failure, the reliability could be increased.

  A second practice is including excess capacity to overcome failures by force fighting. While this approach may be the only one possible for a class of failures, several actuators use this approach simply to reduce the complexity of the FDFM system. In these cases, the result is increased size, weight, and power requirements. In addition, the increased wear from force fighting will ultimately increase the rate of failures.

  Actuation designs could also be more efficient if fewer sensors were used. In existing systems, the loss of a component results in the shut down of an entire channel. Reducing the number of sensors through analytical redundancy, for example, could reduce the number of channels from four to three. Of course the additional burden in computer resources and communications must be weighed against the overall benefit.

- Control system reconfiguration.

None of the actuators reviewed alter the control system in any way following a failure. In addition to altering the control system to regain performance, reconfiguration could offer additional fault recovery options. Inner loop sensor failures might be compensated for by altering the feedback loop. Inner loop feedback improves the dynamic response of an actuator, but is often not absolutely necessary for continued emergency flight and landing. Effectiveness of control reconfiguration for failure management needs to be evaluated for each actuation system.

### 2.1.2 Detailed Examples of Current Technology

#### 2.1.2.1 Standard F-16 Integrated Servo Actuator

Figure 2.1-6 is a schematic diagram for a standard F-16 actuator (Baker 1988). This actuator is used for controlling the horizontal tail, flaperons, and rudder of the F-16 aircraft. Because of the critical function of this actuator, it has considerable redundancy and fault detection and reconfiguration provisions. It is designed to accommodate dual electrical and hydromechanical failures. The actuator operates with a 3,000 psi supply pressure. The actuator has a force output of 36,000 pounds, a maximum slew rate of 5.5 inches per second, and a nominal frequency response having a 3 Hertz (Hz) bandwidth. This actuator is designed to accept two failures in the electrical system and one hydromechanical failure and continue to operate. Upon detection of a third failure the actuator will move to a centered position.

In the standard configuration (figure 2.1-6), the secondary actuator controls the position of the main control valve. The secondary actuator is controlled by three jet pipe servovalves. Failure detection and reconfiguration is accomplished hydromechanically using comparator spools to monitor the servovalve driving pressures and to drive current commands. A first failure of SV1 or SV2 will cause transfer of control to the standby servovalve SV3. A first failure of SV3 will lock the valve onto SV1 and SV2 control. Mechanical feedback loops are used for control of the servovalves, main control valve, and actuator ram. Solenoid valves are used to remove or apply pressure to the servovalves and the fail-center mode upon request from the actuator control electronics.

Note that the torque motors are not included inside the mechanical feedback loops. This results in the actuator position being commanded by current supplied to the servovalve coils.

The dual fail operation for electronic control is provided by each servovalve having two electrically separate windings. Either winding is capable of full servovalve control. These windings are driven by the FCC in active standby mode so that a first failure of a servo amplifier will be corrected within the flight control computer without activating any voting logic. The actuator control electronics incorporates analytical redundancy. This redundancy uses a mathematical model for the actuator position, including the dynamic rate plus position feedback signals. Comparison of the mathematically calculated position to the position measured using LVDT that is incorporated into the actuator

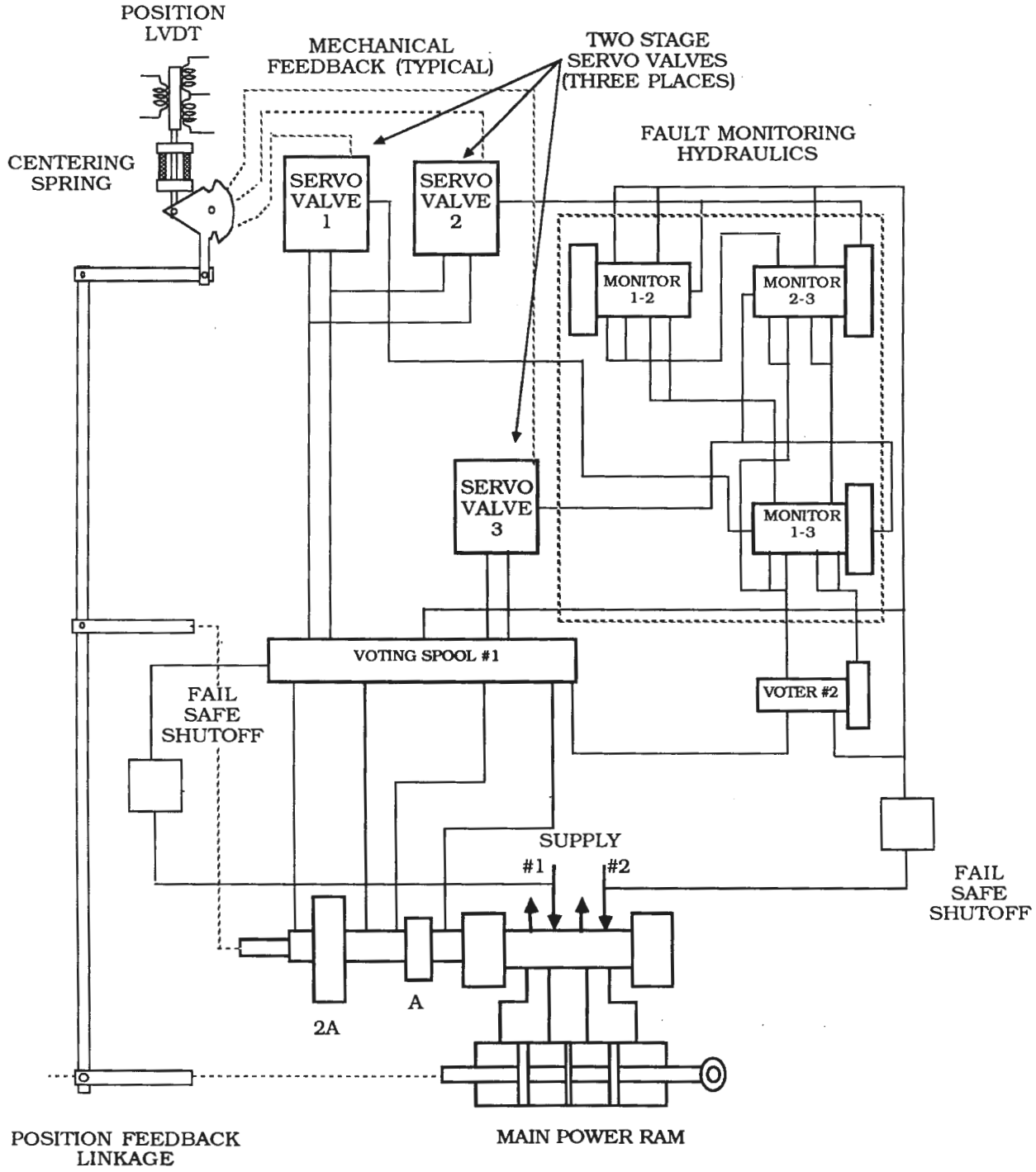



FIGURE 2.1-6. F-16 INTEGRATED SERVO ACTUATOR

determines an error signal. This error signal is used to determine if the actuator has failed. If the actuator has failed, move the piston to the centering mode of operation. In the centering mode, actuator position is determined by a centering spring attached to the position feedback linkage from the main actuator and the main control valve.

2.1.2.2 F-18 Actuator

The F-18 stabilator actuator (Harschburger 1983), illustrated in figure 2.1-7, has considerable redundancy and failure management. There are four single stage electrohydraulic servovalves which are paired to allow failure detection by comparison of their output pressures. This direct detection design uses four differential pressure transducers on each pair of servovalves. Failures of the servovalves, detected by these sensors, are contained by means of a solenoid valve which shuts off the hydraulic supply to that pair of servovalves.

Each of the servovalves is commanded electrically by four coils with the coil on one servovalve connected in series with a coil on each of the other servovalves. Each coil string is driven by a separate amplifier. Outputs from the four digital flight control computers are interfaced to a single amplifier supplying a string of coils. With this flux summing arrangement, the currents from the four separate amplifiers are effectively added.

Amplifier and coil circuit failures are detected by comparing the current from each amplifier to that calculated by a digital circuit model in the FCC. When a failure is detected, the amplifier for that coil channel will be disabled. This actuator design is able to meet performance specifications with two channel failures so that no failure management is needed. Detection of more than two channel failures results in the hydraulic flow to both pairs of servovalves being shut off, thus reverting to mechanical control.

Each pair of servovalves drives one piston of the dual tandem actuator. The two pistons are mechanically linked to a main control valve. This is motivated by the requirement for mechanical reversion in the event of a non-recoverable piston jam.

There are four LVDTs for sensing position for feedback on both the servo ram and the power ram. Each LVDT is connected to one of the four FCCs. Each FCC drives one of the servo amplifiers. Failure of an LVDT sensor or the command from the flight control computer is detected either by the servo amplifier current failure detection or by comparing the position of the servo ram with an analytical model. Sensor fault detection by direct comparison of sensor outputs is not possible because the digital quadruplex FCC architecture does not allow any cross-channel communication.

Common mode failures, resulting in loss of hydraulic power, are negated by a bypass valve that equalizes the pressure on either side of the ram piston, allowing the control surface to float.

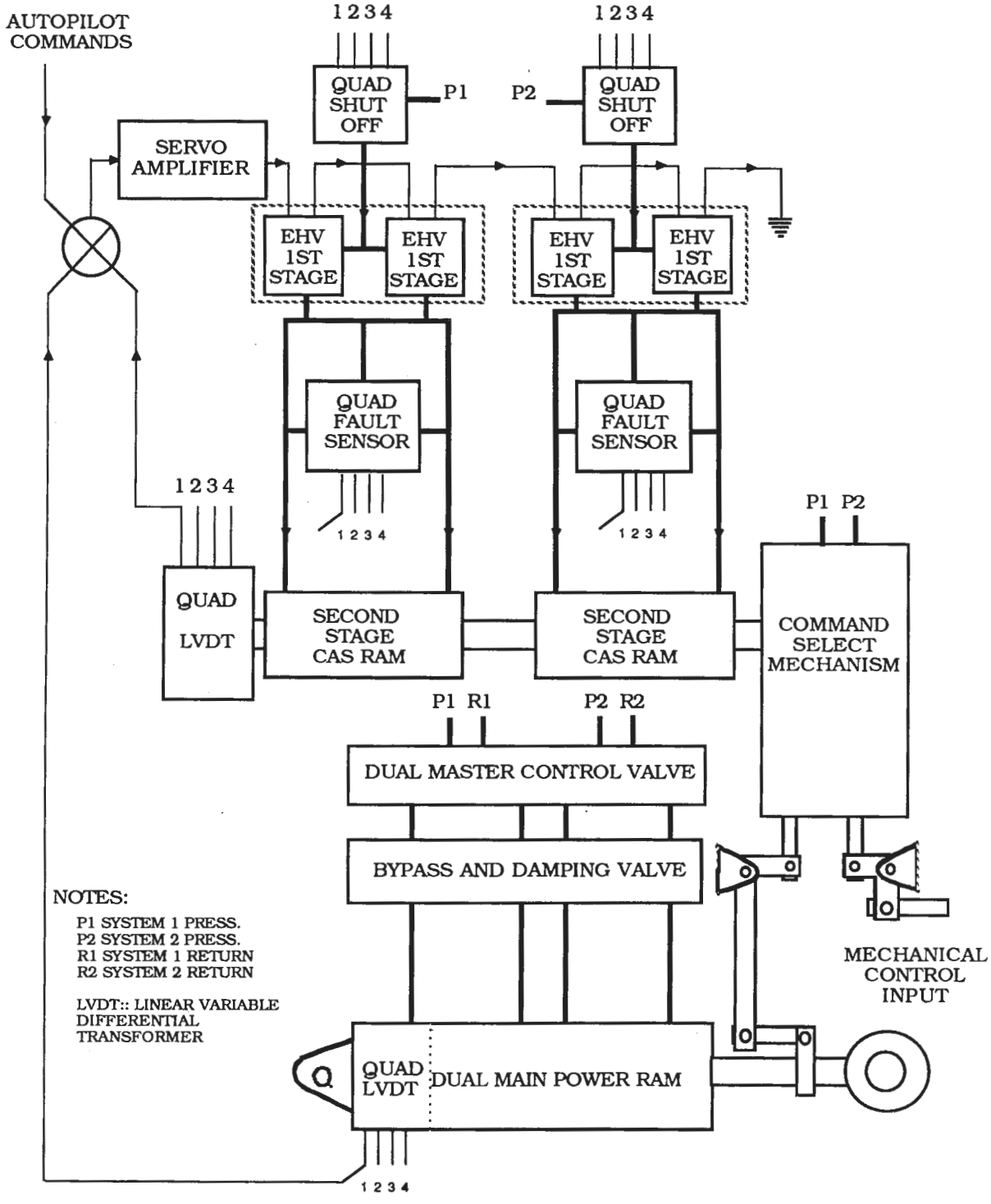



FIGURE 2.1-7. F-18 STABILATOR ACTUATOR

2.1.2.3 V-22 Actuator

The V-22 swashplate actuator (McManus 1985), illustrated in figure 2.1-8, is controlled by a triplex system. Three of these actuators are used on each main rotor. This system has considerable redundancy and failure management. The triplex uses self monitoring to detect failures within a local channel without depending on the other channels. However, the redundancy of the V-22 actuation system is not as great as for the F-16 quadruplex FCC. The triplex is configured as a duplex with failure reversion (a two-fail-operate system). There are two hydraulic supplies.

The actuator main control valve is driven with two unbalanced modulator pistons. Either of the two two-stage servovalves in system one are configured to drive one modulator piston with the third two-stage valve in system two driving the second smaller piston. Since the second piston has one-half the area of the first, this sets up a force fight. Since the drive currents are the same, the smaller piston loses. Each of the three servovalves are commanded separately by the three FCCs. In the normal no-failure condition, system one controls the main valve. Either FCC1 or FCC2 can perform the function. Any failure in FCC1 or FCC2 will be automatically detected by the comparisons between FCC1 and FCC2. Any failure in FCC3 will be unnoticed as long as either FCC1 or FCC2 is operating. Any two-failure situation will leave the system functioning with full performance.

Servovalve failures are detected using the position sensor on the second-stage spool of each servovalve. The current to each servovalve is measured and compared to the current calculated from a digital model in that FCC channel for detection of amplifier and servovalve failures. In the event of failure detection that channel will be shut down.

Each of the FCC channels contains a dual processor. The two FCC processors, A and B, in each channel monitor themselves. Either one can perform fault reversion. Some sensor inputs are compared across the channels to detect failures not covered by in-line monitoring. Single self-monitored sensors are used. These inputs are passed to the redundant processors (A and B) via a nonredundant sensor interface and single input/output (I/O) processor. Self checks by the two processors also check the I/O processor and sensors. Each processor calculates the output commands and expected response. Each processor can control engagement of the servo loops through discrete outputs. Commands from processor A are sent to the actuators by the I/O whose performance is checked by processor A and B using wraparounds. A watchdog timer is used to track each time frame; all actuators associated with an FCC channel will shut down upon failure detection.

The six LVDTs are used to monitor the position of the servovalves, the main control valve, and the power ram. Failures of LVDTs, including the triplex LVDTs on the main control valve and power ram, are detected using a self-test approach because no cross-channel communication is allowed for this FCC architecture. The response to failure of a servo amplifier, control coil, servovalve, or LVDT is the same: operate the shutoff valve or bypass valve to disable hydraulic flow to the servovalve in the failed channel.

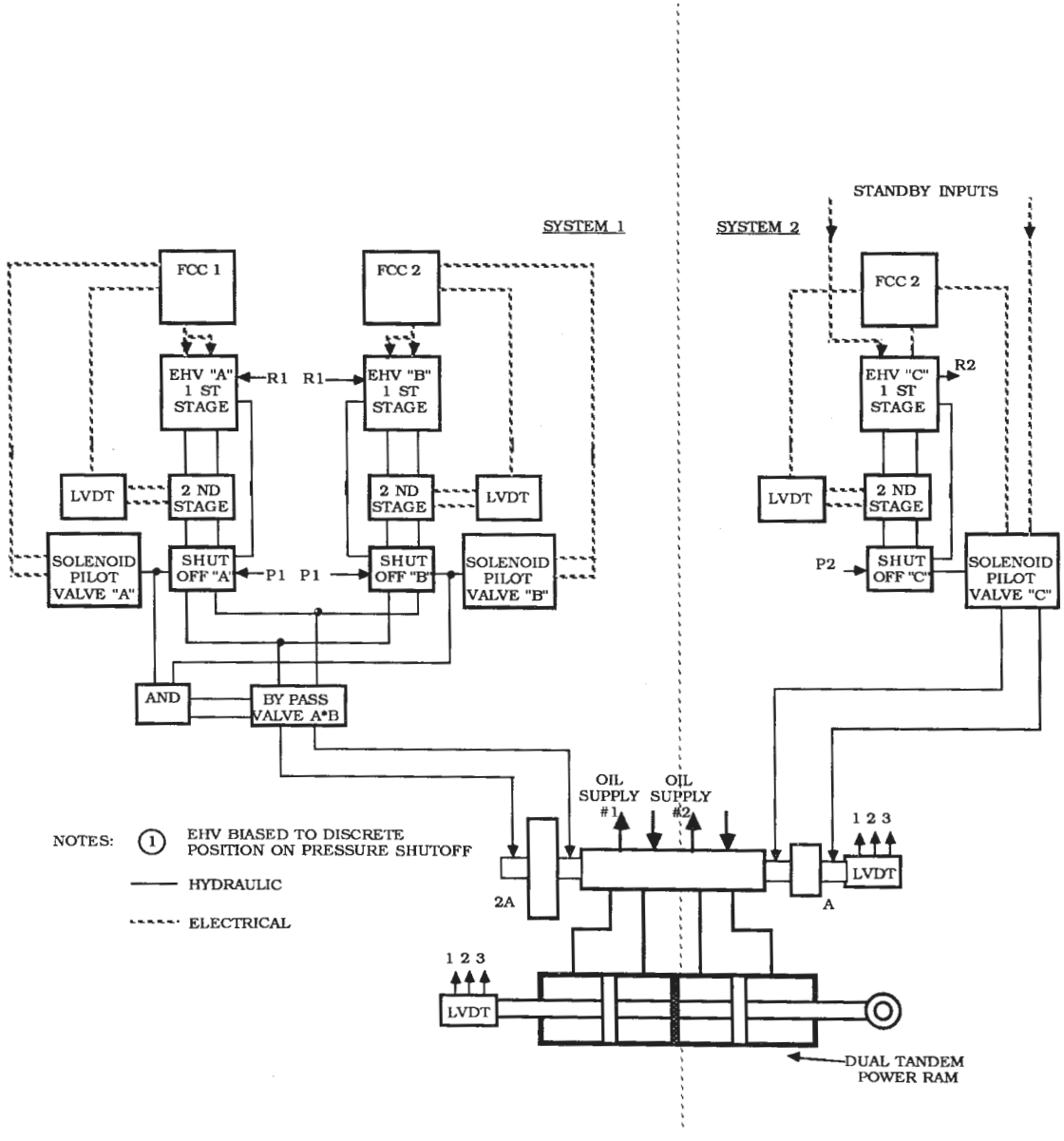



FIGURE 2.1-8. V-22 SWASHPLATE ACTUATOR

## 2.1.3 Detailed Examples of Advanced Technology and Research

### 2.1.3.1 Dynamic Controls' F-16 Direct Drive Actuator

The objective of this investigation, reported in Jenney 1989, was to implement and test a direct drive actuator for a current advanced technology fly-by-wire aircraft. The F-16 integrated actuator was chosen for the investigation because it was still in production and a sample flaperon/horizontal tail actuator was available for use.

A direct drive actuator directly modulates the flow of the main piston by hydraulic servovalves. This eliminates the requirement for a separate secondary actuator to position the main control valve.

Direct drive actuators operate the main servovalve using force motors having sufficient force. This technology was flight tested in 1981 on an F-4E at Edwards Air Force Base and has been subsequently refined in a laboratory investigation reported in Jenney 1989. The schematic diagram, shown in figure 2.1-9, illustrates a single-fail-operate direct drive actuator design. For the F-16 application a third channel of electronics was added to accommodate the dual-fail-operate requirement. Each channel was self-monitoring and independent of the others. Six feedback transducers were required to retain complete independence for redundancy purposes. The system designer placed the actuator inside a position feedback loop. This configuration is desirable because power is required only when the actuator is moving. Using the actuator in this manner also relaxes the requirement on linearity of the drive spool motion.

For this direct drive configuration, actuator performance degrades upon failure. Using electrical feedback for the direct drive actuator position also allowed inclusion of the force motors and control valves inside the position loop. This operation allowed simple reconfiguration following failure. Upon any single failure of the three self-monitoring control channels or both hydraulic supplies, transfer was made from the electrical-control mode to the self-centering mode. This transfer was accomplished with solenoid valves as for the normal F-16 actuator. With a multiple coil electrical control for each valve, the force applied is a direct function of the coil currents. When any channel fails, the position gain (and hence the flow gain) automatically changes. The actuator responds to position commands, with degraded performance, even if two of the three electrical channels fail.

Performance of this actuator could be maintained following failures by adding electronic elements. Using feedback of the spool position in an interloop would tend to keep the flow gain constant, with an increase of complexity. Gain changing after failure can be used to keep the valve performance constant following the loss of a channel. However, gain changing requires coupling between channels, which reduces control independence and could possibly introduce common mode failures. In the interest of high reliability, gain changing or interloop position feedback was not used for the prototype design.

The initial prototype design included analog failure monitoring. A second phase of the investigations (Jenney 1989) added microprocessor failure detection and correction. For either of these approaches, the failure monitoring would be checked pre-flight. These monitoring systems were designed so that failures in the monitoring system appear as single channel failures and could not cause hardover outputs.

The force motors were driven using pulse width modulation to reduce the heat sink requirements for the drive amplifiers. Because the coils are driven full-on or full-off, the power dissipated in the amplifiers is smaller than if operated in a linear mode. This minimizes the heat generation in the amplifiers and the size of the heat sinks. A potential disadvantage is the electrical noise generated by the current switching transients.

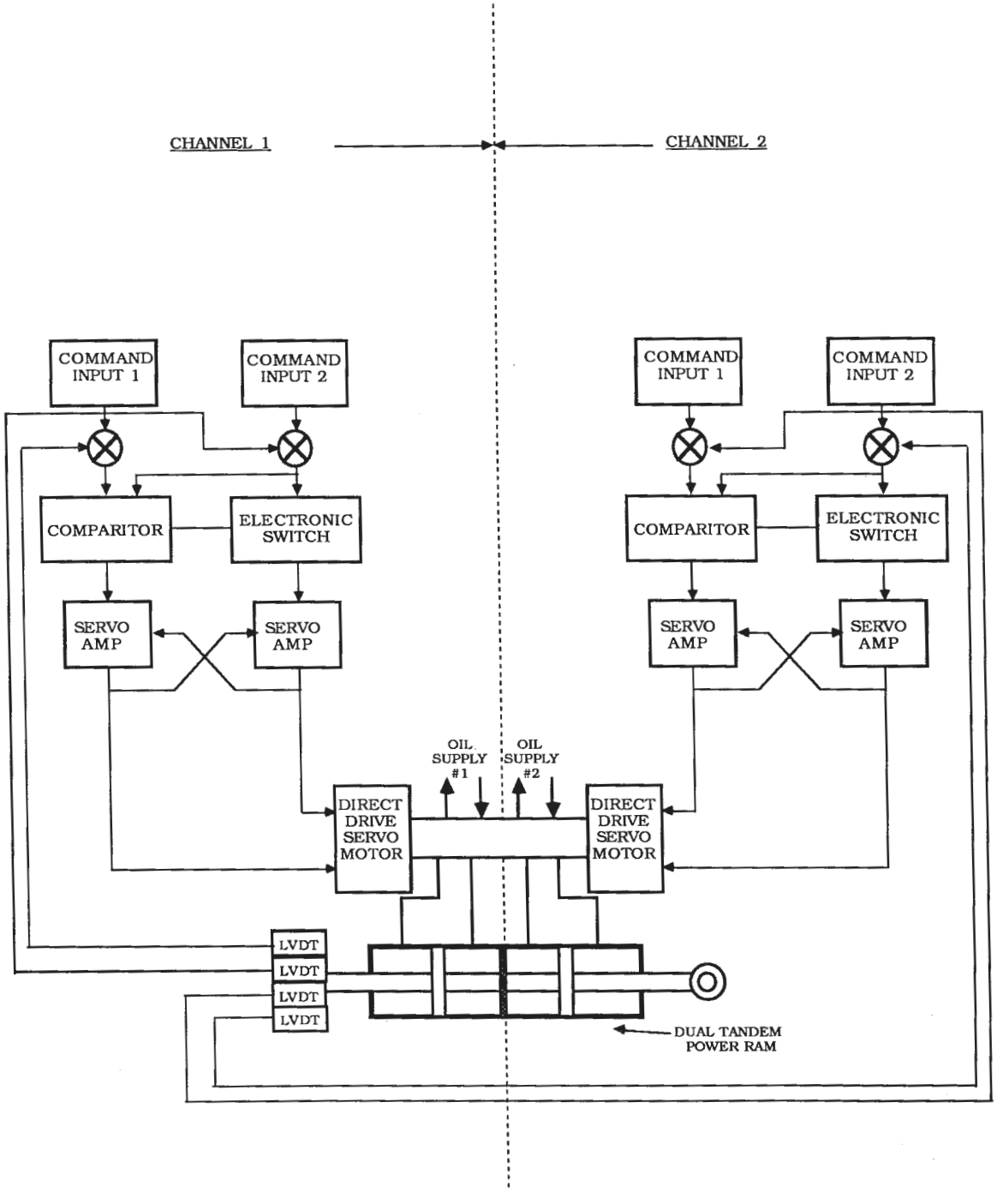



FIGURE 2.1-9. SINGLE FAIL OPERATE DIRECT DRIVE ACTUATOR

Figure 2.1-10 shows a block diagram of the two-fail-operate direct drive actuator designed for the F-16. All three channels operate normally, two for a single failure operation and one for the dual fail operate. Note that the three channels are self monitoring and no electrical mixing of the channels occurs. Since each channel uses two command inputs, the failure logic can detect input failures. These input signals are strapped together if failures are detected and corrected ahead of the control actuator system. Six feedback transducers allow each channel to be self monitoring. When a failure of a feedback transducer, summing amplifier, or command input is detected by the comparator, the inputs to the servo amplifier will be grounded. Two amplifiers are used for each channel. These two amplifiers are cross strapped together so when an amplifier fails, the other amplifier will be driven in the opposite direction, cancelling the output of the failed amplifier. If an amplifier fails resulting in a no-output condition, then the gain of the remaining amplifier will increase, tending to keep the channel force gain constant.

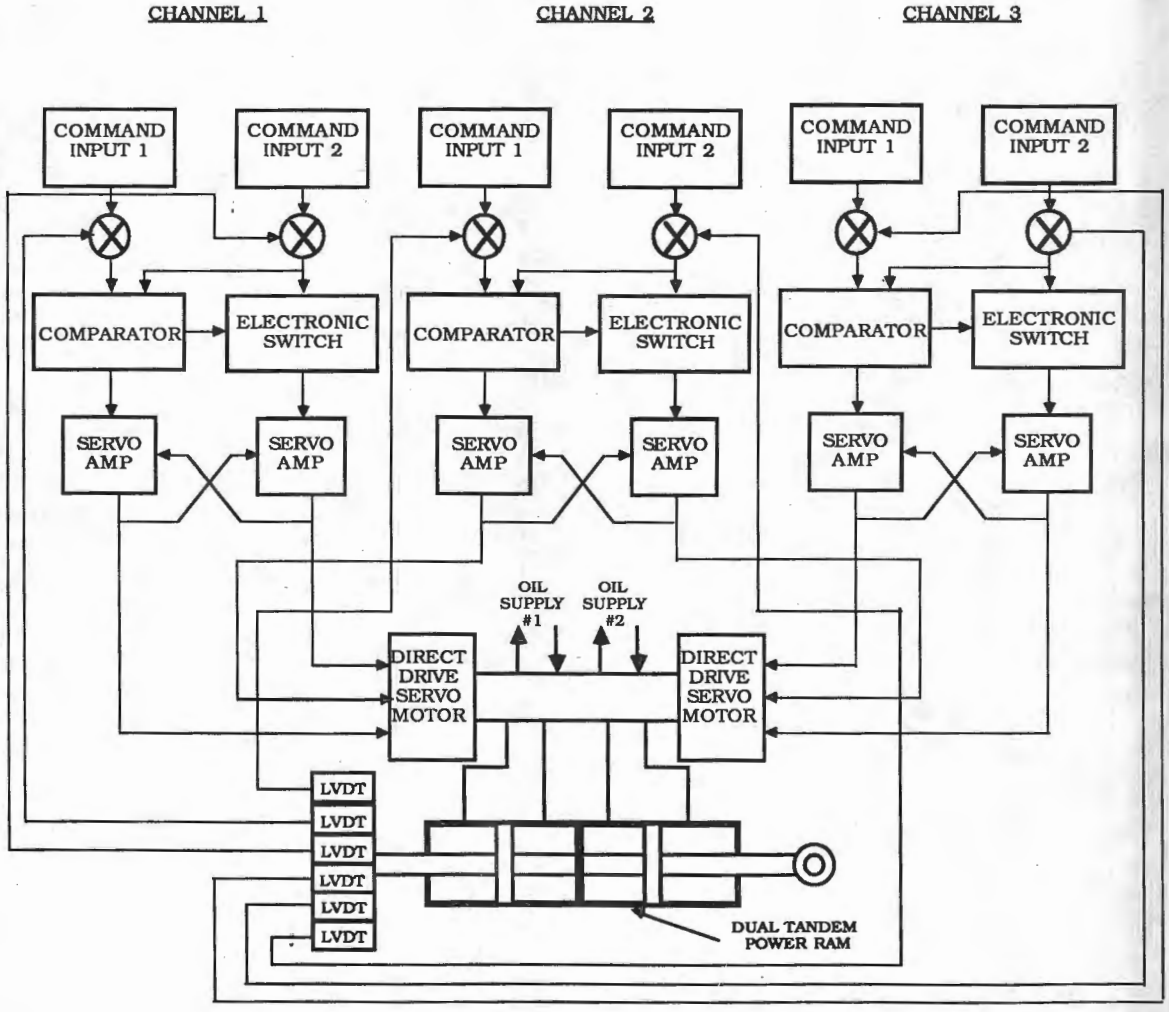



FIGURE 2.1-10. DYNAMIC CONTROLS' F-16 DIRECT DRIVE ACTUATOR

The direct drive control valve concept for laboratory investigations is based on a moving coil design. The coil assembly is suspended by a folded linkage incorporating both a centering spring and flexures for the pivot points. The coil assembly consists of three individual coils, electrically isolated from one another, with four layers of windings in each coil. The coils move within a magnetic circuit energized with a rare earth permanent magnetic material. Stainless steel 416 was used for the pole piece and outside cup. Tascore 21 was used for the permanent magnet material. The force output from the coil was 55 pounds with all coils energized at 1.2 amps. The resistance of each coil was nominally 4 ohms. Figure 2.1-11 shows the structure, coil, and housing. This is a wet coil design flooded with return oil. Use of a wet coil eliminates the need for a dynamic seal and provides cooling for the coil and damping for the coil motion. Since the oil is fed through a filter disk and only circulated locally, the problem of the magnetic structure acting like a magnetic oil filter does not occur.

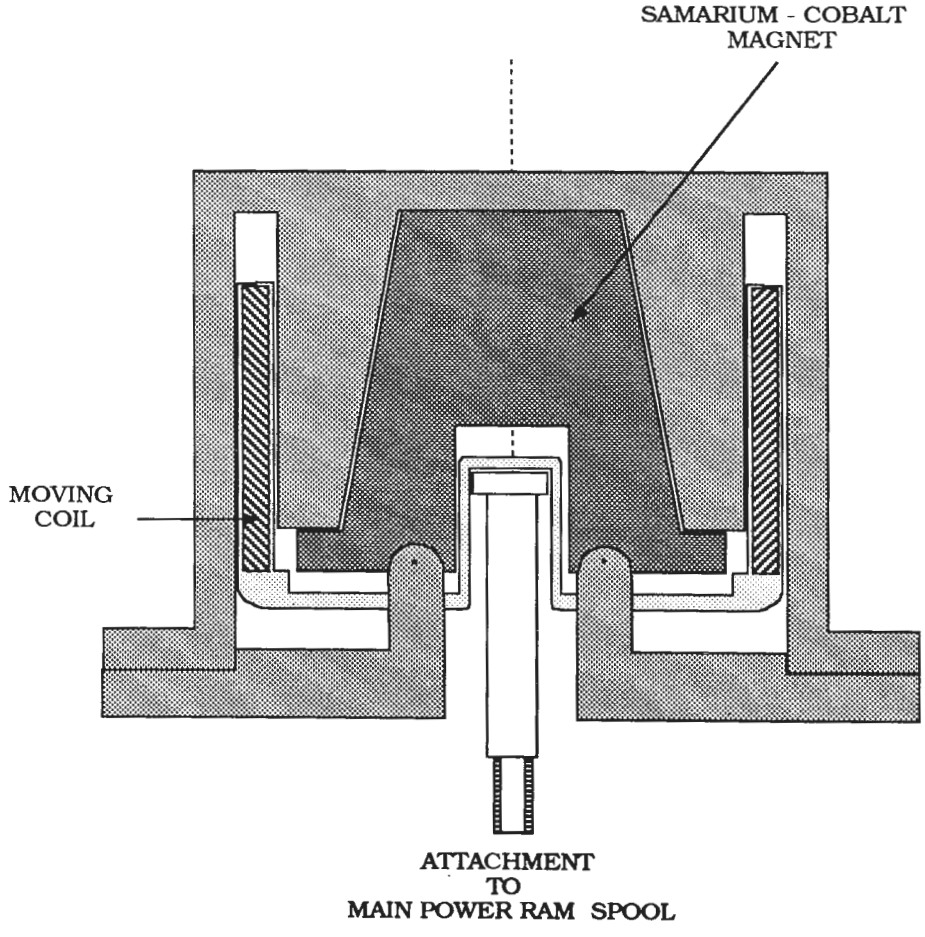



FIGURE 2.1-11. MOVING COIL TORQUE MOTOR

Two force motors were used for the F-16 actuator, one on each end of the main control valve. These motors were connected together in the valve package with a large flow passage so that the pressure could balance between the magnetic structures. The return pressure line of one hydraulic system supplied oil for both force motor coils.

The main control valve that both motors positioned was a three part design. The valve moved in a sleeve which was normally held stationary by a locking mechanism. Electrical feedback normally controlled the position. When mechanical feedback was used in the fail-operate mode, the sleeve was unlocked and driven by the mechanical feedback linkage. In this fail safe mode the driving electronics for the center spool are turned off and the control spool maintained in a center position by springs that center the force motor valves.

2.1.3.2 Bell-4Valve

The Bell-4Valve actuator (Abrams and Donley 1984, and Murphy and Haskins 1982 and 1984), illustrated in figure 2.1-12, uses four two-stage flapper valve servovalves to meter the hydraulic flow directly to the dual tandem power ram.

Any failures of the servovalve coil or drive wire are detected by comparing the drive current to that calculated from a simple analog model. If a failure is detected, then that particular valve is disengaged by operating a solenoid valve. Active servovalve failures are detected by directly comparing positions of the second stage spools of the four servovalves. For hardover failures, the control system causes a flow bypass around the piston of the failed servovalve.

Because the Bell-4Valve is able to accept a hardover servovalve failure, the detection thresholds are set so that only large failures are detected. The advantage of this approach is that false alarms are reduced. If both channels driving a piston fail, a bypass valve is activated for that piston. The FDFM system is implemented in analog logic.

2.1.3.3 Boeing/Moog DISAC

The DISAC prototype actuator (Waffner and Chenoweth 1984, and Chenoweth and Slaugh 1985), illustrated in figure 2.1-13, was developed to be controlled and managed by two microprocessors. It differs from the Bell-4Valve actuator in that only two servovalves are used.

In the primary mode, each microprocessor controls the servovalve and bypass valve for one channel. However, if one microprocessor fails a self-test, that processor drops control and the other processor takes control of both channels. Logic exists to keep both microprocessors from attempting to control the same component at the same time. Both processors access data from both channels by means of duplicate sensors. One sensor interfaces to each processor. In addition, both microprocessors are able to operate each servovalve and bypass valve using separate coils in each component.

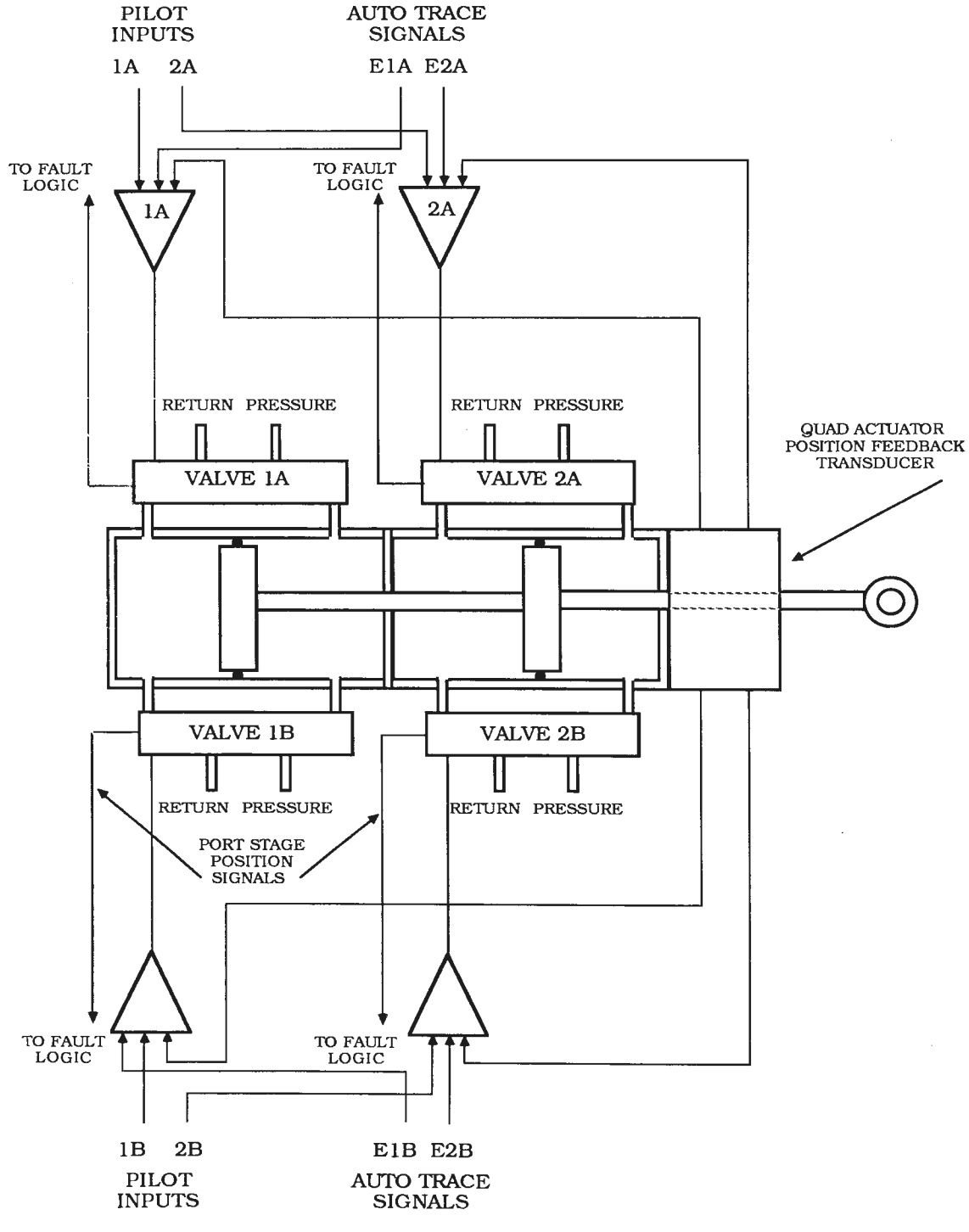



FIGURE 2.1-12. BELL-4VALVE ACTUATOR

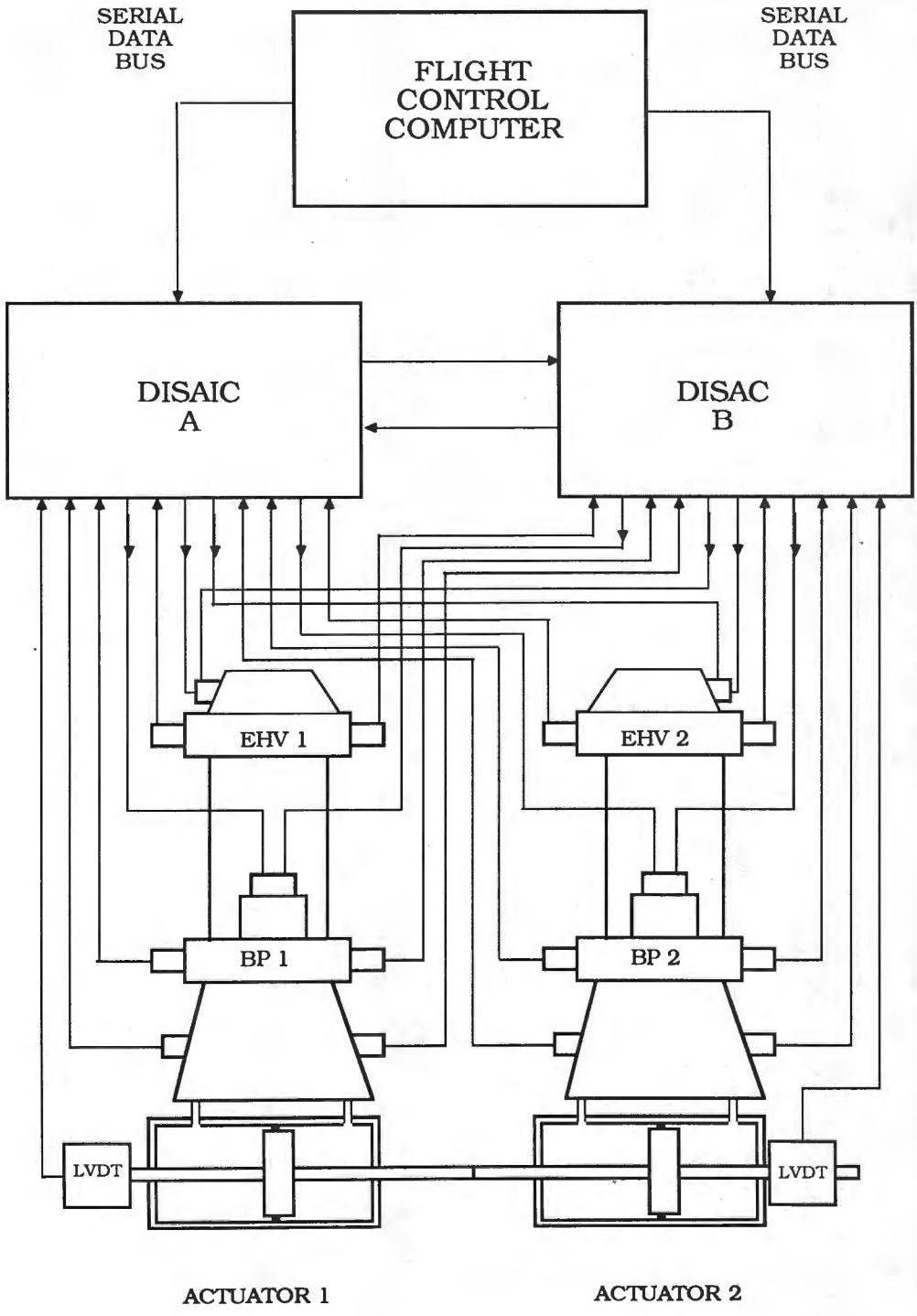



FIGURE 2.1-13. BOEING/MOOG DISAC ACTUATOR

Failure detection on this actuator consists of comparing the measured position of the second stage spools to fast and slow models of the servovalve to detect failures. LVDT failures are detected using self-test. The failure management response could be either to turn off the channel with the failed component or allow the other microprocessor to take over operation of the channel.

One unique feature of this actuator is the use of position switches on the bypass valve to verify its operation during preflight tests. Bypass valves failing open is a latent fault that cannot be observed during normal flight operations. Preflight testing at least verifies bypass valve operation prior to takeoff.

2.1.3.4 Dynamic Controls All Digital Actuator

Recent research and development, reported in Jenney 1989, illustrates innovation in implementation of a direct drive valve. This all-digital hydraulic servovalve has digital signals applied directly to poppet valves controlling hydraulic flow to the main piston. In this implementation the poppet valves control flow in binary increments. Each valve is driven on or off by the value of a single bit in a binary word. Thus, the overall flow is proportional to the binary value of a digital command. The flow is controlled by means of a piezoelectric (PZ) driven poppet valve.

The poppet valves used in this configuration had a step response 30 times faster than the electromechanical valves normally used for control of the main piston. The concept has potential for very fast response of the main actuator. The position control loop is closed with a microprocessor and position transducer. Results from a laboratory demonstration achieved 0.5 millisecond operating speed, although the PZ poppet valves failed to meet their speed objective by a factor of two too slow.

A schematic of the operating elements of the digital control valve and actuator are shown in figure 2.1-14. This implementation used four orifices. Four poppet valves are used for directional control. Valves V1 through V4 are connected from a cylinder poppet of the valve to hydraulic pressure return (operating these valves controls the direction of the piston). Opening valves V2 and V4 causes the valve to move to the right (as shown in figure 2.1-14).

A gallery is connected to the combined output flow from the flow modulation poppet valves and orifices. As shown in figure 2.1-14, the four orifices, labeled O1 to O4, are used for flow modulation. Orifice O1 is connected to the hydraulic supply while O2, O3, and O4 are connected to valves V5, V6, and V7. These three valves, in combination with the non-switched orifice O1, provide metered flow to the directional valves.

The rate that the actuator moves is determined by the size of the orifices and the number that are opened by the poppet valves. The sizes of the orifices are scaled so that the flow increases in powers of two (i.e., O3 causes twice the flow of O2, and O2 causes twice the flow of O1). Thus binary coding of the digital word connected to the three orifices allows eight steps from off to full

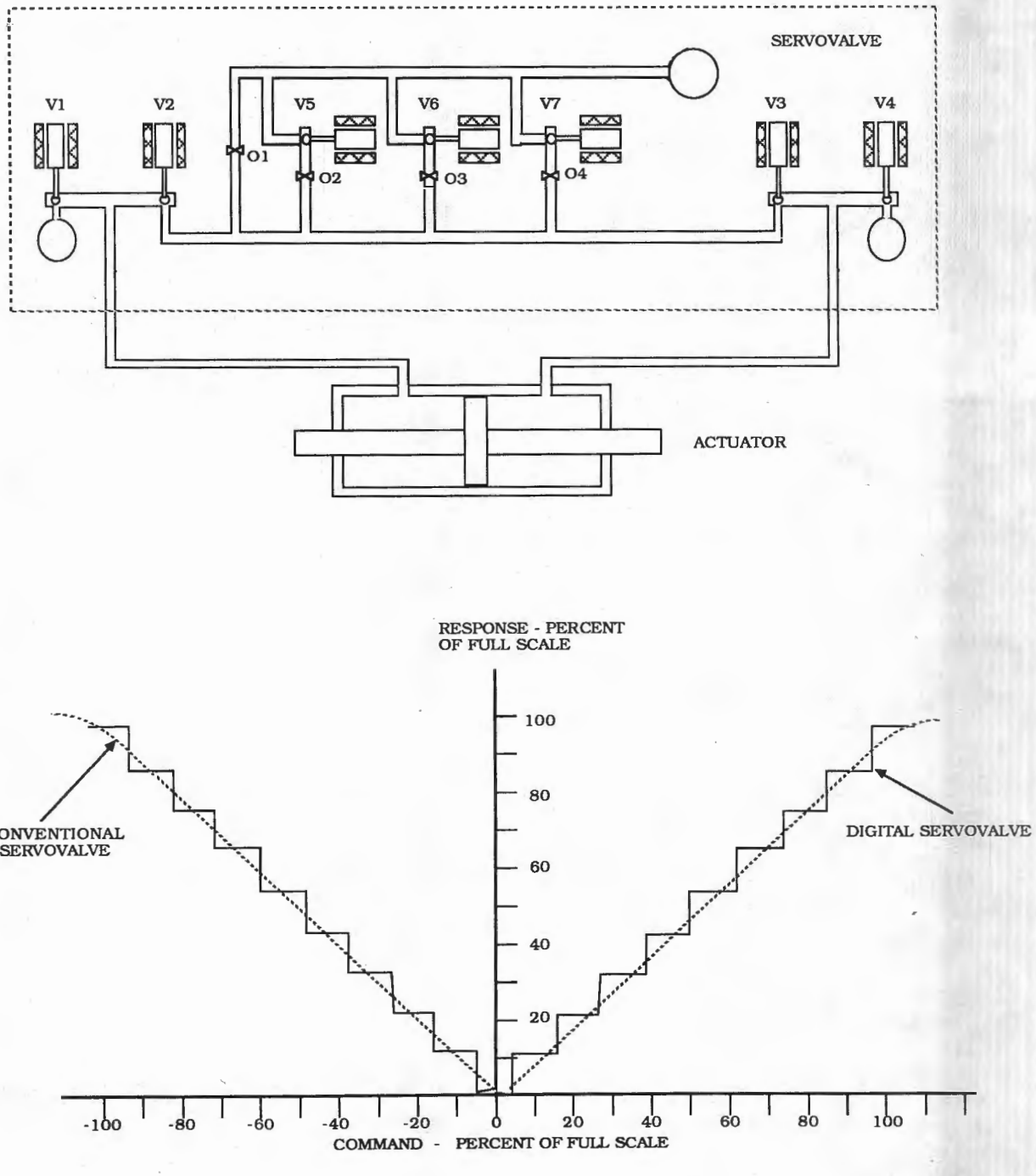



FIGURE 2.1-14. DYNAMIC CONTROLS' ALL DIGITAL SERVOVALVE ACTUATOR

on. With eight steps, flow linearity is maintained within ±6.25 percent of a linear output. A production article could be built having additional steps and bits in the digital control word. Eight bits would provide 256 flow step combinations (better than 0.5 percent resolution); this is in excess of a more normal requirement of two to three percent linearity for actuator position control. Addition of one more poppet valve would provide a flow linearity of ±3.1 percent, comparable to most analog servovalves.

From a reliability and redundancy viewpoint, this digital valve appears very promising. The valve motion is determined by fixed elements, the orifices. Control is afforded by digital devices that only operate in an off or on state. Further, if any one of the orifices or associated valves fails to operate, then the actuator functions at lower rates of motion. Degraded performance is usually more desirable than "hardovers" or "no response". The critical control element for establishing hydraulic flow is the poppet valve. Valve response times vary widely for electrical control of mechanical motion. This prototype demonstrates that great improvements in valve speed are possible.

A conventional magnetic solenoid valve is limited in response time by electrical and mechanical constraints. A survey of electromagnetic poppet valves, conducted as part of the research reported in Jenney 1989, provides some interesting results. The on-off response times for commercial poppet valves varied from 2.5 to 14 milliseconds and cost estimates were between $1,500 and $4,000 each. All of these valves failed to meet the required response of the control actuator. One reason for this can be found in the dynamics of the mechanical valve components. A typical solenoid valve consists of a spring, armature, and coil, as shown in figure 2.1-15. The motion of these elements requires the magnetic circuit to build up forces to accelerate the armature and open the valve in a

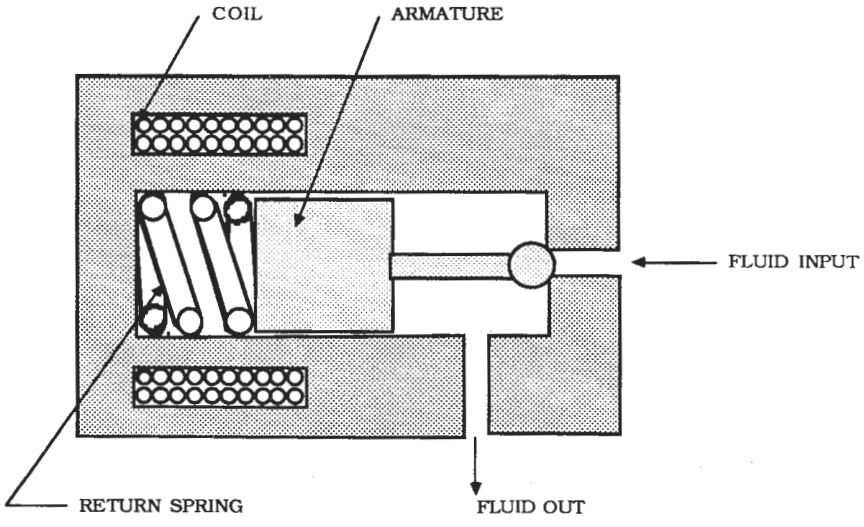



FIGURE 2.1-15. SOLENOID POPPET VALVE

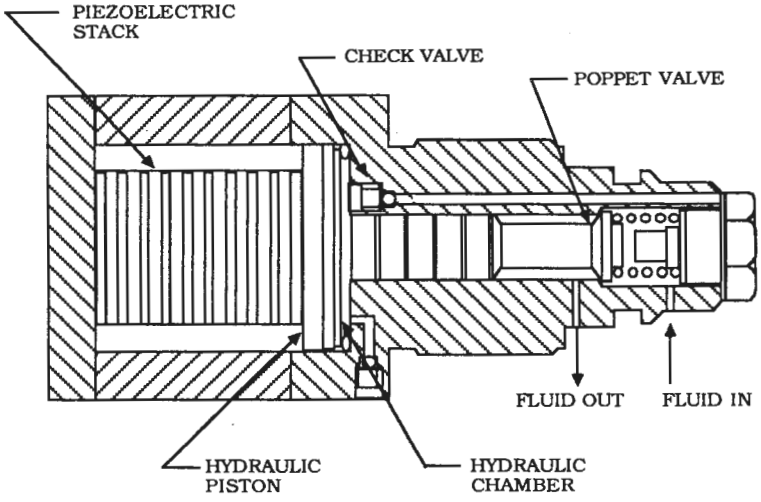
short time. The armature and coil must be sufficiently large to produce forces great enough to overcome the spring restraining force and inertia. For example, an armature weighing 0.022 pounds and travelling 0.02 inches in 0.001 seconds requires a driving force of 2.27 pounds. The spring preload to close a 0.30 inch diameter valve against a 3,000 psi pressure is 2.12 pounds. Thus the magnetic forces must exceed twice the spring force. An additional constraint is in the magnetic coil. A large number of turns on the coil will build up forces for the same current. However, the coil inductance increases with the square of the turns, requiring more time to build up the drive current. The RL time constant ultimately limits the current build up and speed of operation of the poppet valve.

A moving permanent magnet armature has also been tried for this application on other research projects and was found to have the same RL time constant limitation on rapid build up of control currents.

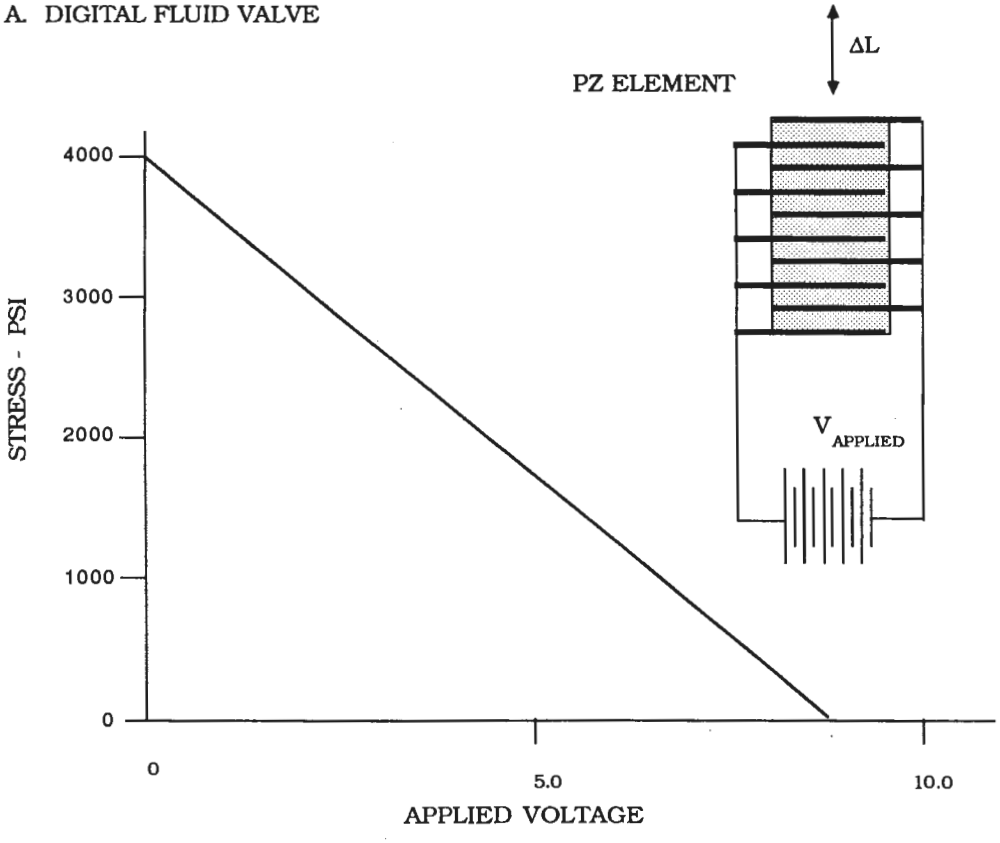
A review of PZ elements for application to poppet valves indicated feasibility of constructing a valve having a 1 millisecond operation time. A PZ material will distort when an electric field is applied to it. Conversely, an electric field will be generated when the material is distorted by a force. Figure 2.1-16 illustrates a poppet valve constructed from a multilayer PZ element.

The quantitative values for the relationship between the mechanical deformation and electric fields depends upon the material type, crystal axis orientation in the field, and physical configuration of the device. The output force capability of PZ materials is very high (on the order of 3,000 psi), although the strain is very small (on the order of 0.1 percent of the thickness). The electrical characteristic of PZ material is the same as a capacitor. Because of the high force to mass ratio for these materials, the strain can be achieved in a few microseconds. Typically, fields of 25,000 volts/inch are required to produce a strain of 0.001 inch. To reduce the driving voltage, PZ drivers are produced in thin layers, bonded together, and connected electrically in parallel. For a field of 25,000 volts/inch, a layer thickness of 0.020 inch only requires an applied voltage of 500 volts. When the material is preloaded, the strain will be determined by the modulus of elasticity and the stored energy. Since Young's modulus for a typical PZ material is about 25 percent that of steel, quite large preloads are no problem. The charge/discharge characteristic displays a large hysteresis: on the order of 20 to 30 percent for typical materials. This hysteresis limits the application of PZs to digital rather than analog control devices.

The laboratory investigation reported in Jenney 1989 achieved an operating digital controlled actuator having PZ poppet valves with 500 microsecond operation times in a hydraulic supply pressure of 3,000 psi. The microcomputer was an off-the-shelf Z80 having a real-time program processing time of 600 microseconds. This is very impressive compared to the fastest hydraulic poppet valve that has an operate time of greater than 5 milliseconds.




A. DIGITAL FLUID VALVE




B. PERFORMANCE OF PZ STACK

FIGURE 2.1-16. PIEZOELECTRIC SERVOVALVE

Since these devices operate very quickly and are easily driven from 270 volt dc with digital controllers, future aircraft should have numerous applications of PZ devices. The high speed and high force capability of the PZ actuators should be applicable to direct-drive force motors for controlling spool type servovalves.

2.1.3.5 Dynamic Controls' F-15 Flutter Suppression Actuator

There is a trend toward increasing the operating pressures of military aircraft hydraulic systems. Higher system pressure allows the use of smaller, lighter weight actuators with smaller drive areas. This conflicts with the normal design technique of using a larger drive piston area to increase actuator stiffness to avoid flutter of the control surfaces. An increased stiffness is usually accomplished by increasing the ram piston area (i.e., by using an actuator having a higher force output and flow demand than required for maneuvers). An Air Force project (Jenney 1989) developed a prototype actuator having electronic control signals for changing the impedance at the flutter frequency and thus effectively damping control surface flutter.

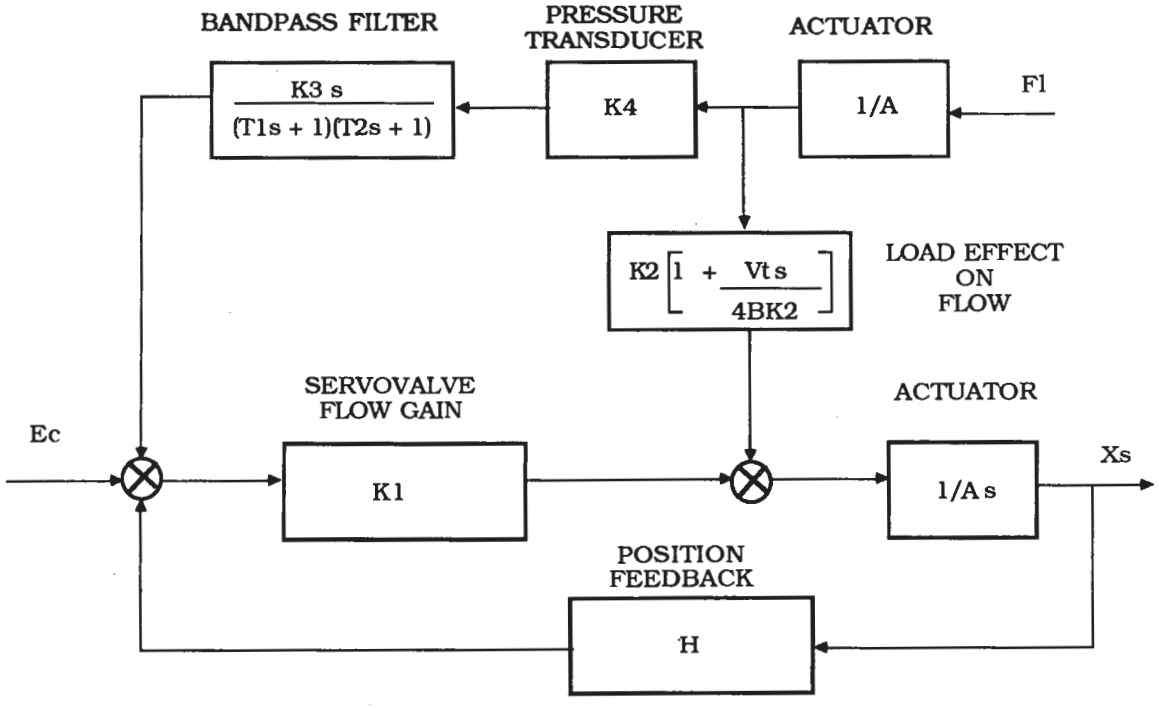
The control surface flutter phenomena are aeroelastic, self-excited vibrations in which airstream energy is absorbed by the control surface. The resulting surface motion consists of bending and rotation, each having a unique frequency and amplitude for a given airstream velocity. If either mode of oscillation is suppressed, classical flutter will not occur. The control surface rotational motion is primarily determined by the spring constant of the actuator and the rotational moment of inertia of the surface. Normally the rotational motion of a control surface is lightly damped. The normal technique for reducing flutter is to raise the resonant frequency by increasing the stiffness of the actuator.

An alternate technique for damping control surface flutter is to change the damping of the surface oscillation to be greater than critical damping, then the motion will be suppressed. This damping could be accomplished by load pressure feedback, thus changing the impedance of the actuator at the flutter resonant frequency. The technique of impedance modification has a good performance payoff by allowing actuators to be sized for maneuver requirements. As an example, the Tornado aircraft uses hydromechanical load-pressure feedback on the horizontal tail actuator for damping flutter.

Figure 2.1-17 shows the block diagram of the flutter suppression actuator. With no-load force, $F_1$, the control loop is a conventional electrohydraulic position control system. The effect of the load force is to reduce the flow to the integrating actuator from the servovalve. This reduction is both a steady state and dynamic effect. The steady state effect is a reduction of the flow to move the actuator because of leakage. This load pressure sensitivity is represented by the term $K_2$. A secondary effect of the load force feedback reduces the differential pressure across the servovalve (which causes a reduction of the valve output flow). This effect was ignored to linearize the model. This analysis assumption does not introduce large errors if the differential load pressures are much less than the differential supply pressure across the servovalve (within the constraint $P_1<(2/3)P_s$). The dynamic effect of the load pressure is due to the compliance of the fluid in the actuator and depends on the rate of change of load pressure. The effect is to reduce the dynamic flow

gain of the servovalve (particularly when the flow is a function of actuator motion).

The load pressure feedback, shown in figure 2.1-17, results in a positive feedback loop. The bandpass filter in series with the differential pressure signal across the actuator drive area provides a lead over a double lag filter function. This filter effectively blocks the low frequency effect of feeding back load pressure. Thus, the static stiffness and steady state position accuracy of the main control loop are not affected.




Notes:

Ec = Command input (volts)
T1, T2 = Bandpass filter time constants
K3 = Bandpass filter gain
K4 = Differential pressure
K1, K2 = Servovalve flow gain
Vt = Actuator fluid volume
B = Bulk modulus of fluid
A = Actuator drive area
s = Laplace operator

FIGURE 2.1-17. DYNAMIC CONTROLS FLUTTER DAMPING ACTUATOR

## 2.2 All-Electric Actuators

Electromechanical actuator (EMA) designs have evolved into a proven technology with a proven record. The material discussed in this section draws upon the discussion of Hair 1985 (refer to AiResearch 1976 for early work). The operation of these actuators is based on the use of high density rare-earth magnetic material and brushless commutated motors, coupled to a mechanical transmission. The motor converts an electric current into torque, which is transferred through the gear box into mechanical motion. The output gearing is connected to the flight control surface with a proper torque speed characteristic for control surface motion.

There are three types of actuator configurations: linear, rotary, and rotary hingeline. For a dual-drive motor configuration, the motor output can be velocity summed or torque summed. The linear actuator uses a ball screw mechanism to transmit mechanical force. Rotary actuators use a planetary gear arrangement for surface motion. A rotary hingeline actuator integrates the actuator into the hingeline and thus into an aerodynamic surface envelope, eliminating the need for any fairing.

### 2.2.1 Actuation Concepts

An actuator must be designed for a specific level of performance. This performance obviously depends upon the control surface application for a particular aircraft. Actuators for the trailing edge will have different performance than for a rudder. Efficient structural load transfer is also a benefit of the hingeline configuration, as illustrated in figure 2.2-1. Each actuator is unique in terms of performance: stall load, no-load speed, operating bandwidth, stiffness, and rotational envelope.

In terms of reliability, one of the first questions that needs answering is fault tolerance. Whether the actuator performance degrades gracefully with multiple faults (i.e., fail-op/fail-safe, fail-op/fail-op/fail-safe, etc.). These performance parameters depend upon the surface application and design limits. Once requirements are established, an electric actuator can be designed to a desired level of fault tolerance. Since faults in components occur more frequently than can be allowed for most applications, the level of fault tolerance determines the ultimate level of reliability for any given actuator. For a typical case, a triplex FCC system (with skewed sensors and reversion mode using analytical redundancy estimators for critical feedbacks) results in a two fault tolerant system. Obviously, redundancy must be applied to the actuation system.

Design to a level of performance is not enough. Performance, reliability, maintainability, cost, size, weight, and power parameters must be met for each application. All these requirements must be taken into the design process and evaluated for each change.

Electromechanical actuation affords considerable flexibility in redundancy designs. Multiple channels can be accommodated in an integrated package. The single motor design, shown in figure 2.2-2, provides actuation where redundant surfaces can be used for fault tolerance. The motor is mounted on one end and,

in the event of motor failure, surface motion is restrained by the brake mounted on the other end. Dual channel actuators, having a single output shaft driven by two motors, can tolerate single failures. A hingeline actuator tends to favor a dual motor for each channel, such as that shown in figure 2.2-3, due to the natural integration of the motors and gearbox. The motor/gearbox outputs may be summed as torques or speed. In the case of a torque summed actuator, both motors are connected along a common shaft. In the event of a failure, the failed motor is decoupled through a clutch and the remaining motor delivers one-half of the original torque. Similarly, a velocity summed dual-channel actuator allows operation at one-half speed, the failed motor being isolated with a brake.

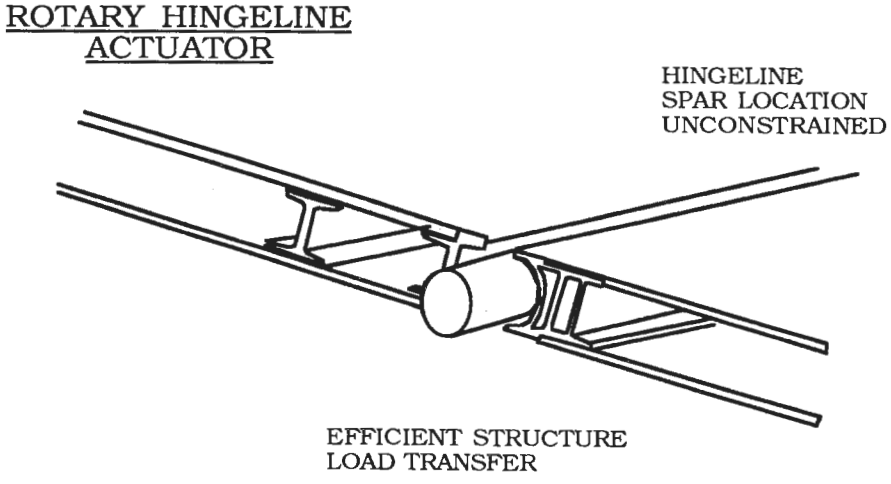



FIGURE 2.2-1. ELECTRIC ACTUATOR CONCEPTS

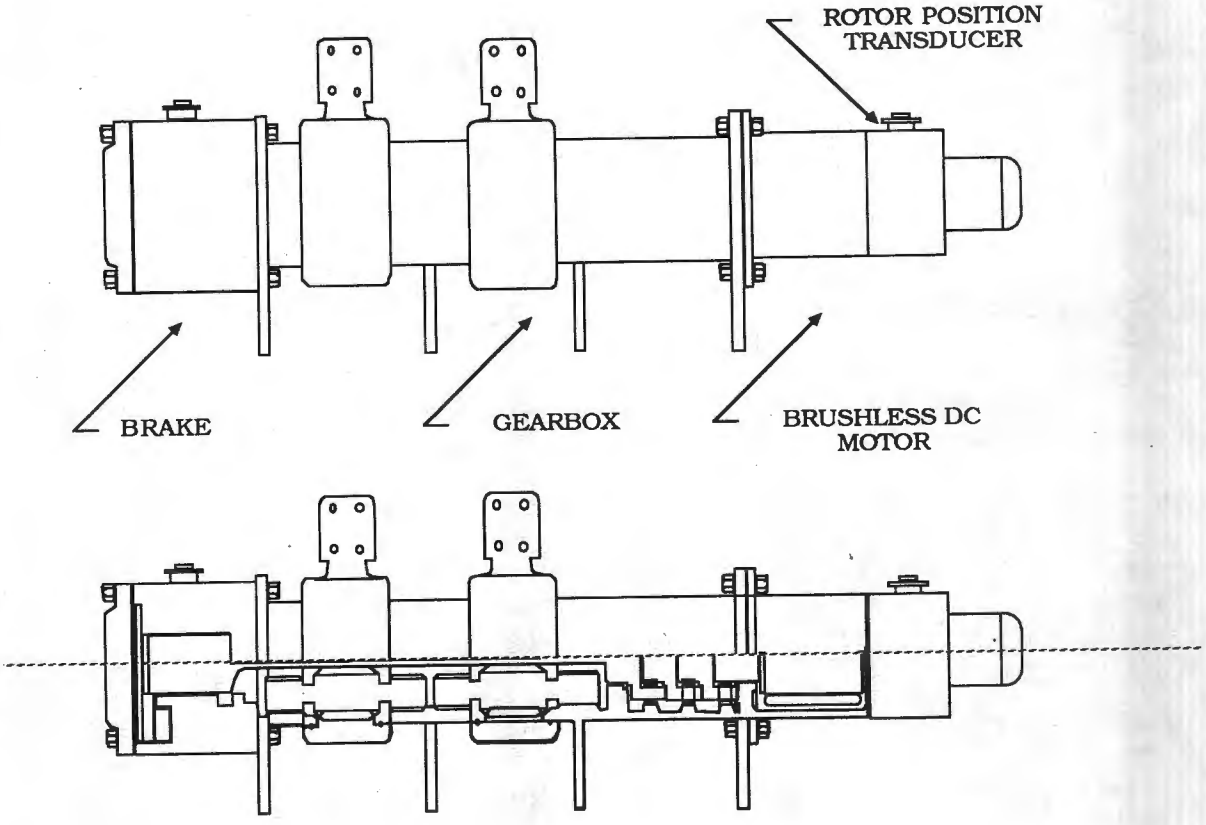



FIGURE 2.2-2. SINGLE MOTOR HINGELINE ACTUATOR

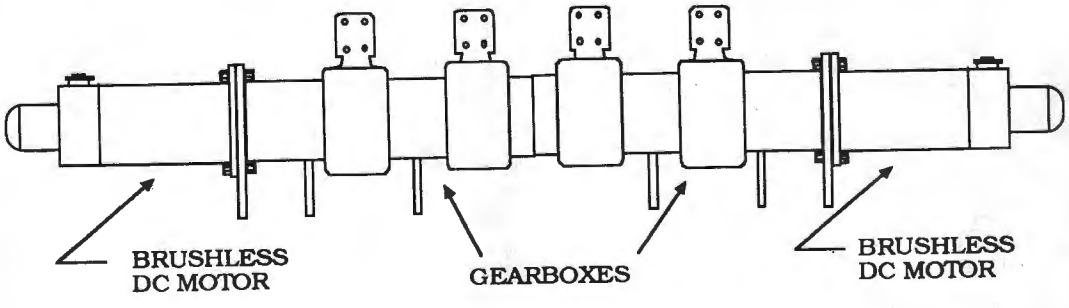



FIGURE 2.2-3. DUAL MOTOR HINGELINE ACTUATOR

Actuation performance under single faults is assured by sizing the actuator for 100 percent performance with loss of a single motor. This requires a 200 percent capability for a dual channel under no-fault conditions. Multiple channel configurations have lower size penalty. A four channel actuator, having four motors driving a common shaft, would be sized for three motors to provide the required torque. This would only require one-third increase in motor capability. If a single motor fails, then this quad configuration operates with full output.

### 2.2.2 Active Isolation

The ability to actively isolate motor and gearbox failures with brakes and clutches is a key advantage for EMAs over hydraulic systems. Although various computer logic and hydro-logic schemes can isolate failures in electrohydraulic actuators, jamming one piston of a dual actuator results in the complete failure of the actuator. The isolation, by brakes or clutches, of a failed motor or gearbox allows the EMA to continue with any single failure.

Thus active isolation results in tolerance for any single failure in a motor/channel. Typical of this case would be loss of power to one motor of a dual channel. A fault detection and isolation system would determine which motor was without power and, for a velocity summed actuator, engage the brake to isolate and contain the failure. If the brake were not engaged, then the motor would back drive the failed motor through the gearbox. A jammed motor could be isolated without the brake. A jammed motor is the critical failure for the torque summed actuator. When one motor jams, the clutch will be activated to release the jammed motor. This requires a signal from the fault detection system. However, the motor remains engaged in the event of a failure in the clutch electrical circuit. A subsequent second motor jam could not be acted upon with a failed clutch circuit.

### 2.2.3 Gearbox Technology

The gearbox is critical for operation of these EMAs. All the designs use some form of gears for mechanical transmission. Rotary actuator designs employ planetary gears in simple and compound arrangements to transmit the appropriate torque and speed. Linear actuators use ball and tube gears having similar criticality and constraints.

Planetary gearboxes have large gear reductions (on the order of 500 and 1,000 to one) and can be constructed in small volumes. The gearbox is the place where all loads and signals come together to form the final force output to the surface. This makes the gearbox the critical link. Without redundancy, an EMA relies on several gears to move the surface. This gear train should have redundancy; the cost of this redundancy is significant in terms of size, weight, and complexity.

### 2.2.4 Reliability

An Air Force sponsored study performed by Grumman (Hair 1985) developed some insight into the relative reliability of EMAS and electrohydraulic systems. For the study two actuators were compared for controlling an F-14's rudder surfaces. A prototype EMA was designed and tested as part of the effort so the results are

a valid point design. The selected electrohydraulic actuator was for the F-14 rudder actuation system. A failure modes and effects criticality analysis (FMECA) was conducted according to defined criticality classes I through IV (results are in table 2.2-1 [Hair 1985]). An initial observation is the increased number of failure modes for the EMA compared with the electrohydraulic unit. This increase is largely due to the increased number of parts, particularly in the gearbox. There is a need to reduce the number of parts in these actuators. Current research is on simpler actuators, development of direct drive valves, and reconfigurable flight control systems relying on simplex actuators.

TABLE 2.2-1. CANDIDATE ACTUATOR FAILURE MODES COMPARISON

| ACTUATOR TYPE/ CRITICALITY CLASS | F-14 ELECTROHYDRAULIC | F-14 ELECTROMECHANICAL |
|---|---|---|
| CLASS I | 3 | 11 |
| CLASS II | 10 | 12 |
| CLASS III | 3 | 0 |
| CLASS IV | 114 | 266 |
| TOTAL MODES | 130 | 289 |

The electrohydraulic actuator studied for the F-14 rudder system only exhibited three failure modes which resulted in Class I failures. These failures could result in the immediate forced landing of the aircraft and probable damage to the aircraft. These two modes are piston rod/rod end mounting structure failures and failures of the valve housing. Consequently, the probability of a failure is very low.

EMAs, on the other hand, exhibited 11 catastrophic failures. Two of these were structural mounting failures while the remaining ones were due to failures of nonredundant gears within the gearbox. Nonredundant gears are the major area for potential failures. The root of this problem is that the gears can fail in a number of ways, including wearing and shearing gear teeth, fracturing, and jamming. There are also multiple ways of failing the actuator controller. All of these failures combine to make the probability of failure larger than for electrohydraulic actuators. Since the structural failure modes have low probability of occurrence, the gearbox modes constitute the major contribution to increased failure probability of this EMA.

## 2.3 Actuation Reliability and Redundancy Technology

Fault diagnosis is the process of determining if a failure has occurred and, if so, what component or subsystem has failed. This information is sent to the

failure management system which determines an appropriate response to the failure. The tasks making up fault diagnosis and failure management are frequently referred to as failure detection, fault isolation, and system recovery and reconfiguration. The information flow for these tasks is illustrated in figure 2.3-1. In this section the contents of these tasks are discussed in general terms drawing from Baker 1988. This material provided a basis for further understanding of fault diagnosis and failure management.

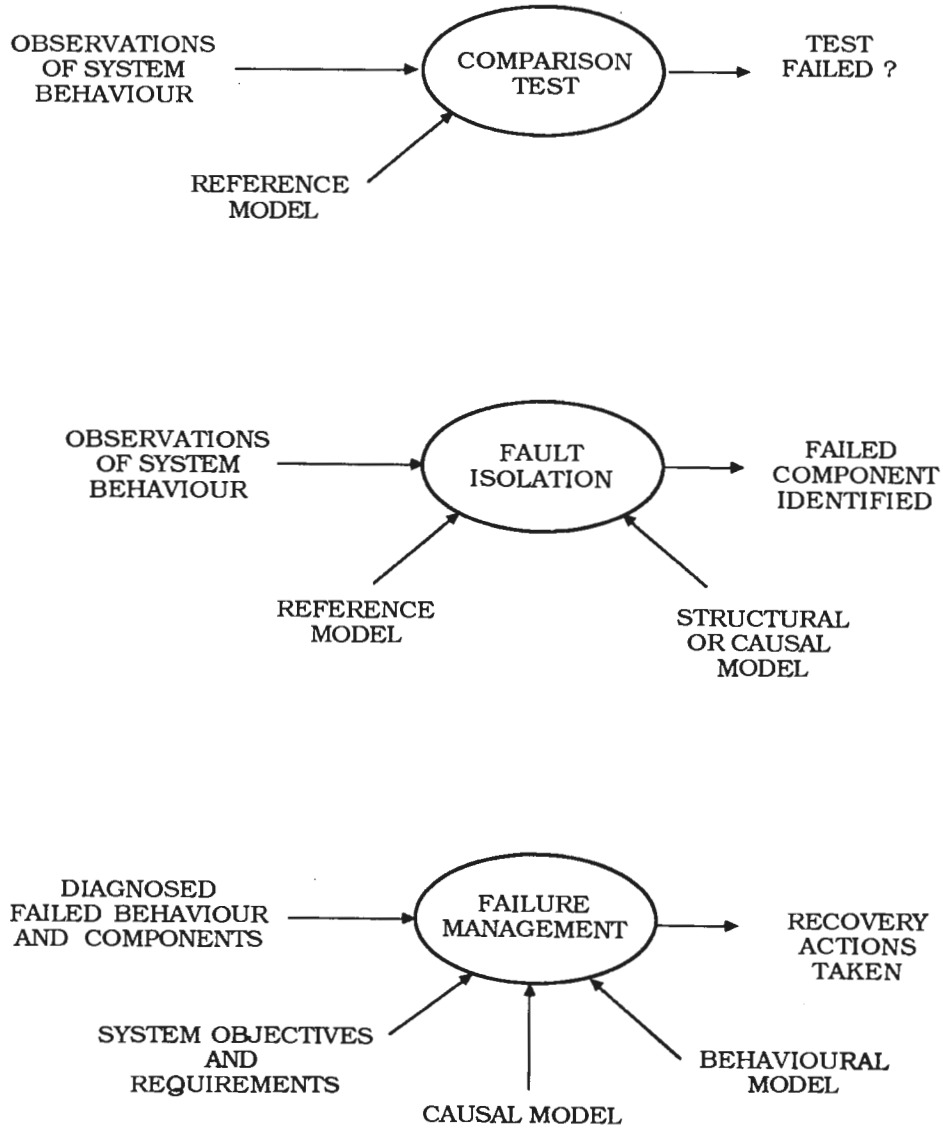



FIGURE 2.3-1. FAULT DIAGNOSIS AND FAILURE MANAGEMENT

### 2.3.1 Failure Detection

Failure detection is the operation of distinguishing between normal and abnormal behavior of a system. The detection process consists of a continuous cycle of monitoring, information processing, and comparison testing.

In general, a failure is detected by monitoring the behavior of a component, subsystem, or system of interest; converting the raw data into a useful form by filtering or numerical processing; and comparing the resulting measurement with the predicted normal values. A failure is detected when the comparison test inputs differ. The outcome of this comparison test is usually a binary value "ok" or "failed". This approach neglects simultaneous failures in the two channels and assumes the comparator did not fail. Common mode failures must be detected in other ways. For example, common mode failures can be detected by monitoring individual channel inputs and outputs and by monitoring the comparator in a calibration test.

Performance of the failure detection system depends upon the sensor measurements, reference models used to predict the measured values (or system state values derived from the sensors), and the comparison test. The sensors are an important part of the detection process because they are an additional source of failure that must also be managed.

Reference models can take on several forms. Use of hardware reference models is referred to as direct redundancy. Use of software models is referred to as analytical redundancy. In the case of software reference models, the most common models are analytical functional relationships. Using these models and information from the system other than direct measurements, a reference for the measured values is synthesized. Accuracy for these reference models can vary from approximate (to detect hardover failures) to highly accurate (to detect soft failures in performance optimization systems).

Either the normal behavior can be modeled or the failed behavior can be modeled. When modeling normal behavior, the reference model can be implemented in hardware or software. For hardware reference models, a duplicate set of hardware can be utilized to develop the normal behavior.

When modeling the failed behavior of a system, modeling all possible component failure modes is very challenging because the failures occur in many different ways and at different times. As a result the failure models are usually approximate.

One failure modeling method in use monitors each critical component or failure mode. The disadvantage of this scheme is that the failure modes must be defined in advance, before a system is fully developed. Nevertheless, there are acceptable systems that detect faults utilizing models of failed behavior.

Two failure models that are commonly used for fault detection are range checking and trend checking. Range checking declares a failure whenever an output exceeds a conservative estimate of the operating range for that variable. Trend checking declares failure when a variable has an abrupt change that is not physically possible.

Threshold detection is the most common fault detection method. For this method a threshold is set on the difference between the model and measured behavior. Errors occur from sensor noise so thresholds must be set high enough to account for these errors. Detection thresholds do not have to be constant. Some systems use scheduled thresholds.

Missed failures and false alarms plague all these fault detection systems. False alarms are of two kinds: removal of good equipment by faulty detection, and failure to remove faulty equipment.

System reliability depends upon the fault coverage of the detection scheme. Careful testing is necessary to verify that all possible faults can be detected by whatever detection method is used.

### 2.3.2 Fault Isolation

The process of isolating faults after a failure has occurred is called fault isolation. Faults may be isolated at a component, card, or channel. The results of this process are correct isolation, incorrect isolation, or no decision.

Approaches are local isolation, arbitration, and generate and test. Local isolation identifies failed components immediately upon test failure. Its implementation relies upon sensors monitoring specific components.

Arbitration uses comparisons of only two sources of information. Other information channels, such as from multiple sources, are tested two at a time and grouped by pairs. The failed components are then identified by using the test results from the pairs to isolate the failed component. The data for arbitration may be either direct data from redundant hardware or indirect data calculated from analytical models

The arbitration method is most efficient when used with sensors, as the outputs can be compared two at a time and isolated directly. When isolating other components or subsystems using arbitration, the sensed information used to isolate a failed component must be separately validated. For example, if individual sensors were used to isolate three redundant components, at least two sensors would be required on each component. With only one sensor per component, a sensor failure cannot be isolated from a component failure. A sensor failure on one of the three components can be isolated if the component being measured could be assumed not to fail at the same time, and if one of the sensors agrees with sensors on the other components.

Most other approaches to fault isolation include generation of a hypothesis of which component has failed and then testing the validity of the hypothesis. This requires identifying candidate failed components and testing to verify if the component failed. If the candidate has failed, isolate it; otherwise repeat the prior steps with another candidate.

Hypothesis testing requires a simulation of candidate component failures and generation of data for comparison with measurements on the system. This process is very challenging to implement for a large system.

There are several benefits and disadvantages of the generate and test method. This is a very powerful method, having fewer sensors and processors than other methods. Indirect methods have some ability to adapt to changes. However, this method has several weaknesses. There is uncertainty in the convergence of the process, the solution times are variable, and the candidate selection process is arbitrary. Computation requirements are greater for generate and test than for other methods of fault isolation.

### 2.3.3 Failure Management

Failure management logic evaluates failures and determines the proper response. Part of this logic must determine, in a very short time, where in the process to return control. Control should be returned at a point where a certain level of performance can be maintained.

The logic process and algorithms for fault management depend upon the process and the equipment still functioning. Few systems take fault isolation information and execute a predetermined sequence of actions.

Recovery usually means to shut down the processor and restart, bringing the system back to a known condition. Two processes are used: the first is reconfiguration to eliminate failed components, and the second is to recover control. Logic is also needed to restore control in a condition where the reconfigured system will operate properly.

The recovery process normally includes other actions taken to correct or minimize failure effects in addition to reconfiguration. These actions may include the following:

- Shutting down operation in a manner to provide a safe condition. Actuators may be locked holding control surfaces in a neutral position. Alternatively the actuators may allow control surfaces to float in the air stream.

- Opposing the effects of failure while the system is reconfiguring. For actuators, this may be a force fight with opposing valves or solenoid coils.

- Returning the system to a condition where control can be resumed. For actuators, this may happen automatically during reconfiguration.

Recovery is generally not needed for actuators as they reconfigure quickly; failures only cause small transients. The discussion in section 2.1 covers the reconfiguration methods for several actuators.

# 3.0 Digital Engine Control Reliability and Redundancy

## 3.1 Reliability Models for FADEC Systems

The utilization of full-authority digital engine controller (FADEC) systems for military and commercial aircraft is well under way (Newirth and Bosco 1980; and Barclay, Lenox, and Bosco 1978). The critical design requirement behind this development is achieving high reliability and availability. Current approaches use redundant sensors, processors, and actuators for meeting these reliability needs.

Design of redundant systems requires analytical tools for supporting the design process. The capability to quickly determine the effect of different design decisions on the reliability of the system is of great importance. Reliability models range from simplified parametric analysis of broad system level descriptions to detailed logic gate level models. There are numerous tools for building and analyzing reliability models including algebraic methods, computer simulation, hardware emulations, and logic analyzers. (Dotson 1988, January and August).

This section provides two examples of reliability models for electronic engine controllers, including a parametric analysis using system level models and a more detailed Markov model. An example is presented describing in detail the development of a Markov model for a dual redundant FADEC system. This model contains the detailed features of the design which impact the reliability. This detailed state model is simple to use and widely applicable. In the final subsection, additional examples are presented illustrating design tradeoffs that can be made using algebraic system level models. Although these models require parameters normally defined by more complex Markov models, they provide insight for system designers.

The operational state of a redundant system changes as the result of component failures, decisions made upon detection of faulty components, and reconfiguration resulting from control system actions to isolate and remove faulty components. Markov modeling techniques provide a natural method for determining the effects of random failures and assessing the effects on system performance. The classical fault tree (combinational) approach is less effective for dealing with changes to system configuration. Markov models easily accommodate the time varying probabilities and effects which occur in reconfigurable systems. Implementation of a Markov model is simplified because first order approximations can be used to define the state transition probabilities.

### 3.1.1 Detailed State Model For FADEC

The principal challenge in developing a Markov model is definition of the model states. As the number of components in a system increases linearly, the number of states in a model increases exponentially. The analyst's challenge is to construct a table of states that does not require too many entries. A complex

redundant system could be partitioned into less complex subsystems and modeled in segments. The results could then be combined logically to obtain results for the larger system (Smith et al. 1976). The approach demonstrated in this section is summarized from Gai, Harison, and Luppold 1983. This modeling approach uses prior knowledge about the system to construct a model having an acceptable minimum number of states in a single model. This is preferable to combining results from many models where any insight from the model on system behavior may be lost in the numerous subelements.

The organization of the components in a FADEC is shown in figure 3.1-1. Two processors form the central part of the system. Each processor, associated power supply, and set of dedicated inputs and outputs make up a single channel. Communication between channels is via a Universal Asynchronous Receiver Transmitter (UART) interface. This serial data channel operates at a low bandwith compared to the processor internal operations. Outputs of both processors are symmetric, allowing either channel independently to control the fuel flow for the engine and control of stator vane angles. Inputs to the system are asymmetric and include the following:

- Power level actuator (PLA) sensors
- Engine pressure (P2, Pb, P5, $P_{amb}$) sensors
- Fan and turbine speed (N1, N2) sensors
- Temperature (T1, T2.5) sensors
- Temperature (TD) diodes
- Feedback sensors from fuel flow actuators (WFB)
- Feedback sensors from stator vane actuators ($SVA_{fb}$)
- Interfaces from independent air data systems (AIR)

During normal operation, the primary channel retains control of the engine until the first failure. When a failure of a sensor or actuator associated with the primary channel is detected, the primary channel substitutes the secondary sensor if available. Failure of a sensor which is not dual redundant results in a transition to a control mode not requiring that sensor, and control continues to reside in the primary channel. Transfer of control to the secondary channel occurs only in the event of failure of the primary processor or its power supply.

The reliability requirement of this engine controller is three shutdowns in a million hours. This is a probability of failure of $0.333 \times 10^{-6}$.

Redundancy management for the system is based on self-tests for failure detection and identification. The self-tests for processors and memories include watch dog timers, parity checks, and memory check sums. Built in self-tests are used for all sensors and interfacing electronics, and wraparound tests are performed

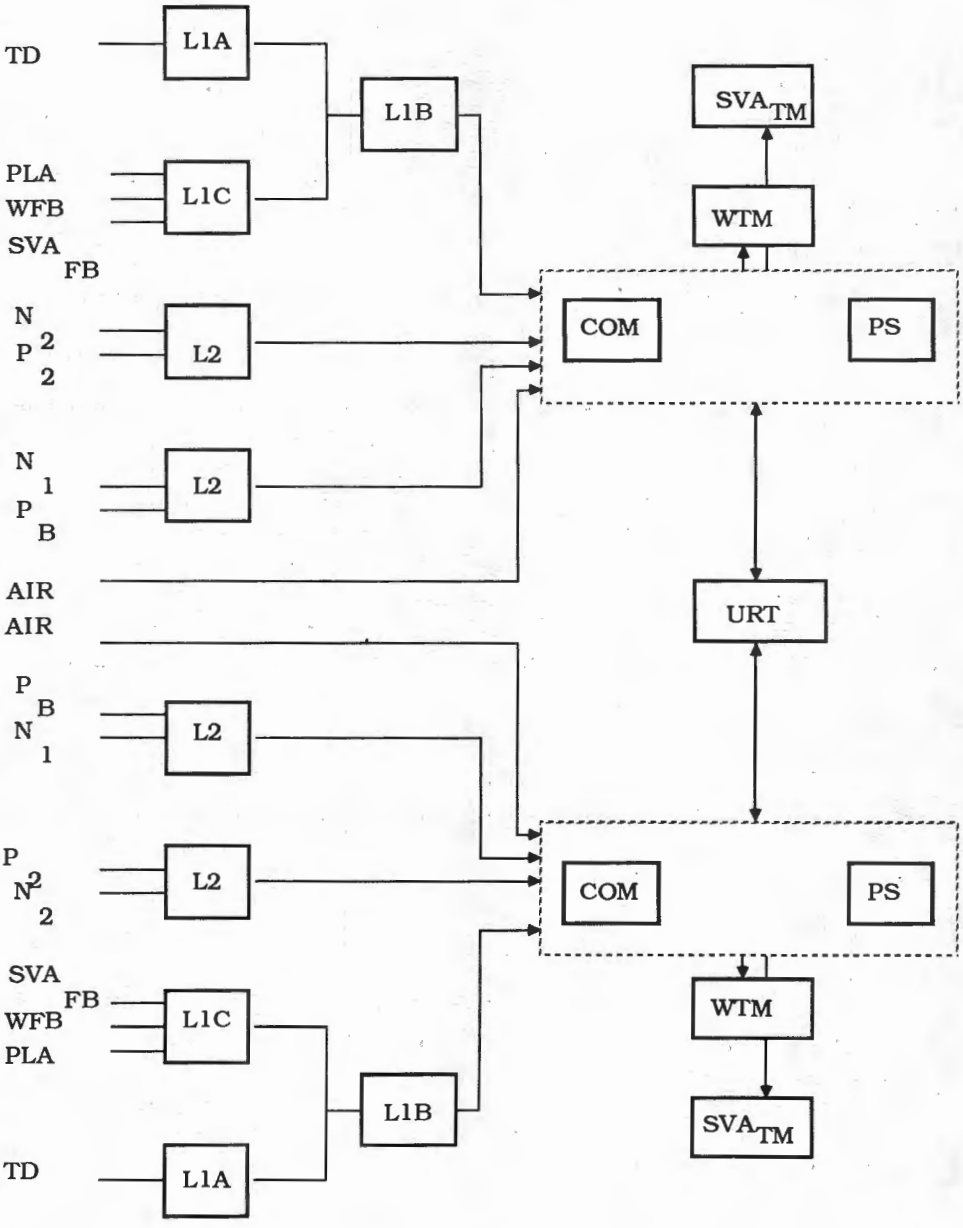



FIGURE 3.1-1. DUAL REDUNDANT ENGINE CONTROLLER

on output commands. In addition, each sensor input is tested for out of range conditions. Other redundancy management techniques include sensor comparisons and use of synthesized versions of sensor measurements as references. These analytical redundancy techniques are not considered in this discussion, but are normally implemented in operational controllers (refer to section 3.2).

The engine controller, shown in figure 3.1-1, operates in four distinct modes: Engine Pressure Ratio (EPR), Low Rotor Speed (N1), High Rotor Speed (N2), and Fuel Flow to Burner Pressure (W/P). The components that support these modes are listed in table 3.1-1. The preferred control method is EPR. For this mode the pressure sensor P5 is used for control. When P5 is not available, control is transferred to the N1 mode. When neither N1 or EPR are available, the N2 mode is used. Loss of N2 reverts to W/P mode.

TABLE 3.1-1. COMPONENTS REQUIRED FOR ENGINE CONTROL MODES

| | SENSORS REQUIRED | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ENGINE CONTROL MODE | PLA | WTM | WFB | T2* | P2* | PB | P5 | N2 | N1 | $P^*_{amb}$ | $SVA_{TM}$ | $SVA_{FB}$ | T2.5 |
| ENGINE PRESSURE (EPR) | X | X | X | X | X | | X | X | | X | X | X | X |
| FAN SPEED (N1) | X | X | X | X | X | | | X | X | X | X | X | X |
| TURBINE SPEED (N2) | X | X | X | X | | | | X | | | X | X | X |
| FUEL FLOW (W/P) | X | X | X | | | X | | X | | | X | X | X |

* This information is derived from available air data.

### 3.1.2 Markov Model

The choice of system states for modeling is made in a systematic manner. Group 1 consists of one state representing the normal mode of operation in which there have been no failures or false alarms. Group 2 consists of those states having one failure or false alarm. Group 3 consists of those states where there have been more than one failure or false alarm.

Group 1 states, being the normal situation, follow from the normal definition of the system diagram. Group 3 states can be aggregated into the inflight shutdown state for any component of a channel or for the sensors. Group 2 states provide the main modeling challenge.

For modeling group 2, first consider the states resulting from fault detection of a failure (or false alarm) of a primary actuator or sensor. The effect of

any detected failure is the same: a sensor or actuator is designated as failed and removed from the system. Thus, for each of the sensors or actuators, one state can be used to model detected failures. Failures or false alarms of dual redundant interface modules are treated in the same way. One state can model effects of covered failures or false alarms of nonredundant components.

If the effect of removing a component from the primary channel is different from the effect caused by removal of its counterpart from the secondary channel, then failures or false alarms of primary and secondary components must be modeled by separate states. The processors, power supplies, and interface modules are in this category since removing them from the primary channel causes the transition from EPR to N1 mode. Removing them from the secondary channel leaves engine control in the EPR mode. Finally, an uncovered failure of any component in the primary control channel is assumed to result in an in-flight shut-down. There are a total of 30 states in group 2. Since any of the states from group 2 can transition to the shutdown state, the 30 states, shown in table 3.1-2, are all that is needed for the model of the dual engine controller.

### 3.1.3 Transition Probabilities

The second step in model development is to determine the probabilities of transition out of the system states for groups 1 and 2.

The transition probabilities for changing states are required to quantify each of the steps shown by the model. They are calculated as being single independent failure events at each time step. This assumption is completely general because the time step interval can be arbitrarily small. The single step transitions for the FADEC under consideration are shown in figure 3.1-2.

Four of the transitions out of state 1 correspond to a change in control mode. These are transitions from the EPR mode to the N1 mode (states 6, 15, 17, and 19). These transitions are due to detected and corrected failures of the P5 pressure sensor, its interfacing electronics, or the computer or power supply which make it available to the system. The transition from state 1 to state 30 accounts for the uncovered failures of components in the primary engine control channel. Finally p(1,1) represents the probability that no transition out of state 1 occurs in a single time step; this is computed as one minus the sum of the transition probabilities of all the other transitions.

As an example of the considerations used in calculating the probabilities, consider the transition out of state 2 (one PLA failure), as shown in table 3.1-3. This transition is due to failures in either of the PLA sensors or false alarms from the primary PLA. In order to calculate the probabilities, it is conservatively assumed that a covered failure of the primary channel caused the system to enter this state. As a result, the UART, processor, power supply, interfacing electronics, and PLA from the secondary channel must be included in the set of required components. Because of the detected failures and control reversion, the system is now vulnerable to failures of these additional components. This would not be the case if state 2 were the result of false alarms associated with a detected failure of the secondary PLA.

The transitions out of state 2 lead to state 26 (EPR mode with two failures), state 27 (N1 mode with two failures), or state 30 (inflight shutdown). The detailed investigation in Gai, Harison, and Luppold 1983 indicates 43 events are included in the transitions to state 26, five events lead to state 27, and 23 events lead to state 30. These computations are beyond the scope of this section and the reader is referred to Gai, Harison, and Luppold 1983.

TABLE 3.1-2. MARKOV MODEL STATE DEFINITIONS

| STATE | DEFINITION |
|---|---|
| 1 | Normal mode, no failure or false alarms |
| 2 | One PLA failure or false alarm |
| 3 | One WFB failure or false alarm |
| 4 | One WTM failure or false alarm |
| 5 | One AIRINC failure or false alarm |
| 6 | P5 failure or false alarm |
| 7 | PB failure or false alarm |
| 8 | One N1 failure or false alarm |
| 9 | One N2 failure or false alarm |
| 10 | One TD failure or false alarm |
| 11 | One LIA failure or false alarm |
| 12 | One LIB failure or false alarm |
| 13 | One LIC failure or false alarm |
| 14 | One L2 failure or false alarm |
| 15 | L3 failure or false alarm on right side |
| 16 | L3 failure or false alarm on left side |
| 17 | COM failure or false alarm on right side |
| 18 | COM failure or false alarm on left side |
| 19 | PS failure or false alarm on right side |
| 20 | PS failure or false alarm on left side |
| 21 | UART covered failure or false alarm |
| 22 | UART uncovered failure |
| 23 | One $SVA_{FB}$ failure or false alarm |
| 24 | One $SVA_{TM}$ failure or false alarm |
| 25 | T2.5 or L4 failure or false alarm |
| 26 | EPR mode with two failures or false alarms |
| 27 | N1 mode with two failures or false alarms |
| 28 | N2 mode with two failures or false alarms |
| 29 | W/P mode with two failures or false alarms |
| 30 | Inflight shutdown condition |

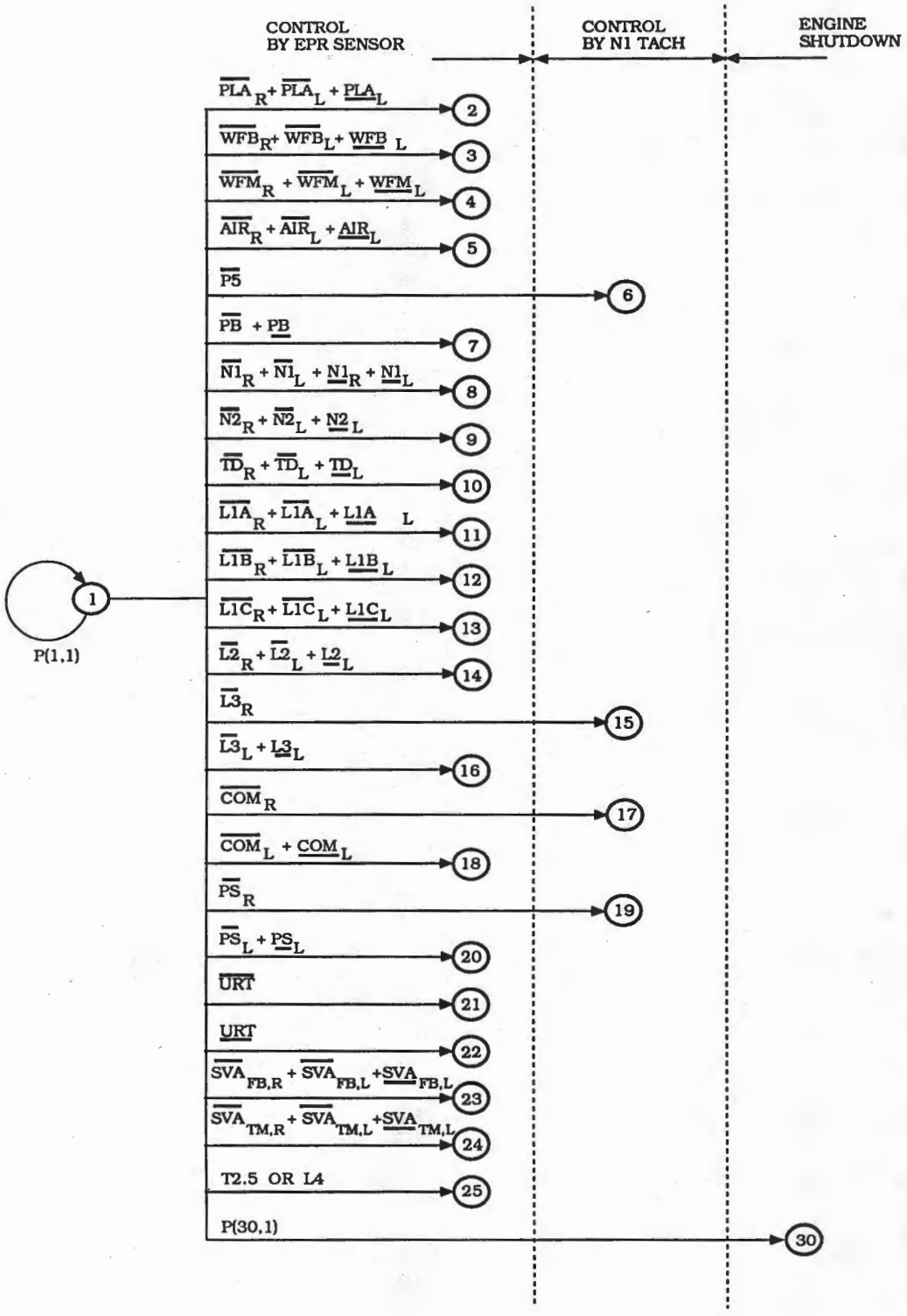



FIGURE 3.1-2. STATE TRANSITIONS FOR ENGINE CONTROLLER OUT OF STATE 1

TABLE 3.1-3. PARAMETERS FOR TRANSITIONS OUT OF STATE 2

| Transitions to State 26 | Transitions to State 27 | Transitions to State 30 |
|---|---|---|
| $\overline{WFB_R} + \overline{WFB_L} + \underline{WFB_L}$ | | $\underline{WFB_R}$ |
| $\overline{WTM_R} + \overline{WTM_L} + \underline{WTM_L}$ | | $\underline{WTM_R}$ |
| $\overline{AIR_R} + \overline{AIR_L} + \underline{AIR_L}$ | | $\underline{AIR_R}$ |
| | $\overline{P5}$ | $\underline{P5}$ |
| $\overline{PB} + \underline{PB}$ | | |
| $\overline{N1_R} + \underline{N1_R} + \overline{N1_L} + \underline{N1_L}$ | | |
| $\overline{N2_R} + \overline{N2_L} + \underline{N2_L}$ | | $\underline{N2_R}$ |
| $\overline{TD_R} + \overline{TD_L} + \underline{TD_L}$ | | $\underline{TD_R}$ |
| $\overline{LIA_R} + \overline{LIA_L} + \underline{LIA_L}$ | | $\underline{LIA_R}$ |
| $\overline{LIB_R}$ | | $\underline{LIB_R} + \overline{LIB_L} + \underline{LIB_L}$ |
| $\overline{LIC_R}$ | | $\underline{LIC_R} + \overline{LIC_L} + \underline{LIC_L}$ |
| $\overline{L2_R} + \overline{L2_L} + \underline{L2_L}$ | | $\underline{L2_R}$ |
| $\overline{L3_L} + \underline{L3_L}$ | $\overline{L3_R}$ | $\underline{L3_R}$ |
| | $\overline{COM_R}$ | $\underline{COM_R} + \overline{COM_L} + \underline{COM_L}$ |
| | $\overline{PS_R}$ | $\underline{PS_R} + \overline{PS_L} + \underline{PS_R}$ |
| | | $\overline{PLA_L} + \underline{PLA_L}$ |
| | $\overline{URT}$ | $\underline{URT}$ |
| $\overline{SVA_{FB,R}} + \overline{SVA_{FB,L}} + \underline{SVA_{FB,L}}$ | | $\underline{SVA_{FB,R}}$ |
| $\overline{SVA_{TM,R}} + \overline{SVA_{TM,L}} + \underline{SVA_{TM,L}}$ | | $\underline{SVA_{TM,R}}$ |
| $(\overline{T2.5 \text{ or } L4})$ | | $(\underline{T2.5 \text{ or } L4})$ |

### 3.1.4 Effect of Maintenance

Redundant fault tolerant systems can operate with one or more failed components and thus have higher reliability than simplex systems. This reliability enhancement only occurs if the redundant system is operated over short times relative to its mean time to failure (Clements 1980). This is because early in the operational life, not all the components are necessary for operation. As time passes and some components fail, more of the redundant components are added to the required set for system operation. As a result the failure probability of a redundant system is not constant, but increases with time. Therefore, the proper use of a redundant system is to operate it over a much shorter time than the mean time to failure. In addition, periodic maintenance is used to replace all the failed components, bringing the unit up to "like new" operational condition. Only in this way can a system be operated with low probability of system failure during the operational cycles.

The effect of the maintenance time interval on the system failure probability can be demonstrated with a simple model. Let N(t) be the number of failures per hour at time t, where $t \ll T_s$ the mean time to failure for the system. If, during operation, maintenance is performed every $T_m$ hours, where $T_m \ll T_s$, then the expected number of failures in t hours is

$$N(t) = (t/T_m) * P_{isd}(T_m) \qquad [1]$$

The probability of failure of a dual redundant system with coverage C is given by

$$P_{isd} = (T_m/T_s) * [(1-C) + T_m/T_s] \qquad [2]$$

Equations 1 and 2 imply that the number of failures in a million ($10^6$) hours with maintenance is

$$N_m = (10^6/T_m) * [(1-C) + T_m/T_s] \qquad [3]$$

The parameter $T_s$, mean time for system failure, is determined from the Markov model solution. Since this is beyond the scope of this section, the reader may refer to Gai, Harison, and Luppold 1983 for details. The steps involve determining the model parameters for each of the state transitions in figure 3.1-2, and development of powers of a matrix, or alternatively, solutions of electrical circuit equations. The base line values for the component failure rates used for these calculations are shown in table 3.1-4.

The results of solving the Markov model for the probability of being in a given system state are as follows:

| | |
|---|---|
| Operating in EPR Mode | 0.995 |
| Operating in N1 Mode | $4.3 \times 10^{-3}$ |
| Operating in N2 Mode | $3.2 \times 10^{-6}$ |
| Operating in W/P Mode | $1.2 \times 10^{-6}$ |
| Inflight Shutdown | $4.0 \times 10^{-4}$ |

TABLE 3.1-4. BASE LINE COMPONENT FAILURE RATES

| Component | Probability of Failure in 1 Hour |
|---|---|
| PLA | $2.7 \times 10^{-6}$ |
| WFB | $2.7 \times 10^{-6}$ |
| $SVA_{FB}$ | $1.0 \times 10^{-6}$ |
| WTM | $1.3 \times 10^{-6}$ |
| $SVA_{TM}$ | $1.3 \times 10^{-6}$ |
| AIR | $2.5 \times 10^{-6}$ |
| P5 | $4.6 \times 10^{-6}$ |
| N2 | $1.0 \times 10^{-7}$ |
| TD | $1.6 \times 10^{-8}$ |
| L1A | $1.0 \times 10^{-6}$ |
| L1B | $2.1 \times 10^{-6}$ |
| L1C | $9.7 \times 10^{-7}$ |
| L2 | $1.4 \times 10^{-6}$ |
| L3 | $1.7 \times 10^{-6}$ |
| COM | $2.1 \times 10^{-5}$ |
| PS | $3.6 \times 10^{-6}$ |
| URT | $5.0 \times 10^{-6}$ |
| N1 | $1.0 \times 10^{-6}$ |
| PB | $4.6 \times 10^{-6}$ |
| T2.5 AND L4 | $2.7 \times 10^{-6}$ |

The probability of operating in the base line EPR Mode is very high (0.995); the probability of inflight shutdown is fairly low ($4.0 \times 10^{-4}$). As a check, assume perfect coverage, C=1, and use equation 3 above to calculate a failure rate of 2.7 shutdowns in a million hours of operation.

Two other computational results are interesting. Figures 3.1-3 and 3.1-4 show the sensitivity of inflight shutdown to maintenance interval and fault coverage. These results indicate little sensitivity to maintenance interval for a 0.9 coverage. However, the maintenance interval is very important for lowering the probability of shutdown at higher levels of fault coverage. For a simplex system, inflight shutdowns are approximately eight in a million hours as shown by the curves in figure 3.1-3 for 0 coverage. Achievement of the 2.7 per million hours requires a fault detection and recovery system having fault coverage on the order of 0.95.

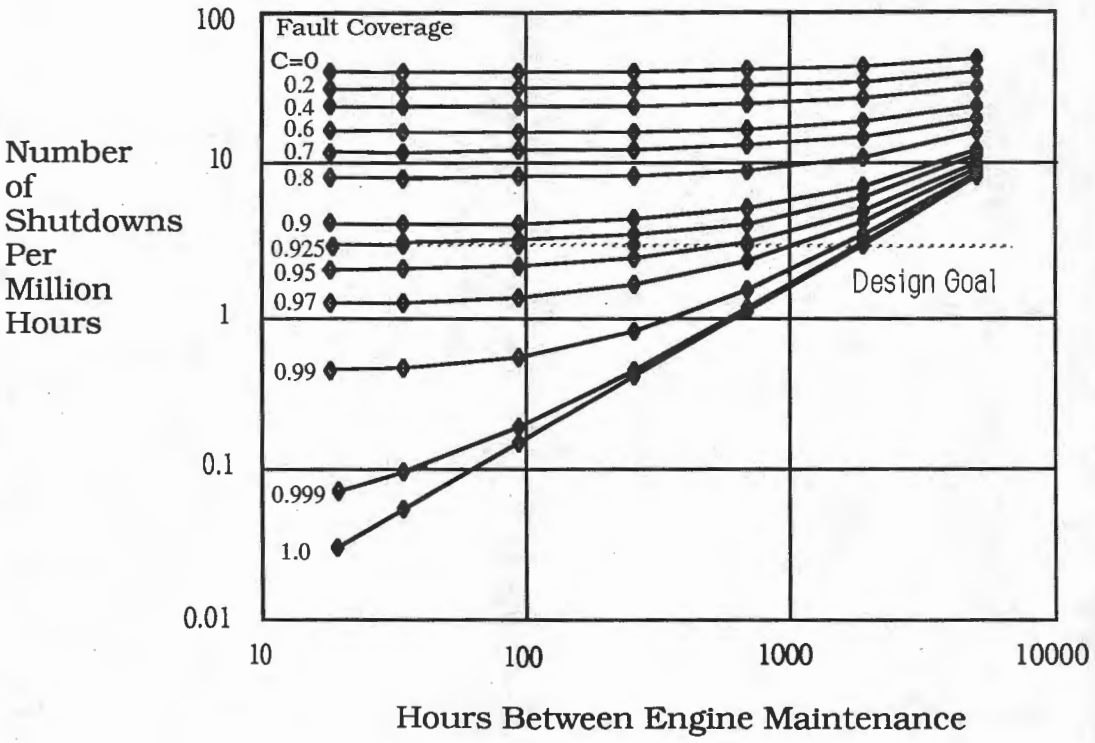



FIGURE 3.1-3. NUMBER OF SHUTDOWNS PER MILLION HOURS OPERATION - DUPLEX

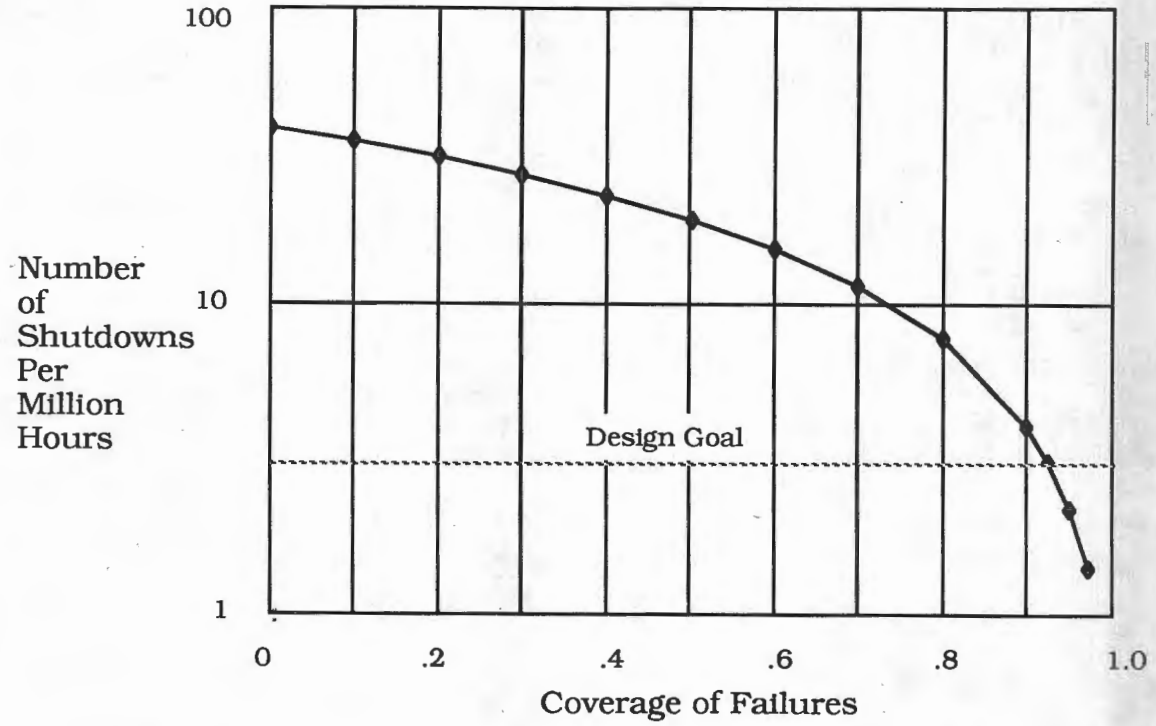



FIGURE 3.1.4. SENSITIVITY OF INFLIGHT SHUTDOWN TO COVERAGE

These results illustrate that attempting to improve system reliability by reducing the maintenance cycle limits the effectiveness of this approach unless very high levels of fault coverage can be achieved. For a controller having $T_m$ of 150 hours and a mean failure time, $T_s$, of 15000 hours, coverage in excess of 0.99 is required to reduce the number of shutdowns by shortening the maintenance cycle.

### 3.1.5 Markov Models for System Safety and Availability

In the discussion of this subsection, the emphasis is on the two fundamental reliability objectives: system reliability and system availability. In general terms, flight safety is determined by having the minimum suite of equipment to insure no loss of life. Availability is the probability of being operationally ready at any point in time. In terms of safety, the probability of failure at a given time provides a measure of system safety. Reliability modeling is used to define and control the design for meeting the system safety and availability goals.

There is a fundamental conflict between high safety and high availability whenever redundant equipment is used for providing high reliability. For example, consider a system having a failure rate of 100 per $10^6$ hours (MTBF is 10,000 hours) in operations having a two-hour flight duration. This system has a failure probability ($P_f$) of $2 \times 10^{-4}$ and an availability ($P_a$) of $2 \times 10^{-4}$. The situation is quite different if a triplex (triple redundant) system is configured from these components. This new system has an improved failure probability of $8 \times 10^{-12}$ (i.e., $P_f^3$), but a reduced availability of $6 \times 10^{-4}$ (i.e., $3 \times P_a$) and MTBF=3333 hrs. Thus, while the triplex reliability is increased dramatically (nearly eight orders of magnitude) over the simplex system, the system availability has decreased by a factor of three.

This safety/availability conflict can be summarized simply. Addition of equipment in redundant or standby mode increases the likelihood that enough equipment will be working to provide safe flight. For adding redundant equipment, increasing the number of components increases the number of failures (not the system failure probability) in a given time. Since all the components are needed for dispatching the system, redundancy reduces system availability while increasing the useful operating time.

Since perfect reliability is not attainable, it is much more important to recover from failed components than to prevent them (Wulf 1975).

The probabilities can be calculated in broad terms using a system level Markov model having the following parameters:

| | |
|---|---|
| $L_t$ - Total Failure Rate | Rate of component failures |
| $L_n$ - Nonfunctional failure rate | Failures that have no effect on the system operation |
| $L_f$ - Functional failure rate | Failures that affect system performance |

| | |
|---|---|
| $L_u$ - Uncovered failure rate | Functional failures that are not detectable or recoverable |
| $L_c$ - Covered failure rate | Failures that may be detected, isolated, and recovered. |

These terms are interrelated

$$L_t = L_n + L_f \quad [4]$$

$$L_f = L_u + L_c \quad [5]$$

For a single piece of equipment, coverage may be defined as

$$C = 1 - L_u/L_f \quad [6]$$

For a system or channel the coverage may be defined as

$$C = [\Sigma C_i * (L_c)_i]/[\Sigma(L_c)_i] \quad [7]$$

where $C_i$ is the coverage for the i th component and $(L_c)_i$ is the corresponding failure rate.

Coverage is a term that requires careful attention to define properly. It is most accurately defined by testing the system by the fault injection techniques discussed in Chapter 5. In a multi-channel system, different coverage values are defined depending upon first or second failures. The first failure in a redundant system is more readily detected by the hardware and software implemented for this purpose (coverage often approaching one). Second failures in a duplex and third failures in a triplex being dependent upon self-test BIT and BITE mechanisms have a lower value of coverage (0.95 to 0.999).

The relationships defining system probability of failure in terms of component MTBF, component reliability (R), availability (Q), and coverage are as follows for duplex, triplex, and quadruplex systems (Hamilton Standard, August 1980):

- Duplex

$$P_{fl} = 2RQ(1-C_1) + Q^2C_1 \quad [8]$$

- Triplex

$$P_{fl} = 3R^2Q(1-C_1) + 3RQ^2C_1(1-C_2) + Q^3C_1C_2 \quad [9]$$

- Quadruplex

$$P_{fl} = 4R^3Q(1-C_1) + 6R^2Q^2C_1(1-C_2) + 4RQ^3C_1C_2(1-C_3) + Q^4C_1C_2C_3 \quad [10]$$

Where R = 1-Exp(-MTBF x hrs), Q = 1 - R, and $C_1$, $C_2$, and $C_3$ are the coverage of the first, second, and third failures respectively.

An important characteristic of coverage may now be illustrated. The likelihood of failure is extremely sensitive to the coverage of first and second failures. The first failure coverage in a triplex and the second failure coverage in a quadruplex is often considered to be unity. This is because detection techniques for first failures use cross-channel communications and direct comparisons. Changing this assumption on coverage slightly has a large effect. For example, changing the value of first failure coverage from 1.0 to 0.999 is summarized in the following figures for a system having a MTBF of 10,000 hours:

| | $P_f$/hour ($C_2 = 1.0$) | $P_f$/hour ($C_2 = 0.999$) |
|---|---|---|
| Duplex | $8 \times 10^{-6}$ | $8 \times 10^{-6}$ |
| Triplex | $1.2 \times 10^{-9}$ | $3 \times 10^{-7}$ |
| Quadruplex | $1.6 \times 10^{-13}$ | $4 \times 10^{-7}$ |

The duplex system is only shown for reference as it has no coverage for second failures. Such a change in coverage might result from ruling out cross-channel checking in a system design. The interesting point made by these figures is that the apparently small change in coverage for second failures has degraded the triplex and quadruplex to the extent that both these systems have approximately the same probability of failure.

Effects of periodic maintenance on the number of shutdowns can be developed parametrically using these algebraic models. The discussion in section 3.1.4 also used these system models for a dual system. The number of failures in a given number of hours is directly proportional to the time, provided that the probability of failure is small during the time interval. Following periodic maintenance, any system components that have failed are replaced and the system is in a "like new" state. As long as the maintenance time interval is small compared with the mean time between component failures, the number of failures can be determined from the probability of system failure.

For a triplex system an expression for the number of failures in a million hours, derived from equation 9 is

$$N_f = 10^6/T_m[3R^2Q(1-C_1) + 3Q^2C_1(1-C_2) + Q^3C_1C_2] \qquad [11]$$

where $T_m$ is the hours between periodic maintenance. The other terms are defined in equation 10. Figure 3.1-5 shows the result of these calculations for first failure coverage of 0.999 and second failure coverage ranging from 0 to 1. Comparison of the data with that in figure 3.1-3 illustrates the benefit for a triplex, even for small values of coverage for the second failure.

## 3.2 Analytical Redundancy Design for Improved Engine Reliability

Control system reliability requirements for the next generation of aircraft propulsion systems are leading engine control systems toward dual or even triple redundancy. Sensor redundancy can be minimized by use of analytical models that provide signals for a failed sensor. These signals are derived from aircraft parameters and measurements using other sensors. This minimizes the need for physically replicated sensor sets. Analytical redundancy used together with a

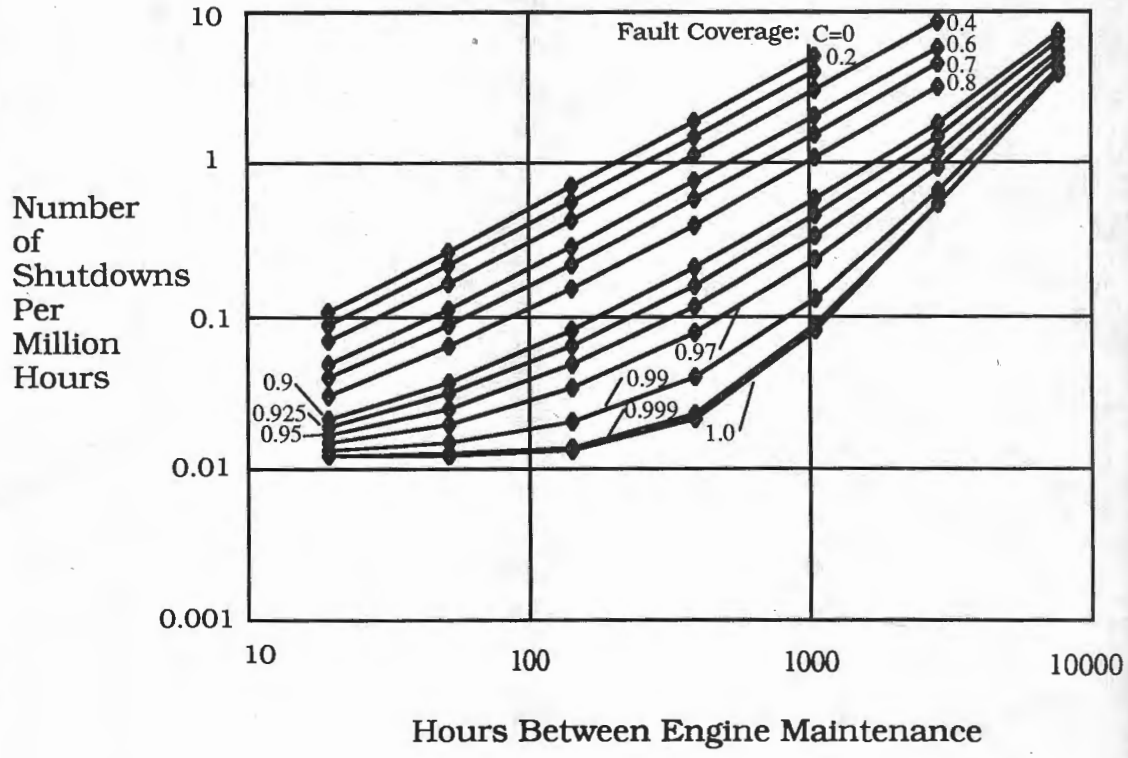



FIGURE 3.1-5. NUMBER OF SHUTDOWNS PER MILLION HOURS OPERATION - TRIPLEX

This section describes an analytical redundancy approach for increasing the reliability of aircraft engines reported in Swan and Vizzini 1988 and Brown and Vizzini 1986. (Also refer to Detroit Allison June, 1982 for a fighter aircraft study.) The original work was sponsored by the Naval Air Propulsion Center as the FADEC program (Vizzini and Toot 1980) and designated as a Failure Indication and Corrective Action (FICA) system. The initial work established the feasibility of this approach. The program represents an extension of the initial effort for a one-of-a-kind technology demonstrator to an operational product engine controller. Included in the new work is an analytical demonstration of operation over a full range of engine modes and parametric envelopes. Also included is a demonstration of fault isolation and recovery from failures in sensors and actuators. These investigations utilized a non-linear analytic model for a GE23A jet turbine engine to provide operational parameters and real-time modeling.

### 3.2.1 Analytical Redundancy Engine Control Concept

The control system layout, shown in figure 3.2-1, was used for the investigation reported in Brown and Vizzini 1986. It contains three main modules. The GE23A engine system predicts the outputs of engine actuators and sensors as a function of the sensed environmental conditions, actuator signals, and model updates. This engine model has been validated by prior work to represent the operation of a real engine, and is used to simplify the experiment and for providing known engine environments. The engine actuator inputs drive the non-linear engine model which provides sensor data for the control system. The engine module in the controller is an analytical model specifically designed to support the analytical redundancy system by providing model predictions for sensor outputs. The component tracking filter uses the sensor data and model values to compute differences between the sensed values and model values. This information is used to generate model updates and error information for failure detection and isolation logic. The failure-accommodation module logic computes the validated sensor information as a function of the sensor and actuator status, model values, and sensor data. In the event of actuator failure, this module provides the data for reconfiguring the engine control mode.

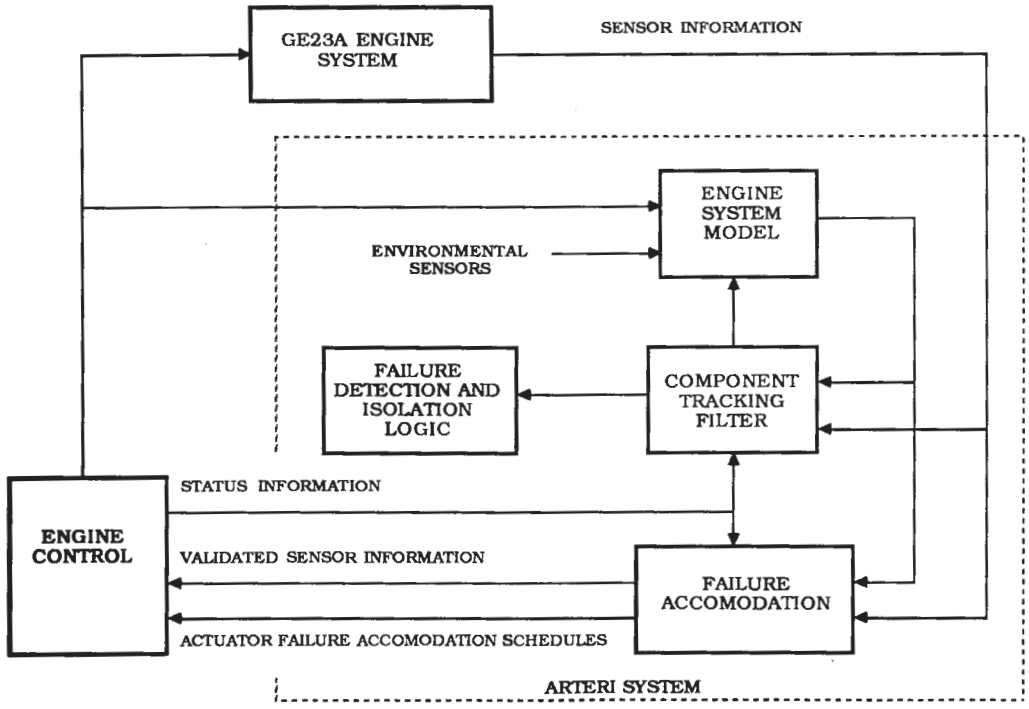



FIGURE 3.2-1. SCHEMATIC OF ANALYTICAL REDUNDANCY ENGINE CONTROLLER

### 3.2.2 Component Tracking Filter

The component tracking filter is the key element in this system. This adaptive filter updates engine module parameters as a function of differences between sensed and modeled values. This filter eliminates the need for manually tuning the engine model to an individual engine. Manual tuning was required for prior engine modules to accommodate differences between engines of the same type number.

There are two types of sensor failures: hard failures, where the sensor output violates physical limits on position or rate of change; and soft failures, where the sensor calibration gradually changes and slowly drifts, providing erroneous but believable values. Hard failures are the easiest to detect by parameter range checks or by range rate checks. Soft failures are much more difficult to detect.

The Analytical Redundancy Technology for Engine Reliability Improvement (ARTERI) system was designed to detect, isolate, and accommodate soft sensor failures while using the same sensor data to update the engine module parameters. The model accuracy needed for accommodating sensor failure is achieved by updating the engine module parameters to match the actual engine under control. This update process minimizes the errors between the sensed and measured values. Depending upon the rate of model update and the drift rate for failed sensors, the model may keep up with the failed sensor and corrupt the engine module. This makes fault detection and isolation difficult and may lead to high false alarms and removal of good equipment. All single-channel model-filter-based FDFM schemes have this same limitation. The ARTERI approach minimizes false alarms by integrating the functions of fault detection, isolation, and reconfiguration.

The approach to FDFM also uses the adaptive nature of the component tracking filter. Once the component tracking filter parameters were updated to match the actual engine parameters, the filter gain was reduced. Filter design requirements included the need to produce unique failure signatures, minimize corruption by soft failures, and provide a smooth transition to the model value. Smooth transitions were accomplished by decoupling the filter update process and by gain controls.

Sources of mismatch between the model and engine data are modeling inaccuracies and measurement uncertainties. Engine to engine differences occur from component quality and variation. Modeling approximations were necessary to meet the requirements of real-time operation. Measurement uncertainties are the random variation of a sensed parameter from its true value and the noise in electrical measurements. Compared with modeling inaccuracies, the measurement uncertainty is considered to be a small source of mismatch between modeled and measured data.

Two kinds of error result from model inaccuracy: steady state errors and dynamic errors. Steady state errors are the constant or near constant differences between measured and predicted values. These errors are due to parameter differences and biases in the measurements. Dynamic errors are the differences that occur during transient conditions such as during engine speed up and slow down. These errors are due to internal iteration loops, approximations for the component models, and high frequency sensor noise.

The component tracking filter, shown in figure 3.2-2, integrates four separate updates: state update, steady state error update, input update, and component update. Mismatch is minimized by a frequency decomposition of the sensed engine parameters and the model values. Actuator state updates are computed from position errors and a diagonal matrix of update parameters. This portion of the filter adjusts for steady state and dynamic errors. Engine model updates for steady state errors are computed as a function of time delayed differences between sensor measurements and predictions. Dynamic errors are treated as the difference between measured and predicted plus the sum of the steady state errors. Time constants for the filter lag are selected to be four to 10 times slower than the dominant engine response time. Engine state updates are computed as a function of the dynamic errors and an update matrix. The structure of this update is similar to a Kalman filter. However, only measured states are updated. The same structure is used for the component and updates loops. These updates drive the steady-state errors towards zero.

The structure of the component tracking filter allows the various filter updates to be computed at different sample rates. This reduces the throughput requirements of the filter. Since the state updates are related to the dynamics of the model, they are computed at the same 10 to 20 millisecond rate. The steady state error and input updates are low frequency loops computed at a rate four times slower. The lower rate is based on the assumption that the component performance levels change slowly over the life of an engine and component updates may be computed even more slowly.

For performance over the full engine operational envelope, the engine controller gains are scheduled as a function of power setting and flight condition. Initial gains are scheduled as a function of fan speed and sea-level operational conditions, thus accounting for various power settings. These first order curve fits are modified as a function of flight condition (altitude and mach number) based upon measured engine inlet environments. Inlet temperature and pressure measurements, divided by their sea level values, provide scale factors to correct the sea level model errors for speed, temperature, and pressure.

### 3.2.3 Failure Detection And Isolation

Errors computed from the component tracking filter form three separate error vectors. These vectors define an n-dimensional space, where n equals the number of vector components. When a failure occurs, the path that the vector follows is defined as that failure vector's signature.

The criteria established for computing the engine state update and input updates decouple the error vectors. When the error vectors are decoupled, the failure of one component will not affect the error signals for the others. The update criteria are designed as follows:

- Design the state update matrix so that the error produced by a failure in one sensor does not create errors for other sensors because of an update.

- Design the input update matrix so that the error produced by failure in one input does not create an error in other input channels because of the update.

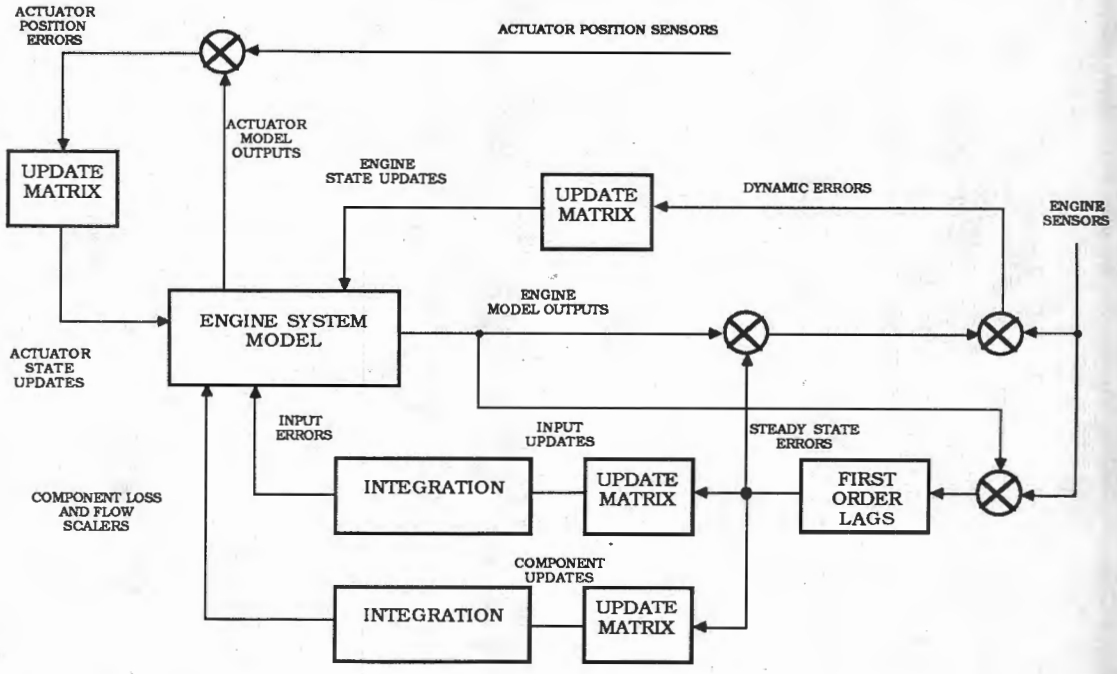



FIGURE 3.2-2. COMPONENT TRACKING FILTER

As a result of this decoupled design, engine state and input updates produce axial signatures. When failure occurs with this design, a change in error signal will only be observed in the failed channel. This design method is discussed at length in Brown and Swain 1987. The advantage of the axial error vector is that the error signatures only need to be tested; storage of the vector is not required. Out of bound indications will only appear in the error signal of the failed component. Bounds on the variation and rate of variation need to be established for each component. The individual error signals are compared with the established bounds to detect faults.

Examples of steady state and dynamic failure signatures for a soft failure in a fan speed sensor are shown in figures 3.2-3 and 3.2-4, respectively. Note that for these plots the threshold value for each channel is the full scale value (+FT and -FT) for each plotted parameter. Failure isolation can easily be accomplished with the data shown.

### 3.2.4 Failure Management

Failure management logic is based upon the following priorities: first, engine safety; and second, maintaining thrust and specific fuel consumption.

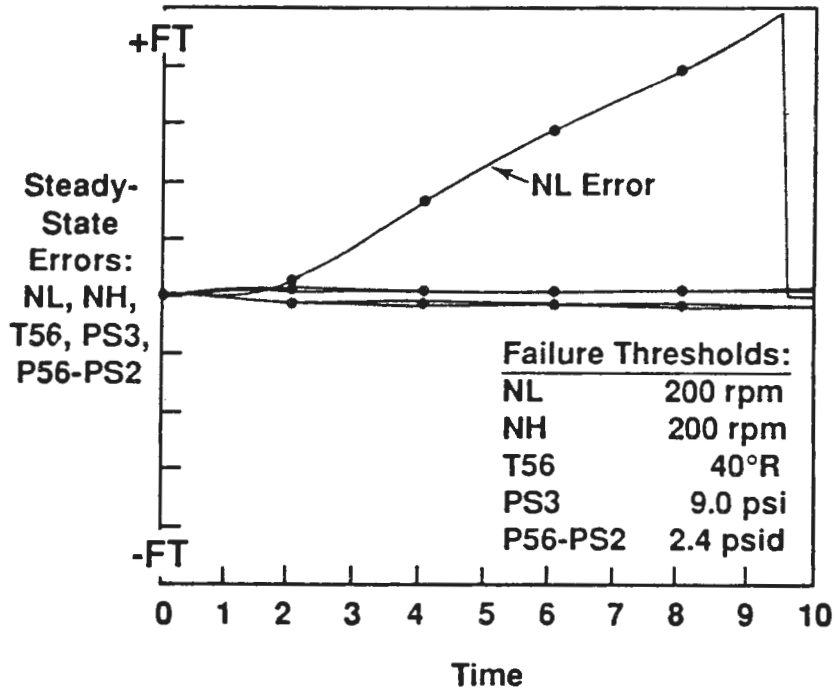



FIGURE 3.2-3. STEADY STATE ERROR DETECTION - FAN SPEED SENSOR

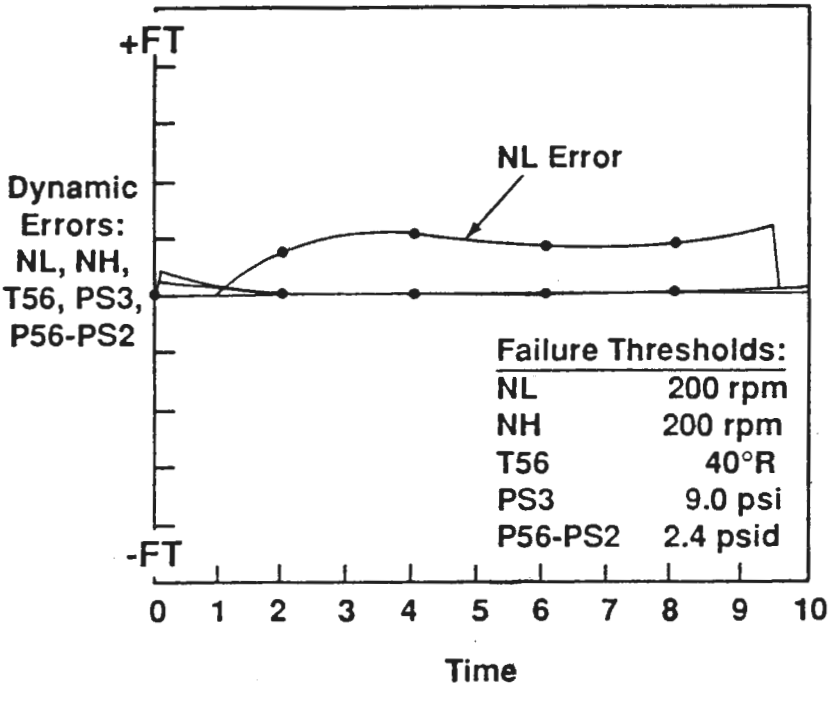



FIGURE 3.2-4. DYNAMIC ERROR DETECTION - FAN SPEED SENSOR

Failures of sensors and actuators must be detected and acted upon to isolate and correct engine control modes before any serious engine damage or loss in performance occurs. Corrective actions for sensor or actuator failures should not induce any large transients that could lead to loss of control. These corrective actions are accomplished by logic activated by error signals exceeding error threshold values.

Sensor failures are accommodated by analytical redundancy where the analytical sensor models replace the failed sensor. Smooth transitions are accomplished by computation of the valid sensor value from the sensor value, model value, and control gains. Actuator failures are accommodated by reconfiguring the control mode using fail-fix servovalves and failure accommodation schedules.

The primary failure usually observed for an actuator is a short or open circuit of a torque motor. When such a failure occurs, the fail-fix servovalve will cause the actuator to move to a preferred position. This position (open or closed) is selected to maintain safe engine operation and maintain performance as high as possible. The control schedules for the remaining actuators use precomputed schedules to minimize loss of engine performance. When safe operation cannot be maintained, the engine will be shut down. For other actuator failures, the corrective action is to shut off the torque motor allowing the fail-fix servo actuator to move to the preferred failure position.

Table 3.2-1 (Brown and Swain 1987) summarizes the results of the actuator failure accommodation studies. Note that for a compressor variable stator vane (VSV) failure, a fully open direction could result in blade flutter and possible engine damage, while a fully closed direction considerably reduces thrust.

### 3.2.5 Test Results

A number of tests were conducted to validate the design and implementation of the prototype engine controller. These tests included convergence tests to verify operation with different engines over the operational envelope; transient tests to verify the ability to track during large transient changes in engine operation including acceleration and deceleration; and fault detection, isolation, and accommodation tests to verify the controller ability to detect and correctly handle hard and soft failures.

Results for the soft failure tests, being the most difficult, are summarized in table 3.2-2. Figure 3.2-5 illustrates the resulting corrective action for a fan speed sensor failure. Note that the closed loop control for fan speed maintained constant speed prior to isolation of the failure. The controller tracks (maintains the demanded speed) from the failing sensor signals by increasing fuel flow. This is evident by noting the change in fan speed computed by the model. Also note that normal engine operation was resumed by the time the failure was detected. The results of accommodating a soft failure in the main fuel metering valve actuator sensor (WFM) is shown in figure 3.2-6.

The program (Brown and Swain 1987) summarized in this section developed an operational prototype engine controller having fault tolerance for both hard and soft failures. Operation of the prototype was validated by tests using a

non-linear simulation of a GE23A turbofan engine, although the design is adaptable to any engine.

TABLE 3.2-1. ACTUATOR FAILURE ACCOMMODATION TEST RESULTS

| Actuator | Preferred Failure Direction | Effect on AB Operation | Effect on Dry Operation | Fault Accommodation Schedules |
|---|---|---|---|---|
| Afterburner Fuel Metering | Closed | Not Permitted | Normal Operation | N/A |
| AB. Exhaust Nozzle Area | Closed | Not Permitted | Maximum of 12% Thrust Reduction of 46% Reduction in Fan Stall Margin for Certain Altitude and Mach Number above 15,000 ft. and Mach 1.5 at IRP<br><br>Increase in Flight Idle Thrust | Reduced Corrected Fan Speed Control Reference Schedules |
| VSV. Compressor Variable Stator Vane | Closed | Not Permitted | Shut Down Engine | N/A |
| WFM. Main Fuel Metering Valve | Closed | Not Permitted | Shut Down Engine | N/A |

Tests on the prototype control of a simulated GE23A engine demonstrated the capability for achieving the following:

- Updating the control system engine model to match engine variations due to engine component quality and deterioration.
- Detecting and isolating hard and soft failures of environmental sensors, internal engine sensors, actuator position sensors, and engine actuation systems.
- Maintaining normal engine operation following single sensor failures.

- Maintaining safe engine operation following single failures of engine actuators.

Continued development and testing with a demonstrator engine is recommended.

Analytical redundancy appears to be viable for application to engine control and there should be commercial applications for this technology.

TABLE 3.2-2. SOFT FAILURE TEST SUMMARY

| | Failures Simulated | Result |
|---|---|---|
| Engine Sensors | NL, Bias; Ramp ± 50 rpm/sec<br>NH, Bias; Ramp ± 50 rpm/sec<br>T56, Bias; Ramp ± 5° R/sec<br>PS3, Bias; Ramp ± 3 psi/sec<br>P56-PS2, Bias; Ramp ± 8 psid/sec | Detected, Isolated, and Accommodated |
| Actuated Sensors | VSV Bias; Ramp + 10%-st/sec<br>WFM Bias; Ramp -5%-st/sec<br>A8 Bias; Ramp +5%-st/sec | |
| Actuator | VSV Bias; Stuck Actuator, $NL_{Rel}$ 105 - 95<br>WFM Bias; Stuck Actuator, $NL_{Rel}$ 105 - 95<br>Uncommanded A8/Open, + 10 ma TMC | |
| Environ-mental Sensors | T2, Bias; Ramp +2.5° R/sec | |
| | PS2 Bias; Ramp -.25° R/sec | Detected, Turnoff Component Update |

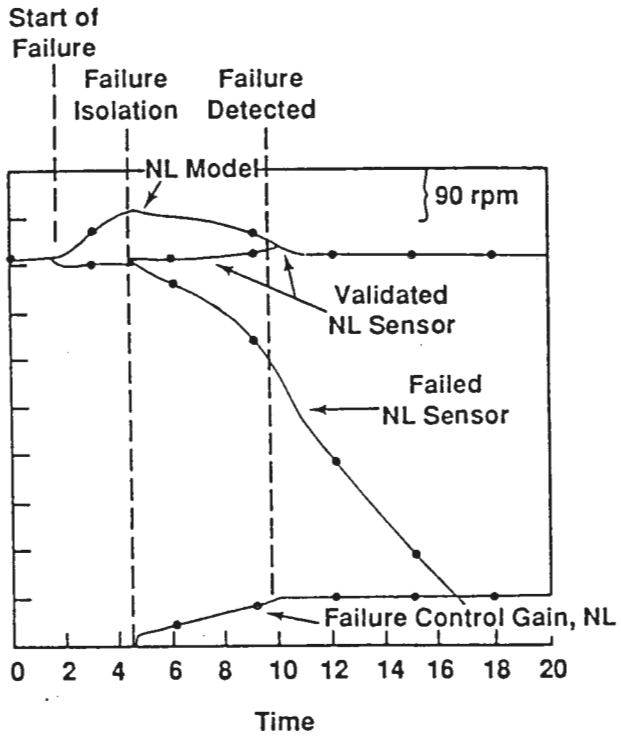



FIGURE 3.2-5. CORRECTIVE ACTION - FAN SPEED SENSOR

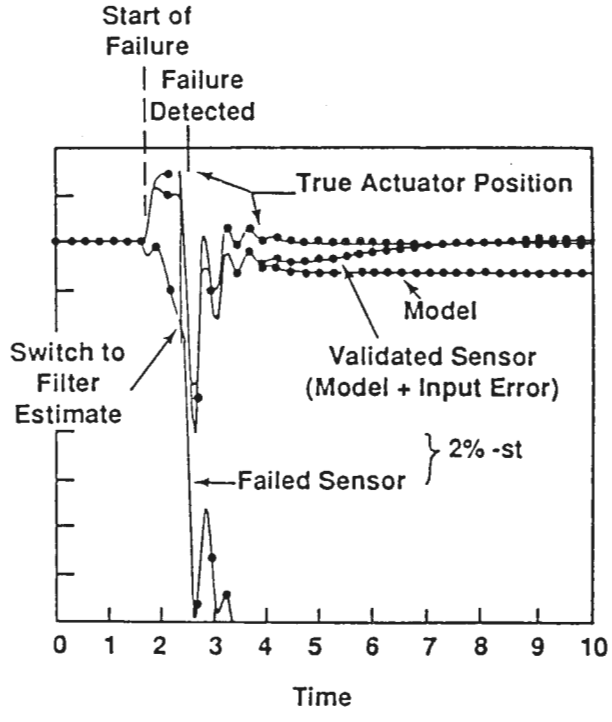



FIGURE 3.2-6. CORRECTIVE ACTION - FUEL FLOW ACTUATOR SENSOR

# 4.0 Primary and Secondary Electrical Power

## 4.1 Aircraft Power Systems

### 4.1.1 Power System Reliability Requirements

Electrical power system reliability directly affects the reliability of flight control and other flight critical avionic systems. The reliability of a single channel power system having a generator, generator controller, and load switches will not meet these reliability requirements. Hence, fault tolerance, including detection, isolation, and recovery, is an integral part of an aircraft power system. Material for this section is drawn from Mehdi and Leong 1985, Bret 1987, and Perkins and Marek 1977.

Requirements for reliability and redundancy, developed for the flight control and engine control systems, must be maintained for power generation and distribution. In a conventional power system, electrical generators driven by the aircraft engines provide the primary electrical power. A single generator on each engine can provide enough electrical power. A modern aircraft having computer equipment uses multiple generators and a backup auxiliary power unit (APU) to supply no-break power (at least no breaks longer than a few milliseconds). Flight critical systems operate from batteries charged from the generators.

Multiple generators may be needed on each engine to provide the reliability necessary for fly-by-wire flight controls, although a single generator could supply the electrical power. The F-16, for example, has a quadruplex redundant flight control system and a single jet engine. This aircraft also has multiple generators and batteries to provide an uninterruptible source of power.

Computer components are usually more susceptible to transients and noise on power lines than older technology. Added filtering is needed either in each component or in the power generation and distribution system.

Battery systems are needed on board the aircraft for operating critical systems in the event of an engine-out condition. Flight critical equipment must have a reliable source of electrical power to maintain safe flight and landing capability under extremely unlikely failure conditions.

Fast acting contactors and microprocessor generator controllers are needed for several reasons. Besides being quickly handled, faults in the electrical system must be handled so not to disturb power quality. Power transfer from one bus to another, such as from ground power to aircraft power, must not interrupt the supply. The normal contactors operate in a period of about 25 milliseconds. This delays the shut-down and turn-on time to about 50 milliseconds (DO-160 or MIL-STD-704). (Refer to Bret 1987 for a discussion of no-break power transfer.)

There is a trend in military power systems towards using 270 volt dc power for actuation systems. This places added requirements on power generation and distribution systems currently designed for 28 volt dc and 115 volt ac.

### 4.1.2 Electrical Power System Elements

Electrical power generation for a multiengine aircraft uses generators with constant speed drives mounted on the main engines. These generators can be isolated or paralleled for operation on the aircraft. Multiengine aircraft normally have parallel systems with three or more generators. Electrical systems for two-engine aircraft are usually operated in isolation. There usually are provisions, however, for tying the buses together if an engine or generator faults.

Failures can occur in the generator channel, distribution bus, avionic equipment, and wiring between these items. Protective devices and contactors segment the system for isolating faulted components. These contactors also supply power to the operating equipment.

Information measured from parts of the system provides fault identification and reconfiguration by control of these contactors. The electrical system is instrumented for measuring voltage and current at many locations. The instrument signals are fed to contactors in the power system control units including the generator control unit, bus control unit, and distribution center. Each group of contactors and instrumentation independently controls a portion of the system.

Because of independent control of the power system zones, faults in the system may allow an operating generator to reconnect or stay in parallel with a channel having an uncleared fault. This situation can be complicated if more than one failure happens or if a good generator is shutdown in error.

Fault tolerant power generation and distribution systems use fault detection and isolation at the local level. This tolerance may be accomplished by combining several techniques in the power system that include the following:

- Generator redundancy
- Power bus redundancy
- Functional load redundancy
- Power routing using bus interties
- Load shedding on prioritized basis

An aircraft power system may be configured in several ways to achieve high reliability. Power system redundancy may be accomplished by using several isolated channels or by reconfiguration using contactors.

An isolated channel approach, shown in figure 4.1-1, provides maximum independence between channels; it does not allow reconfiguration of power system elements for fault recovery. This approach provides isolation for power faults.

However, each single channel has only one power source, contactor, bus, and power control unit. The operation of the power controllers is discussed in section 4.2.1. Since the failure of any element leads to power failure, system reliability suffers. The reliability of a channel is the product of the element reliabilities. This isolated channel approach will meet flight control system reliability goals with the reliability of today's power systems.

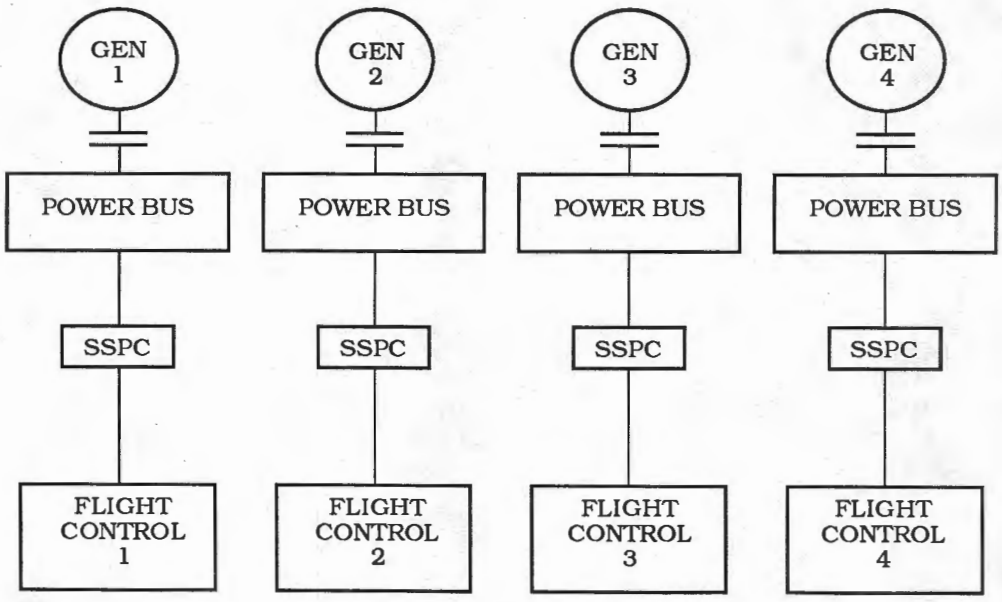



FIGURE 4.1-1. ISOLATED CHANNEL POWER SYSTEM

A reconfigurable approach, shown in figure 4.1-2, uses cross bus interties to provide power source redundancy. This approach does not have a "brick wall" isolation for flight control channels. This redundancy approach needs careful design and testing of the fault detection and fault isolation contactors. This example shows a three-generator power system supplying a four-channel flight control system. The use of two main power buses and contactors allows any generator to power any flight control system.

The reliability of the reconfigurable system is higher than the isolated channel system. Although in this system, correcting the effects of power system faults needs reconfiguration of working elements for maintaining power. The reliability of these contactors and their controls must be higher than those used for the system in figure 4.1-1. If the contactor controls fail, then the system will not operate properly. Present aircraft use manual contactors as a backup, if not the primary, for system configuration. Some systems use automatic load shedding (disconnecting non-critical loads) to keep critical equipment working.

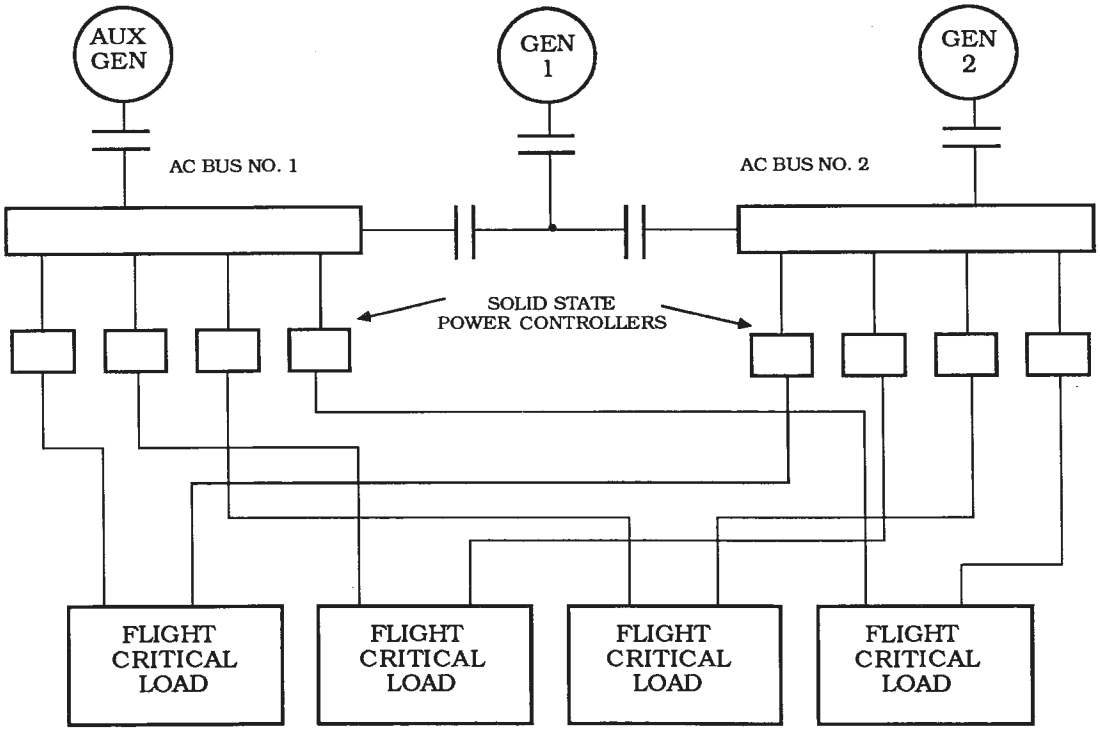



FIGURE 4.1-2. RECONFIGURABLE POWER SYSTEM

The most serious faults in any system are those affecting all of the channels at the same time. These common mode failures are often the hardest to protect against. The two-bus system, shown in figure 4.1-2, has two contactors between the main buses that give some common mode protection against a simultaneous bus fault and single contactor failure. Such a failure would prevent the isolation of the main generators. Careful system analysis and design can assure that such unlikely events have a lower probability of happening than any other event causing loss of flight control.

Power system design documents should provide fault trees and FMECA that include common mode effects and the normal component failure rate analysis.

### 4.1.3 Technology Trends

Advanced technology avionics have increased electric power demand and need power sources having improved reliability and power quality. The increased capability provided by application of electronic equipment to aircraft results in more equipment onboard, although the power demands of single systems have not increased. Large scale integrated semiconductor circuits are more susceptible to electrical transients and measures must be taken to protect them. A properly designed avionic system can operate in a very noisy environment with present aircraft power quality. The trade-off between protecting each unit of equipment

versus improving the power system source often comes out in favor of improving the source. This is particularly true of large commercial aircraft.

Uninterruptible power supplies (UPS) are needed for operating electronic equipment in the event of an interruption, or total loss, of electrical power generation. Flight critical systems need a back-up electrical power supply if all the engines or generators fail on a two-engine aircraft. Uninterruptible power can be supplied from batteries that fill in the gaps made by switching generators and by loss of generation.

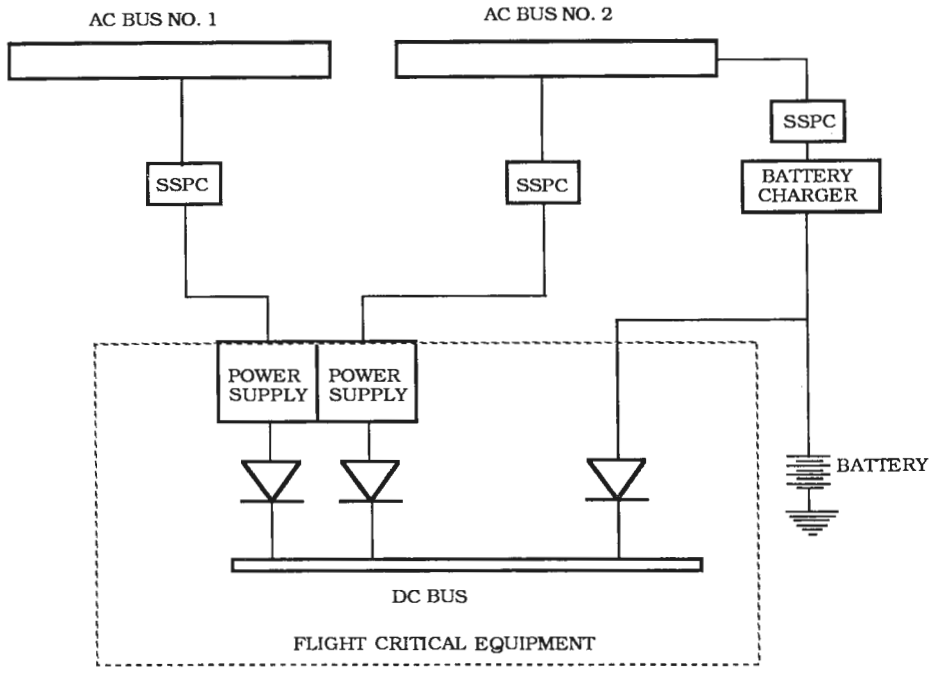
Two basic types of electrical power are available onboard an aircraft: ac and dc. Ac power is used for high energy electrical devices such as motors for landing gear and hydraulic pumps. Dc power could be used by electronic equipment, although present equipment is often configured for ac power. Some equipment operates from 28 volt dc and generates other dc levels internally. There is a trade-off between supplying power that the equipment most easily uses or putting power conversion equipment inside each component. For large aircraft the trade-off is on the side of supplying appropriate power to the equipment and minimizing power conversion in the equipment.

A current military aircraft system design trend to 270 volt dc power generation and distribution has several advantages over ac power. The major one being the elimination of the complex control and synchronization needed to parallel ac generators and to maintain frequency control with engine driven generators. An important consideration is supplying a UPS for computer equipment.

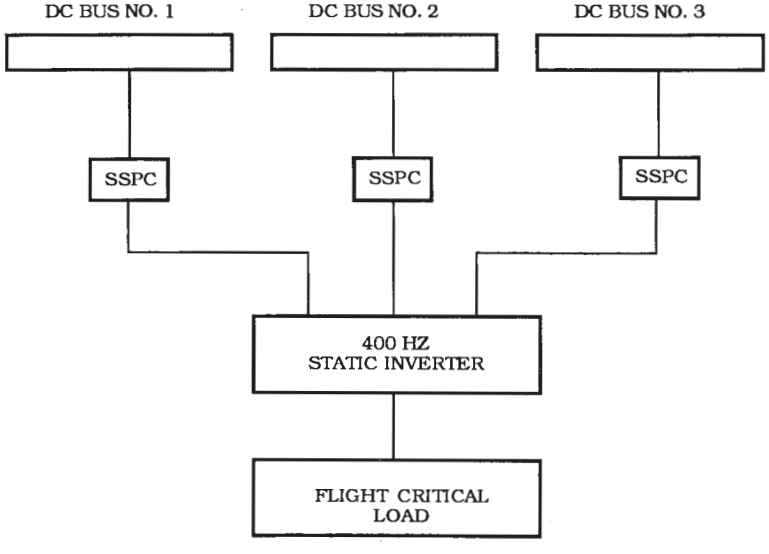
One method of providing uninterruptible power is by use of batteries supplied by rectifiers from ac generators. When a battery is used for primary power, other sources of power may easily be paralleled with diodes, as shown in figure 4.1-3. The battery is fully charged under normal conditions by a charger on the main power supply. Upon failure of the transformer rectifier unit (TRU), the battery will handle the load. This is called a battery float UPS. Batteries can handle the complete flight critical part of avionics and flight control for a period of thirty minutes to an hour.

Lead acid batteries are preferable to nickel cadmium because of the long time on float. Lead acid batteries may be fully charged with a 28-volt supply. But, nickel-cadmium batteries designed for 28 volts need a charger operating at about 36 volts. When discharging, the nickel-cadmium battery voltage drops quickly to 28 volts and stays there over the entire discharge time. Under normal float operation, the nickel-cadmium battery powered equipment must handle 36 volts. TRUs are typically unregulated and the higher voltage will pass to ac equipment as well.

Another UPS method is suppling dc directly from dc generators and using inverters to provide ac power where needed. Multiple sources of power can supply a flight critical bus. Figure 4.1-3 illustrates such a configuration for a TRU supplying a flight critical load.




A. BATTERY FLOAT UNINTERRUPTIBLE AC POWER




B. STATIC INVERTER VERSION OF UNINTERRUPTIBLE A C POWER

FIGURE 4.1-3. INTERRUPTIBLE POWER SUPPLY METHODS

## 4.2 Power Systems Detailed Examples

### 4.2.1 Fault Protection

The complexity of power generation and distribution systems for attaining high reliability needs automated fault protection. The following concepts provide isolation and protection against power system faults:

- Differential current protection
- Solid-state power controllers (SSPCs)
- Electromechanical power controllers
- Thermal circuit breakers

Generator feeders and main buses use differential current protection. This type of protection uses balanced transformers and voltage detection. The system provides fast and accurate protection levels. Operation is by a difference in current between two ends of a power feed, one being the load end and the other the bus end. Thus, any fault in the feed will be detected as a difference in the measured currents. Protection zones can be positioned to detect and isolate faults.

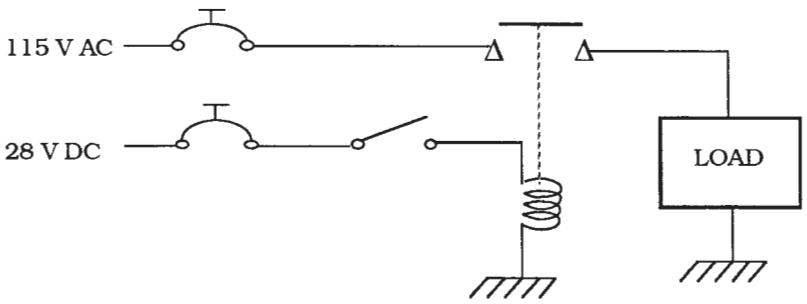
SSPCs replace the older thermal circuit breakers. SSPCs provide the same over current protection of the older units and provide a fast acting on-off switching control of electrical loads. Figure 4.2-1 shows an application where an SSPC replaces two circuit breakers and a relay for load switching. Fast fault clearing times can be achieved by using field effect transistors as the power switch. Test results from Mehdi and Leong 1985 indicate the fault isolation performance for five-amp 115 volt ac controllers with field effect transistors. When switched onto a 2,400 amp fault, the maximum current let through by these contactors was only 82 amps; the devices switched in about 10 microseconds.

Electromechanical power controllers or remote controlled circuit breakers perform the same function as SSPCs. Because they are mechanical devices, the switching speeds are slower and the devices are larger. These devices are suitable for high currents, as with larger motors.

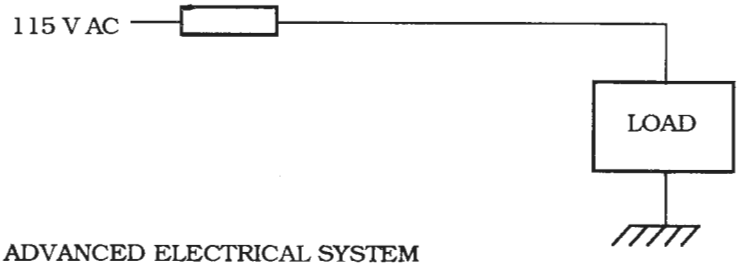
### 4.2.2 High Voltage dc Generation and Distribution

This section provides an overview of the technology for high voltage dc generation and distribution. This technology was developed under a Navy program (Perkins and Marek 1977) and defined an Advanced Aircraft Electrical System (AAES).

Key features of this technology are high voltage dc generation, microcomputer generator controls, and SSPCs. These components were designed and built to provide electrical power for an A-7E fighter aircraft. A prototype system, built and tested for a single channel, supplied the system parameters. Trade studies were conducted on the benefits of applying this technology to an entire aircraft.




A. CONVENTIONAL




B. ADVANCED ELECTRICAL SYSTEM

FIGURE 4.2-1. SOLID STATE POWER CONTROLLER SWITCHING

The AAES performed better than the conventional system in several areas. The areas of improvement included the following:

- Full system built-in test (BIT) showed failures down to the avionics unit and detected faults related to wire, connector pin, and terminations.
- The prototype included load management and power conditioning in the AAES although these functions are not feasible in the conventional system.
- The AAES design resulted in improved reliability by use of redundancy and distributed load management; neither is possible in the conventional system.
- Size, weight, and maintainability were all improved over the conventional system.

A trade study conducted to compare the High Voltage dc (HVDC) system with the conventional electrical system further illustrates the advantages of this system. An HVDC power system was designed for an A7-E aircraft using the prototype HVDC equipment parameters. The prototype AAES needed power conversion from 270 volt dc to standard 28 volt dc and 115 volt ac power. Since the penalty of distributing the three power levels biased the prototype parameters, the trade study included a "True HVDC" and "Projected HVDC" design. The AAES power system block diagram is shown in figure 4.2-2. Also shown are the military designations

for the hardware in the design from Perkins and Marek 1977. This prototype design uses samarium cobalt permanent magnet type generators for the 270 volt dc main and emergency generators.

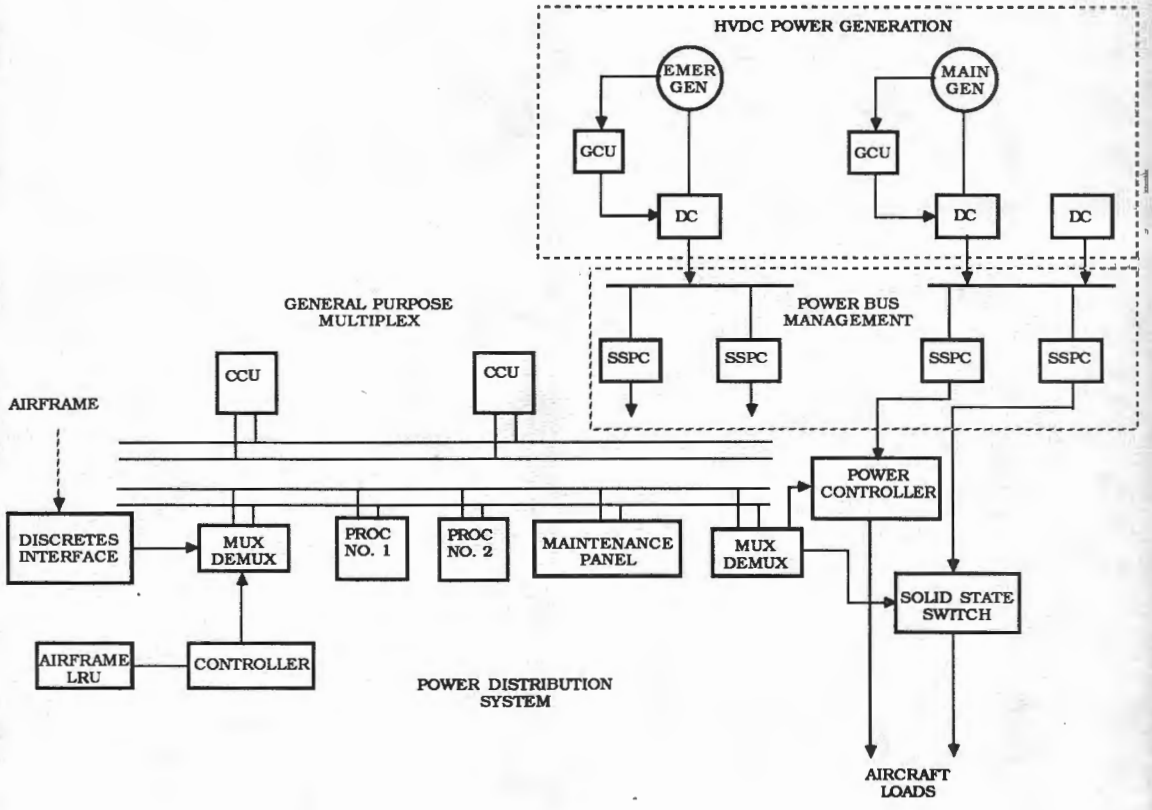



FIGURE 4.2-2. ADVANCED AIRCRAFT ELECTRICAL SYSTEM DIAGRAM

Comparative data from the trade study are presented in table 4.2-1; the "Projected" column shows the projection of the trade study to a new avionics suite that does not need 28 volt dc and 115 volt ac. If the avionics equipment were powered directly from 270 volt dc, then the 28 volt dc and 115 volt ac power conditioning and distribution could be minimized. Thus the "Projected" data is important for new aircraft.

Several benefits for an HVDC power system are given in Perkins and Marek 1977.

To the Equipment Manufacturer:
- Reduced voltage transients
- Uninterruptible power
- Smaller voltage variations
- Improved protection
- Smaller size and weight

TABLE 4.2-1. HVDC VS CONVENTIONAL POWER

| PARAMETER | CONV | PROTOTYPE AAES | PROJECTED HVDC |
|---|---|---|---|
| Weight (Pounds) | 190 | 76 | 64 |
| Power Generation<br>Power Bus Management<br>SOSTEL Power Distribution | 9 | 3 | 3 |
| Equipment | 177 | 223 | 163 |
| Wiring | 312 | 121 | 109 |
| Total | 688 | 423 | 339 |
| Efficiency (%) | 61 | 77 | 80 |
| Reliability - MTBF (Hours) | | | |
| Essential Loads | 63 | 509 | 625 |
| All Loads | 170 | 1442 | 1966 |
| Maintainability (MMH/FH) | 0.498 | 0.245 | 0.228 |
| Cost | | | |
| Nonrecurring | Base | 1.3B (-330) | 0.5B (50) |
| Life Cycle | Base | 0.3B (70) | 0.2B (80) |

NOTE: Numbers in parentheses show percentage of improvement over conventional power.

To the Airframe Manufacturer:
- Lower system weight
- Improved power quality
- Simplified system design
- Ease of parallel operation
- Higher overall efficiency
- Lower personnel hazard
- Easier change incorporation
- Elimination of constant speed drives

To the Engine Manufacturer:
- Lower overhung moment
- Lower weight

High- versus low-speed performance
Smaller size
Higher efficiency

To the Owner:

Lower life cycle cost
Improved reliability
Improved maintainability with BIT
Higher system efficiency
Lower personnel hazard
Lower weight

These benefits and economic factors in favor of HVDC power systems assure that further applications are expected in future aircraft.

# BIBLIOGRAPHY


Abrams, C. R. and S. T. Donley, "Flight Test of a Helicopter Fly-By-Wire/Light Actuation Control System", Proceedings of the National Aerospace and Electronics Conference, Dayton, OH, May 1984.

AiResearch, Electromechanical Actuation Feasibility Study, AD-A031 146, US Department of Commerce, May 1976.

Baker, W. and W. Bonnice, Intelligent Fault Diagnosis and Failure Management of Flight Control Actuation System, NASA Contract NAS2-12414, NASA Contract Report #177481, May 1988.

Barclay, B. A., T. G. Lenox, and C. J. Bosco, "Full Authority Digital Electronic Control-Highlights of the Next Generation Propulsion Control Technology", Gas Turbine Conference and Products Show, London, April 9-13, 1978.

Biafore, L. P. and L. P. Grieszmer, "Navy Advanced Flight Control Actuation System - Analysis and Design for a Slimine Building Block Actuator Concept", Technical Report NADC-82039-60, Naval Air Development Center, Warminster, PA, November 1983.

Breit, J., "Aircraft No-Break Electrical Power Transfer", IEEE Transactions, May 1987.

Brown, H., and J. A. Swan, "Analytical Redundancy Design for Improved Engine Control Reliability", General Electric Co., Aircraft Engine Business Group, Cincinnati, Ohio, NAPC-PE-171C, October 1987.

Brown, H. and R. W. Vizzini, "Analytical Redundancy Technology for Engine Reliability Improvement", Aerospace Technology Conference and Exposition, Long Beach, CA, October 13-16, 1986.

Chandler, P. R. and D. P. Rubertus, "A System Approach to Flight Control Reliability and Maintainability", AIAA Paper 84-2463, AIAA/AHS/ASEE Aircraft Design Systems and Operations Meeting, San Diego, CA, October 31 - November 2, 1984.

Chenoweth, C. C. and D. B. Slaugh, "Microprocessor Controlled and Managed Fly-By-Wire Hydraulic Actuator", Proceedings of the National Aerospace and Electronics Conference, Dayton, OH, May 1985.

Clements, R. R., "Reliability of Multichannel Systems", SIAM Review, Vol. 22, No. 1, January 1980.

Detroit Allison, Full-Authority Fault-Tolerant Electronic Engine Control System for Variable Cycle Engines, AFWAL-TR-82-2037, Air Force Wright Aeronautical Laboratories, June 1982.

Dotson, K. J., "Examples of Nonconservatism in the CARE III Program", NASA Technical Memo 100524, NASA Langley Research Center, January 1988.

Dotson, K. J., Analysis and Testing of the SURE Program, NASA Technical Paper 2817, NASA Langley Research Center, August 1988.

Feo, Pio and K. C. Shih, "Requirement Analysis of an Intelligent, Redundant Actuation System", IEEE Transactions, August 1986.

Gai, E., J. V. Harison, and R. H. Luppold, "Reliability Analysis of a Dual Redundant Engine Controller", IEEE Transactions on Reliability, Vol. R-32, No. 1, April 1983.

Hair, K. A., "Electromechanical Actuation Reliability and Survivability", IEEE Transactions, August 1985.

Hamilton Standard, Reliability Advancement for Electronic Engine Controllers, Technical Report AFWAL-TR-80-2063, Air Force Wright Aeronautical Laboratory, August 1980.

Harschburger, H. E., "Development of Redundant Flight Control Actuation Systems for the F/A-18 Strike Fighter", Aerospace Fluid Power and Control Systems SP-554, Society of Automotive Engineers, Inc., Warrendale, PA, October 1983.

Jenney, G. D., "Research and Development of Aircraft Control Actuation Systems", Technical Report AFFDL-TR-77-91, Air Force Flight Dynamics Laboratory, Wright-Patterson Air Force Base, Ohio, September 1977.

Jenney, G. D., et al., "Advanced Actuation Systems Development", Report WRDC-TR-89-3076 Vol. I, Dynamic Controls, Wright Research and Development Center, August 1989.

Jenney, G. D., et al., "Advanced Actuation Systems Development", Report WRDC-TR-89-3076 Vol. II, Dynamic Controls, Wright Research and Development Center, August 1989.

Jorgensen, D. E. and C. M. Hernandez, "Advanced Avionics System for an Aerospace Plane", IEEE Transactions, 1983.

Kenmir, C. V., et al., Non-Electronic Aspects of Avionic Systems Reliability, Bowty Boulton Paul Limited, England.

Kramer, M., "Development and Classification of Expert Systems for Chemical Process Fault Diagnosis", submitted to the International Conference on the Manufacturing Science and Technology of the Future, Cambridge, MA, June 1987.

Lala, J. H., et al., A Fault and Damage Tolerant Network For an Advanced Transport Aircraft, TP10-5:00.

Leonard, J.B., "A System Look at Electromechanical Actuation For Primary Flight Control", IEEE Transactions, August 1983.

McManus, B., "V-22 Tiltrotor Fly-By-Flight Control System", Eleventh European Rotorcraft Forum, City University, London, September 10-13, 1985.

Mehdi, I. S. and P. J. Leong, "Efficiently Meeting Electric Power Needs for Future Aircraft", Society of Automotive Engineers, Paper No. 859356, 1985

Murphy, M. R., "4-Valve Fly-By-Wire/Optical Control System", Technical Report NADC-85017-60, Naval Air Development Center, Warminster, PA, October 1984.

Murphy, M. R. and D. E. Haskins, "A Parallel Fly-By-Wire Helicopter Control System", Technical Report NADC-80132-60, Naval Air Development Center, Warminster, PA, March 1982.

Newirth, D. M., and C. J. Bosco, "Flight Test Reliability Demonstration of Electronic Engine Controllers", Aerospace Congress and Exposition, Los Angeles, Oct. 13-16, 1980.

Perkins, J. R. and A. J. Marek, Overview of the Advanced Aircraft Electrical System (AAES), A-7E Prototype Design, Naecon '77 Record.

Seeman, R., "An Electronic to Electrohydraulic Actuator Interface Concept for Redundant Flight Control System", Aerospace Fluid Power and Control Technologies, Society of Automotive Engineers, Las Vegas, NV, October 4-8, 1976.

Shannon, C. E., and E. F. Moore, "Reliable Circuits Using Less Reliable Relays", J. Franklin Inst., Vol. 262, pp 191-208 and 281-297, September/October 1956.

Smith, T. B., et al., "A Fault Tolerant Multiprocessor Architecture for Aircraft", CSDL Report R-1140, Vol. I, July 1976.

Swan, J. A. and R. W. Vizzini, Analytical Redundancy Design for Improved Engine Control Reliability, Final Review, AIAA-88-3176, Joint Propulsion Conference, July 11-13, 1988.

Tyron, J. G., "Quadded Logic", Redundancy Techniques for Computing Systems, Spartan Books, 1962.

Vizzini, R. W. and P. D. Toot, "Full Authority Digital Electronic Control Application to a Variable-Cycle Engine", SAE 1980 Aerospace Congress, Los Angeles, California, October 13-15, 1980.

Von Neumann, J., "Probabilistic Logics and the Synthesis of Reliable Organisms From Unreliable Components", Automata Studies, Annals of Mathematics, No. 34, pp 43-98, Princeton, 1956.

Waffner, W. D. and C. C. Chenoweth, "New Generation Flight Control Systems Hydraulic Actuation", presented at the 96th Meeting of the SAE A-6 Aerospace Fluid Power and Control Technologies Committee, Tampa, FL, May 1984.

Weinstein, W., et al., "Control Reconfigurable Combat Aircraft Development Phase I - R&D Design Evaluation", Technical Report AFWAL-TR-87-3011, Flight Dynamics Laboratory, Wright-Patterson Air Force Base, Ohio, May 1987.

Wulf, W. A., "Reliable Hardware/Software Architecture", IEEE Transactions on Software Engineering, June 1975.

## ACRONYMS AND ABBREVIATIONS

| | |
|---|---|
| AAES | Advanced Aircraft Electrical System |
| ac | Alternating Current |
| AIAA | American Institute of Aeronautics and Astronautics |
| APU | Auxiliary Power Unit |
| ARTERI | Analytical Redundancy Technology for Engine Reliability Improvement |
| ASEE | American Society of Electrical Engineers |
| BIT | Built-In Test |
| CSDL | Charles Stark Draper Laboratory |
| dc | Direct Current |
| DISAC | Digital Integrated Servo Actuator Controller |
| EMA | Electromechanical Actuator |
| EMAS | Electromechanical Actuator System |
| EMI | Electromagnetic Interference |
| EPR | Engine Pressure Ratio |
| FADEC | Full-Authority Digital Engine Controller |
| FBL | Fly-By-Light |
| FCC | Flight Control Computer |
| FDFM | Fault Detection and Failure Management |
| FICA | Failure Indication and Corrective Action |
| FMECA | Failure Modes and Effects Criticality Analysis |
| HIRF | High-Intensity Radiated Fields |
| HVDC | High Voltage dc |
| Hz | Hertz |
| I/O | Input/Output |
| LVDT | Linear Variable Differential Transformers |
| $N_1$ | Low Rotor Speed |
| $N_2$ | High Rotor Speed |
| NASA | National Aeronautics and Space Administration |
| PBW | Power-By-Wire |
| PLA | Power Level Actuator |
| psi | pounds per square inch |
| PZ | Piezoelectric |
| RL | Resistance/Inductance |
| SIAM | Society for Industrial and Applied Mathematics |
| SSPC | Solid-State Power Controller |
| TRU | Transformer Rectifier Unit |
| UART | Universal Asynchronous Receiver Transmitter |
| UPS | Uninterruptible Power Supplies |
| VSV | Variable Stator Vane |
| W/P | Fuel Flow to Burner Pressure |
| WFM | Main fuel metering valve actuator sensor |

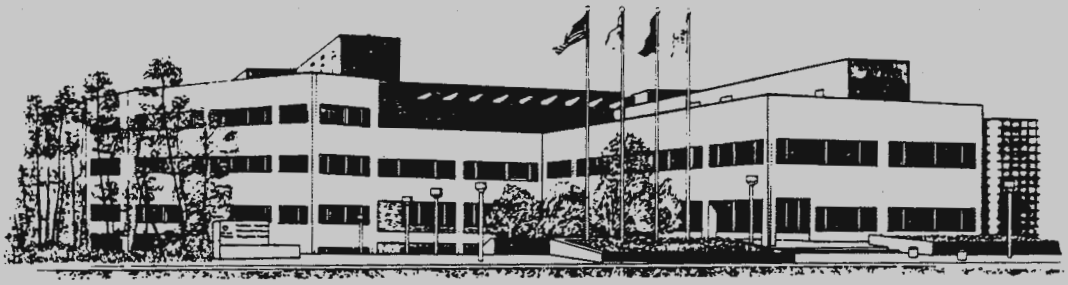
*U.S. GOVERNMENT PRINTING OFFICE: 1990-504-061/20155

U.S. Department of Transportation
Federal Aviation Administration

DOT/FAA/CT-88/10

# HANDBOOK-VOLUME II
# DIGITAL SYSTEMS VALIDATION

## CHAPTER 16
## ADVANCED VALIDATION ISSUES



PREPARED BY:

COMPUTER RESOURCE MANAGEMENT, INC.
950 HERNDON PARKWAY, SUITE 360
HERNDON, VIRGINIA 22070

PREPARED FOR:

FEDERAL AVIATION ADMINISTRATION
TECHNICAL CENTER
ATLANTIC CITY INTERNATIONAL AIRPORT, NEW JERSEY 08405

NOTICE

This document is disseminated under the sponsorship of the U.S. Department of Transportation in the interest of information exchange. The United States Government assumes no liability for the contents or use thereof.

The United States Government does not endorse products or manufacturers. Trade or manufacturers' names appear herein solely because they are considered essential to the objective of this report.

TABLE OF CONTENTS


| Section | | Page |
|---|---|---|
| 1. | INTRODUCTION | 16-1 |
| 2. | THE DIRECTION OF AMERICA'S AVIATION RESEARCH | 16-3 |
| 2.1. | Justification | 16-3 |
| 2.2. | The Goals | 16-3 |
| 2.2.1. | The Subsonics Goal | 16-3 |
| 2.2.2. | The Supersonic Goal | 16-5 |
| 2.2.3. | The Transatmospherics Goal | 16-6 |
| 2.3. | The High-Speed Rotorcraft and Hypersonic Aircraft | 16-7 |
| 2.3.1. | Rotorcraft | 16-7 |
| 2.3.2. | Hypersonic Aircraft | 16-8 |
| 2.4. | Advances in Avionics-Related Technologies | 16-8 |
| 2.4.1. | Overview of Digital Avionics Technology Advances - Artificial Intelligence | 16-9 |
| 2.4.2. | Overview of Digital Avionics Technology Advances - Standardization and Integration | 16-10 |
| 2.4.3. | Overview of Digital Avionics Technology Advances - Others | 16-11 |
| 2.4.4. | An Application of Microelectronics - The "Smart" Engine | 16-12 |
| 2.4.5. | An Application of Heat Removal Technology - Microelectronics | 16-14 |
| 3. | CERTIFICATION AND ADVANCING TECHNOLOGY | 16-19 |
| 3.1. | Challenges for the Certification Process | 16-19 |
| 3.1.1. | The New Generation of Subsonic Aircraft | 16-19 |
| 3.1.2. | A New Application for Vertical Takeoff and Landing Aircraft | 16-20 |
| 3.1.3. | The Experimental Aircraft - X-29 | 16-20 |
| 3.1.4. | New Aircraft - Summary Comments | 16-22 |
| 3.2. | A Specific Problem for Certification - The Propfan | 16-22 |
| 3.3. | A Specific Problem for Certification - The Rotorcraft | 16-25 |
| 3.3.1. | The Full Authority Digital Engine Controller | 16-25 |
| 3.3.2. | Fly-By-Wire Technology | 16-26 |

TABLE OF CONTENTS


| Section | | Page |
|---|---|---|
| 3.4. | The General Problem for Certification | 16-28 |
| 3.4.1. | Intelligent Components, Subsystems, and Systems | 16-28 |
| 3.4.2. | System Integration | 16-33 |
| 4. | RELATED VALIDATION ISSUES | 16-43 |
| 4.1. | Fiber Optic Technology | 16-43 |
| 4.1.1. | Fly-By-Light | 16-43 |
| 4.1.2. | Fiber Optic Sensors | 16-45 |
| 4.2. | Artificial Intelligence | 16-48 |
| 4.2.1. | Voice Command Processing | 16-48 |
| 4.2.2. | How Machine Vision is Accomplished | 16-52 |
| 4.3. | Computer Technology | 16-54 |
| 4.3.1. | Advances in Microelectronic and Component Technologies | 16-54 |
| 4.3.2. | Advances in Parallel Processor Organization | 16-55 |
| 4.3.3. | Advances in Distributed Processor Organization | 16-57 |
| 4.4. | Advances in Control and Display Technologies | 16-59 |
| 4.4.1. | Improved Man-Machine Interface | 16-59 |
| 4.4.2. | Remotely Piloted Vehicles | 16-64 |
| BIBLIOGRAPHY | | 16-67 |
| GLOSSARY | | 16-69 |
| ACRONYMS | | 16-70 |

## LIST OF ILLUSTRATIONS


| Figure | | Page |
|---|---|---|
| 2.2-1 | NATIONAL AERONAUTICAL RESEARCH & DEVELOPMENT GOALS | 16-4 |
| 2.4-1 | OPERATION OF HEAT PIPE IN MICROCHIP COOLING | 16-16 |
| 3.4-1 | AIRCRAFT PERFORMANCE BY PROPULSION SYSTEM | 16-30 |
| 3.4-2 | EXPERT SYSTEM | 16-32 |
| 3.4-3 | SYSTEMS INTEGRATION | 16-35 |
| 3.4-4 | ADVANCED FUEL MANAGEMENT SYSTEM (TOTALLY REDUNDANT) | 16-38 |
| 3.4-5 | ADVANCED FUEL MANAGEMENT SYSTEM INTERFACES | 16-39 |
| 4.1-1 | FLY-BY-LIGHT SCHEMATIC | 16-44 |
| 4.1-2 | FIBER OPTICS IN WING ASSEMBLY | 16-46 |
| 4.2-1 | VOICE RECOGNITION SYSTEM | 16-51 |
| 4.3-1 | DISTRIBUTED PROCESSOR FLIGHT CONTROL SYSTEM | 16-58 |
| 4.4-1 | PILOT/VEHICLE INTERFACE WITH ARTIFICIAL INTELLIGENCE SYSTEM | 16-63 |
| 4.4-2 | REMOTELY PILOTED VEHICLE AVIONICS | 16-66 |


## LIST OF TABLES


| Table | | Page |
|---|---|---|
| 2.3-1 | AIRCRAFT TYPE BY SPEED RANGE | 16-7 |

# 1. INTRODUCTION

The Aeronautical Policy Review Committee was organized to formulate a plan for the advancement of United States aeronautical capabilities during the coming decades. In 1985 this committee issued a report that established three goals for America's aerospace community. The goals are comprehensive and span all areas of flight-subsonic, supersonic, and transatmospheric. The achievement of these goals will require major technological advances in all aeronautical technologies. Aerodynamic advances will include improved capabilities in computational fluid dynamics and unobtrusive instrumentation. Propulsion advances will include the development of a range of variable cycle engines and hypersonic ramjets. Structural technology advances will include the development of adaptive, thermal, damage tolerant, and propulsion structures. Material technology advances will result in new composites, alloys, and super-alloys being used throughout the aircraft. Flight systems will benefit from faster microelectronics, new processor architectures, and new software technologies. System integration will result in the interfacing of various aircraft systems under a single controlling device.

Two attributes are common to many of the advancing technologies: First, the embedding of intelligent devices into systems; and second, the aggregation of various avionic functions into higher order functions. In this tutorial these attributes will be referred to as Artificial Intelligence (AI) and System Integration, respectively. The intelligent device will virtually always be a digital microprocessor or network of processors, enriched with AI software. The direction of integration is to integrate the flight control function into the propulsion, structural, or aerodynamics system in order that a higher level system be created. The fundamental motives for these changes are decreases in operating cost, increases in service, and increases in safety.

The aforementioned attributes of advancing technology imply that digital avionics systems for coming generations of aircraft will be very different than today's systems. The certification of these new systems will require validation regulations and procedures that are different than those used in the past. This tutorial introduces the issues that will be the basis for the adaptations or changes required in certification procedures. Section 2 of this tutorial is oriented toward advancing technology. Paragraph 2.1 and 2.2 introduce the three national goals and the types of advances anticipated from their achievement. Paragraph 2.3 introduces the rotor and hypersonic aircraft. These are mentioned because they do not fall within the direct purview of the national goals, but will make significant contributions to commercial air service and/or technology. Paragraph 2.4 is an overview of the digital avionics advances anticipated in the future. Research and development efforts in AI and systems integration are highlighted. A generic "smart" engine and a heat removal technique for microchips are discussed as enabling technologies for future digital avionic systems.

Section 3 is oriented to the identification of those technological trends that will require adapting or changing some certification procedures. Paragraph 3.1 presents the Airbus 320, Boeing 7J7, Bell V-22 Osprey, and X-29 as examples of aircraft that may require immediate or near future certification. Paragraph 3.2 presents the propfan propulsion system as an example of an innovation that challenges current validation procedures; Paragraph 3.3 presents the Full Authority Digital Engine Controller (FADEC) and Fly-By-Wire (FBW) technologies in the same vein. Paragraph 3.4 discusses the problems of validating "intelligent" and multi-function systems.

Section 4 contains detailed discussions on a number of issues that are critical to certifying aircraft of the future. These issues include Fiber-Optics, AI, Advanced Controls/Displays, and Remotely Piloted Vehicles (RPV).

## 2. THE DIRECTION OF AMERICA'S AVIATION RESEARCH

### 2.1. Justification

In 1982 a report was issued by a governmental interagency group, under the direction of the White House Office of Science and Technology Policy, analyzing the state of aeronautical research and the role of the Federal Government in supporting that research. The conclusions of that report acknowledged the existence of the potential for advances in aeronautical technology that would result in monumental increases in aircraft performance. It was further recognized that the realization of these advances could only be accomplished by a coordinated effort involving both government and industry. This report resulted in the formation of the Aeronautical Policy Review Committee whose purpose was to track the implementation of the recommendations of that report. In 1985 this committee issued a report (National Aeronautical Research & Development Goals - Technology for America's Future, 1985) that established the specific goals that must be vigorously pursued if the advances are to be realized. This report also recognized that, in addition to the technological problems facing American government and industries, another factor must be considered: competition from abroad. In the interest of national economic and military security, United States aeronautics must meet these challenges and remain a decisive influence in world aviation.

### 2.2. The Goals

The Aeronautical Policy Review Committee established three national goals. When accomplished, these goals will be vital to America's preeminence as a world leader in aviation technology. Efforts on achieving the three goals will proceed in parallel, but with a time-phased order of attainment as depicted in figure 2.2-1 (Lupinetti, 1987). The goals are as follows:

- Subsonics goal: to revitalize aircraft technology.
- Supersonics goal: to attain long-distance efficiency.
- Transatmospherics goal: to secure future options.

The immediate emphasis will be on accomplishing the first goal before the end of the Twentieth Century. This will provide the impetus for continuing the effort and accomplishing latter goals after the year 2000.

#### 2.2.1. The Subsonics Goal

The subsonics goal is to move a new generation of subsonic aircraft to a competitive commercial position in the world transportation system. It is important because it is a foundation for latter goals. While United States aircraft still maintain a broad base of technological advantage, the margin

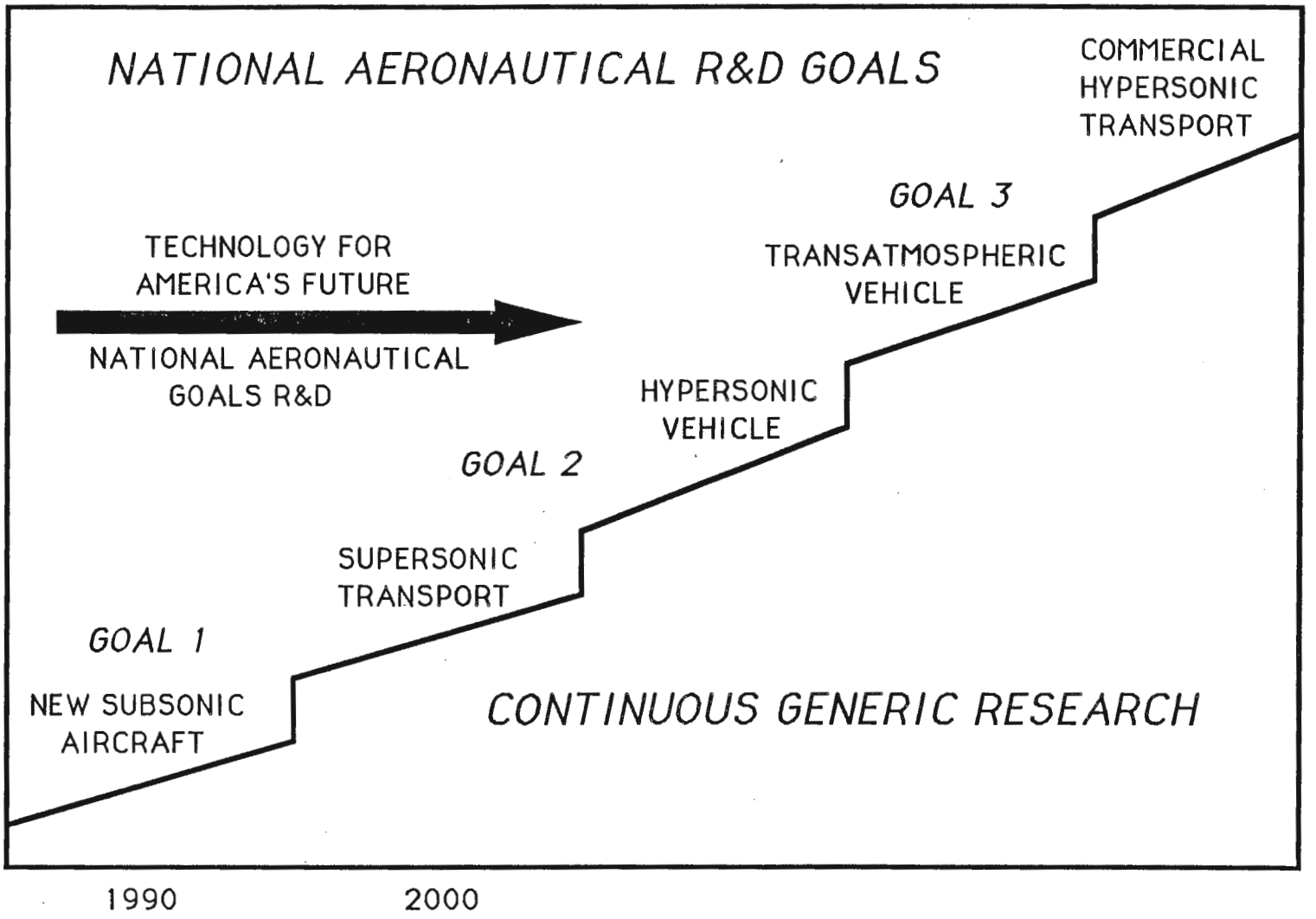



FIGURE 2.2-1. NATIONAL AERONAUTICAL RESEARCH & DEVELOPMENT GOALS
(Lupinetti, 1987)

of that advantage has been reduced significantly in recent years. In an increasing number of technological areas, foreign capabilities are now equivalent to, or superior to, that of America. During most of the last two decades, improvements in aeronautical technologies have been evolutionary. This will continue, but the new generation of subsonic aircraft must also foster revolutionary technologies. These two approaches to research and development, evolutionary and revolutionary, must produce the resources in the private sector that will enable the exploitation of the ensuing opportunities in supersonic and transatmosperic flight. This goal not only calls for the production of a safe, fuel-efficient, and technically superior aircraft but for a modernized national airspace system. This modernized airspace system is crucial if the benefits from the new aircraft are to be fully realized.

The objective is to have the new subsonic generation of aircraft readied by the mid-1990's. Consequently, the United States would enjoy advanced subsonic technology benefits before the 21st Century. Technologies that are essential to the development of the new subsonic aircraft are as follows:

- Laminar flow control advancements that substantially reduce air drag.
- A new generation of super bypass and propfan engines that will provide a more reliable and efficient operation.
- A new composite, high strain structure.
- More sophisticated, fully integrated flight controls and operating systems that will interface with the new national airspace system.

### 2.2.2. The Supersonic Goal

This goal becomes increasingly important because of changing transportation needs. Strategically and economically, the United States is becoming closely allied with the Pacific community of nations. This is in opposition to past development in the European community. The Pacific region's capacity for growth and development should be encouraged by America's enterprise and policy. A major constraining factor in the developing relationship between the United States and Asia is distance. Supersonic air travel will minimize the impact of distance. The development of technologies required for a new generation of supersonic transport aircraft will provide an opportunity for joint developmental efforts with the Pacific nations. Long-term political bonds may result from these cooperative efforts.

With the elimination of the United States Supersonic Transport effort in 1971, aggressive research and development was curtailed in the supersonic field. However, the Supersonic Cruise Research program, sponsored by NASA, maintained an American presence in this field through 1981; this work provides a basis for continuing the effort. Technological advances in the area of cooling methods, internal aerodynamics, coatings, and single crystal turbine blades offer a means for solving some of the problems present in the original effort. These advancements, when applied to a variable cycle engine with less noise and greater fuel efficiency, offer realistic hope for the development of a supersonic transport.

The report of the Aeronautical Policy Review Committee (National Aeronautical R&D Goals - Technology for America's Future, 1985) credits the following technological advances as being important to the supersonic effort:

- The application of powder metallurgy technology and superplastic forming techniques to load-carrying structures.

- The development of new thermoplastic, carbon-carbon, and metal matrix materials.

- The development of new fault-tolerant computers to assist in load alleviation and dynamic damping.

- The development of supersonic laminar flow that will result in significant reductions in the aircraft weight per pound of payload by providing sustained cruise speeds at much greater fuel efficiency.

- The development of computers with greater computational capabilities that will permit the computation of complex flows and optimizations extending the regions for supersonic laminar flow.

The integration of technological advances into a new supersonic transport aircraft would result in reduction of flight times to five hours or less between the most distant points in the Pacific area. For both commercial and military purposes, this is important to America's future.

### 2.2.3. The Transatmospherics Goal

The United States partitions flight into aeronautics and space. Some of the benefits obtained from subsonic flight can be enhanced in supersonic flight. Some of the benefits of supersonic flight can be enhanced in transatmosperic flight. For the United States to be a decisive influence in the world transportation system, it must have the following capabilities:

- Maneuvering craft into and out of the atmosphere.

- Rapidly responding to needs for low earth orbit missions.

- Attaining very rapid transport services from conventional runways between points on the earth's surface.

These capabilities will be required for America's security, both economically and militarily, in the coming century.

Much of the technology required for the convergence of the aeronautics and space flight concepts will be directly obtained from or will be extensions of the new subsonic and supersonic aircraft. The space shuttle experience will be a major source. In other cases, new technologies will be required. As scarce resources are spent in propulsion, configuration, materials, flight system, and fuel/coolant research, consideration must be given to the unique requirements of the transatmospheric goal.

## 2.3. The High-Speed Rotorcraft and Hypersonic Aircraft

While the three goals of the Aeronautical Policy Review Committee define the direction of aeronautical research and development, it is important to mention two types of aircraft that vaguely fit into these goals: the high-speed rotorcraft and hypersonic aircraft. Table 2.3-1 shows the accepted classes of aircraft by speed range.

TABLE 2.3-1. AIRCRAFT TYPE BY SPEED RANGE

| Type of Aircraft | Speed Range |
|---|---|
| Rotorcraft/Vertical Takeoff and Landing | Hover to Mach Range 0.8 |
| Subsonic | Up to Mach 1 |
| Supersonic | Mach Range 2 to 4 |
| Hypersonic | Mach Range 5 to 12 |
| Transatmospheric | Mach Range 8 to Orbital Velocity |

Strictly speaking, the rotorcraft may be considered subsonic, and the hypersonic may be considered transatmospheric. Because of their unique technologies, they are considered separately in this tutorial.

### 2.3.1. Rotorcraft

The short-haul commuter market is expected to grow extensively in the coming years. Vertical Takeoff and Landing (VTOL) craft, e.g., the tilt-rotorcraft should supply much of the capability for this market. These vehicles should permit commercial travel between center-cities at speeds in excess of present-day helicopters. The tilt-rotorcraft concept came into existence during World War II. When the prop-rotors are facing upward, the tilt-rotorcraft hovers like a helicopter; when they face forward, a turboprop airplane is created that can travel at speeds up to 300 miles per hour. The X-wing aircraft is another rotorcraft design that may service this same market. The X-wing is a stoppable rotor aircraft: when the rotor is active, the aircraft can hover like a helicopter; when the rotor is stopped, the aircraft becomes a conventional aircraft with cruise speeds in excess of 400 miles per hour.

The technology required for these two designs is at hand. Propulsion is by turboprop or turbo-jet; aluminum is the material used for body construction; and conventional hydrocarbon fuels/coolants are used. The major disadvantage is the greater cost per payload-mile, and this may require improvement. However, the convenience of beginning and ending flights at small, convenient heliports, rather than at large airports, offsets this cost to a degree. If this commercial market develops, the certification engineer will be required to create new or adapt existing rotocraft airworthiness rules.

### 2.3.2. Hypersonic Aircraft

This discussion is included for completeness. Hypersonic flight fills the gap between supersonic flight and transatmospheric flight. As noted in table 2.3-1, this spans Mach ranges 5 through 12. The potential commercial applications for transports of this class would be for transoceanic, transcontinental travel, the same as supersonic flight. The major difference is that a Los Angeles International (LAX) to Tokyo flight is estimated to require about 5 hours in a supersonic craft (Mach 2-3) or 2 hours in a hypersonic craft (Mach 5). Supersonic aircraft require conventional airports, while hypersonic aircraft would require extended runways at conventional airports.

Research in hypersonic flight started with the X-15 research program and is currently conducted by the Defense Advanced Research Projects Agency (DARPA). The current long-range cruise missile may be the first operational hypersonic vehicle. The problems of sustained hypersonic flight are significant propulsion and aerodynamic heating. The propulsion system must be able to sustain the Mach 5 and up speeds. The cruise missile uses the supersonic-combustion ramjet (scramjet) burning high-density hydrocarbon fuels. The materials required for hypersonic flight must be able to sustain temperatures of 5000 degrees Fahrenheit. Airframe research using supercold hydrogen is solving this problem. Civil aviation is unlikely to benefit from hypersonic flight or transatmospheric flight until well into the 21st Century.

## 2.4. Advances in Avionics-Related Technologies

Digital avionics is one of the fastest growing technologies in aerospace science. It continues to proliferate throughout aircraft and spacecraft, and finds application in subsystems, such as sensors, system controls, crew function automating equipment, displays, and communications. Research and development is currently underway for commercial FBW and Fly-By-Light (FBL) controls. The Airbus A320 is the first airliner in the world with computer-driven FBW controls which prevent the aircraft from stalling and exceeding speed/acceleration limits. While highly innovative, the Airbus Industrie plane appears already to be a commercial success: 439 booked orders and options from 15 airlines before its first scheduled flight. Certification and first entry service are planned for 1992.

### 2.4.1. Overview of Digital Avionics Technology Advances - Artificial Intelligence

Artificial Intelligence and expert systems are emerging in numerous digital avionics areas. Systems incorporating this technology are being designed to perform the following tasks:

- Coordinating sensor data to determine best flight path, thus avoiding terrain threats.
- Automatically initiating the execution of certain emergency procedures.
- Automatically reconfiguring aircraft systems to optimize performance or correct faults.

Artificial Intelligence and expert systems are software entities, and their emergence in digital avionics is accompanied by greater shipboard computer processing and data storage facilities.

The DARPA and United States Air Force (USAF) are cooperating in the Pilot's Associate program. While this program is directed toward combat pilots, the acceptance of this technology will most likely result in its adaptation for commercial flight. This program's emphasis is on providing the pilot with information and aids for assisting in immediate decision-making; a secondary emphasis is to assist the pilot in longer-range planning. The computer software required for this is a set of cooperating expert systems that operate at very high processing speeds; Lockheed-Georgia and McDonnell Douglas are developing this software. A new parallel-processor computer architecture is being developed to support these expert systems.

Other AI efforts involve the integration of various sensor data into a single composite display. Examples of the sensors being integrated are as follows: ground-mapping radar, stored digital map, and the location of nearby aircraft. AI is also an element in speech recognition. In this capacity, AI has advanced to the stage where a system can learn to recognize a pilots voice in two sessions of less then 60 minutes' duration. This system makes use of the latest ITT (consultant committees for) International Telegraphy and telephony algorithms.

As mentioned earlier, the incorporation of expert systems and AI in avionic systems requires a new approach to software systems development. This is due to the greater logical complexity and the size of the software. Numerous efforts are underway with the objective of streamlining software development and maintenance in this environment; the Air Force Modular Avionic System Architecture is an example. Expert systems and AI will be covered in detail in section 4 of this tutorial. They also apply to other technological improvements, such as Advanced Controls and Displays, Voice Command Processing, Systems Integration, and Robotics. These areas will also be discussed in later sections of this tutorial.

### 2.4.2. Overview of Digital Avionics Technology Advances - Standardization and Integration

In addition to AI, standardization and integration of avionic modules is being emphasized. At this time, the impetus for this effort has been felt primarily in military avionics. In 1986, Congress requested that the services "prepare a joint plan for the inclusion of fully integrated, digital avionics...on all aircraft under development." DOD's (Department of Defense) response to this was the development of the Joint Integrated Avionics Plan for New Aircraft. As a result of this plan, common line replaceable modules will use Very-High-Speed Integrated Circuits (VHSIC). The common line replaceable modules would then be used to build standardized data processors, integrated systems (communications, identification, navigation), and signal processors. Standards for architecture, high-speed data bus, avionics computer, packaging, connectors, backplances, and reusable software modules are in the process of being developed. The Navy Advanced Tactical Aircraft, the Air Force Advanced Tactical Fighter (ATF), and the Army Light Helicopter Experimental are among the initial craft destined for standardization. The effects of standardization and integration will be present in future commercial digital avionic systems.

As FBW and FBL controlled aircraft system designs gain greater attention, the development of digital Cathode Ray Tube (CRT) cockpit displays/controllers and integrated air/ground communication systems increase in importance. The V-22 Osprey, a tilt-rotorcraft with potential commercial usage, has its digital avionic systems highly integrated with other aircraft systems. All information to the pilot is provided through an operator's console consisting of four six-inch color displays. Such parameters as engine speed, engine torque, fuel quantity, fuel rate of flow, and electrical system configuration are displayed through this console. Alphanumeric keyboards with small CRT monitors are used to control the aircraft. The FBW Digital Flight Control System (DFCS) consists of three digital processors, each controlling a MIL-STD-1553 data bus. A digital system uses optical disk for the storage of maps which are used as a supplement to other navigational displays. The detection of faults and subsequent isolation of affected avionics subsystems is performed by digital software. The reliability of these digital systems is resulting in their growing acceptance into the area of flight-critical processing.

Commercial aviation is converging on an integration of air, ground, and satellite digital data links. To facilitate this, agreements are being reached on communications architecture and interface standards. With these agreements, different communications equipment will be capable of sharing data without modification to either equipment or software. An important benefit of this will be the ability to bypass faulted areas of the communications network by the process of rerouting. To some degree, the planned Federal Aviation Administration (FAA) Traffic Alert and Collision Avoidance System (TCAS) will require this technology. To a greater degree, the FAA's Automated Enroute Air Traffic Control System (AERA) will require it. AERA involves a complex modernization of all FAA on-the-ground air traffic computers. TCAS is scheduled to be operational in the near future; AERA is scheduled to begin implementation in the mid 1990's and continue implementation through the early years of the next century.

### 2.4.3. Overview of Digital Avionics Technology Advances - Others

In addition to the areas of AI, standardization, and integration, near-term changes are underway in other avionics-related technologies. Examples of these technological concerns and developments are as follows:

- In converting from analog to digital systems, concern is being directed to safety during the transition period. International activity is growing in the area of increasing Radio Frequency (RF) susceptibility standards and determining field strengths in the geographical areas adjacent to airports.

- Computer architectures are changing to permit greater processing speed and data handling capacity. AT&T has developed a family of signal processors that execute mathematical operations only when data is available, not at fixed time intervals. The advantages to architectures similar to this are rapid allocation of resources, high throughput, and parallel processing of multiple sets of data.

- Computer architectures based on the MIL-STD-1750A processor are being developed. These are being put on a single chip with VHSIC and Gallium Arsenide (GaAs) technologies. GaAs is also being used for cascadable arithmetic logic units, used in embedded computers requiring flexible word-lengths.

- Fiber optics technology is being developed to provide the bandwidth necessary for new digital data buses. Older MIL-STD-1553 buses are being replaced with fiber optic systems with compatible protocols and software. An experimental FBL system is being tested on a Sikorsky JUH-10-60A helicopter that utilizes an AT&T fiber optic bus; the bus appears to operate at 1 Million bytes per second (Mbps) to the user, but is actually operating at 20 Mbps.

- The Boeing 7J7, a next generation commercial aircraft, was tested with a voice recognition system. Twenty-five potential flight deck applications for voice recognition have been identified.

- Analog-oriented functions have been replaced with digital-oriented functions. This replacement has resulted in a reconsideration of the boundaries of hardware, firmware, and software. This has primarily been felt in the digital areas of process controllers with limited branches, single chips, and bit slices. Line replaceable units, earlier considered to be pure hardware, now have embedded processors which require software or firmware. DOD is attempting to standardize new software development around DOD-STD-2167 and modifications to MIL-STD-480.

- Signal processors are now being developed to process two and more functions.

- Reductions in weight and volume with accompanying increases in accuracy are being achieved with new global positioning systems.

- The "fly-by-glass" trend, the use of CRT's in place of traditional instrumentation has been under constant development. The use of CRT's places different workload patterns on the aircraft crew. Human engineering and user-friendly design factors are being approved before aircraft design is finalized.

2.4.4. An Application of Microelectronics - The "Smart" Engine

An example of the research underway that dramatically affects traditional technology is the application of microelectronics to propulsion systems. The research cited in this section was performed by A.H. Epstein, and is summarized in ""Smart" Engine Components: A Micro in Every Blade?" (Epstein, 1986).

Most of today's attention in propulsion technology is directed toward the debate over the resurrection of the old turboprop concept in the guise of the "propfan". However, other research is underway that may have major effects on the aircraft of the future. Aircraft gas turbine engines are virtually open-loop devices. They have remained so for essentially three reasons: First, relatively poor sensor technology and limited onboard computational capability have not justified a controller; second, high reliability has been achieved with simple systems; and third, high performance gains have been realized without the need for control loops. However, advances in microelectronics make possible engine control devices with major computational capacities; the result is that gas-turbine components and subsystems may be moved from open- to closed-loop operation. Mr. Epstein feels this may be desirable for four reasons:

- Current engine designs represent a constrained optimization. Improvements in the performance of components or subsystems, off-design, may not result in improved performance of the design: a high thrust-to-weight ratio conflicts with good performance retention and maintainability. A relaxation of these constraints could improve performance.

- Current design practice is expensive; self-adaptive components may allow less expensive development.

- Through dynamic, subsystem reconfiguration, adaptive components could increase damage tolerance and reliability.

- Current technology may not produce the efficient, multifunction propulsion systems that will be required for advanced aircraft.

In order that "smart" technology be incorporated in a propulsion system, a feedback control mechanism must be present that will permit a component's adaptation to local conditions. This feedback control mechanism will consist of sensors, processors, and actuators.

Applications of "smart" engines may be categorized as near-term versus long-term, low risk versus high risk. Examples of potential applications are as follows:

- Active control of turbine blade tip clearance, based on real-time measurements instead of open-loop scheduling. This is a near-term application

that has the development of highly reliable sensors as an enabling technology.

- Active exhaust-nozzle position improvement. The nozzle would be adjusted based on data from the aircraft accelerometers, thus improving vehicle performance. Today's continuously variable nozzles operate with pressure mismatches due to nonideal conditions, schedule problems, atmospheric gradients, and other factors. This is a near-term application which requires the development of variable-exhaust nozzles capable of lasting through many adjustment cycles as an enabling technology.

- Active inlet distortion control. High compressor stall margins (on the order of 10-20 percent) exist in high performance military aircraft because of inlet distortions at high angles of attack. This stall margin could be reduced if the compressor distortion tolerance could be dynamically adjusted. This adjustment could be based on local pressure distribution measurements that result in the activation of individual guide vanes. Enabling technologies consist of actuators and sensors of sufficient bandwidth and reliability, and greater understanding of compressor behavior. This is a longer-term application.

- Active noise abatement is one of many additional potential applications.

Common to virtually every potential application are the requirements for significant technological improvement in actuators, sensors, processors, data transmission, and control system architecture. In order that the requirements for withstanding large forces and for small physical size be met, actuators have traditionally used hydraulics. However, their complexity and maintenance characteristics make it unlikely that large numbers of these devices could be used throughout an engine. Large numbers would be required if the engine were closed-loop in design. Electric actuators are desirable if they can be developed to meet the torque, packaging, and bandwidth requirements. Samarium-cobalt motors and piezoelectric devices offer some promise in meeting these requirements, but each has its shortcomings. (Layers of piezoelectric material, dispersed throughout a structure, have been proposed as actuators capable of supplying distributed forces of relatively high frequencies.)

For the purpose of increasing reliability, a designer may provide additional computing elements in the system, rather than add sensors to the engine. Using a single chip to incorporate signal conditioning, transducer, data reduction, and control may provide the needed signal processing while not affecting reliability significantly. Fiber optics may also offer hope for improvement.

Microelectronics, while the catalyst for developing closed-loop engines, may also be a problem. Control elements dispersed throughout a gas turbine engine will require cooling. Current semiconductor technology is restricted to a modest temperature range. Development of high-temperature semiconductor technology is a partial alternative to a cooling system. (See paragraph 2.4.5 for a discussion of chip cooling.) Conceivably, the processing power currently available through microelectronics may be insufficient. The computational and control functions needed in adaptive applications may be beyond current technology levels. If this is the case, this technology must also be improved.

Advances in microelectronics are commercially driven. The improvements required for aircraft adaptive engines may not result from this commercial motivation. In this instance, motivation may have to be supplied by the propulsion engine community.

Advanced adaptive applications may require control mechanisms beyond the capabilities of current control system architecture. It may even be necessary to distribute the control function over multiple mechanisms. Other subtechnologies, such as nonlinear control and fault tolerance, as well as adaptive control may be utilized.

Questions arise with the issue of certification. How is an engine that contains components and subsystems that are self-adaptive certified? Does every engine require separate certification? The design, as well as the validation and performance verification techniques of the manufacturer, becomes part of the certification process.

2.4.5. An Application of Heat Removal Technology - Microelectronics

A major problem in the growing microelectronic technology is chip cooling. If digital avionic systems are going to provide the capabilities required by advanced aircraft, this problem must be solved. As discussed in paragraph 2.4.4, research in the area of high-temperature semiconductivity offers some promise for improvement. Another approach is the use of micro heat pipes. The research and development efforts mentioned in this section were obtained from an article written by Dr. George Peterson; his article (Peterson, 1987) was a basis for this discussion.

A number of recent developments in heat pipe technology offer hope for alleviating the problem of microchip sensitivity to heat. These developments are as follows:

- Large silicon wafers are being cooled by devices that are one-fourth the size and one-seventh the weight of fins, coolant loops, or cold plates, traditional cooling devices. These new devices are being produced by Radio Corporation of America (RCA).

- Transistors are being cooled by a heat pipe method that reduces junction temperature to 63° C at a power level of 40 W. Hughes Aircraft is the developer of this method.

- Cooling systems have been developed with improved thermal response times and reduced thermal resistance for thyristors. Thyristors are solid-state devices that convert alternating current to direct current.

- Before the year 2000, very small heat pipes should be commercially available in the United States. These pipes should play an important part in the development of Three-Dimensional (3-D) chips.

According to Peterson, a heat pipe is defined and operates as follows:

"A heat pipe is a passive device typically consisting of a sealed container lined with a wicking material and filled with just enough fluid to fully saturate the wick. It operates on a closed vaporization and condensation cycle. Heat added to the evaporator section of the pipe vaporizes the working fluid. The high temperature and corresponding high pressure in this region cause the vapor to flow to the cooler end of the pipe, where the vapor condenses. Capillary forces in the wicking structure generate a positive pumping pressure that forces the working fluid back to the evaporator end."

The operation of the heat pipe is shown in figure 2.4-1. Heat pipes are superior to the best solid conductors, because the pipe has a much higher effective measure of thermal conductivity; this is due primarily to the transfer of heat by the concept of phase change of the working fluid.

Heat pipes are categorized by size: large, medium, and micro. Large heat pipes are used to remove heat from component housings; medium heat pipes are used as mounts for components; micro heat pipes may be embedded within chips. Large heat pipes may be .3 m or longer; micro heat pipes may be less than 5 cm long, and 1 mm in diameter; medium heat pipes cover the ranges in between.

In avionics, a typical use of large heat pipes would be to reduce the heat in housings for various systems or subsystems. Since the housing may be sealed at the points that the heat pipes penetrate, the housing remains a closed system, free from contamination by harsh environments. Medium heat pipes are used as a substitute for single-phase coolant loops or cold plates. These are commonly used to cool semiconductor devices, integrated circuits, or other components mounted on printed circuit boards. In an avionic system, several medium heat pipes may be arrayed into a substrate with a printed circuit board mounted on the substrate. In addition to cooling the printed circuit, this substrate may serve to reduce a "local hot spot". This will increase the mean time between failures of the circuit board. The three research and development efforts alluded to earlier cooling large silicon wafers, transistors, and thyristors use medium heat pipes.

Micro heat pipes should be commercially available in the 1990's. In a commercial semiconductor chip, all chip elements lie in a plane on top of a conducting material, either silicon or GaAs. The effective gate density of a single chip would be increased if the chip contained multiple planes of chip elements; this would enable element interconnections vertically, as well as horizontally. One of the problems inherent in this architecture is the heat buildup that would result. Micro heat pipes offer a potential solution to this problem. Current heat removal systems function at the 50 $W/cm^2$ level. 3-D chip architectures would require heat removal rates of 200 $W/cm^2$. Embedded micro heat pipes could function in this range. Heat accumulations in avionic systems have the following three components: First, the resistance from semiconductor junction to the device; second, the resistance from the device to the heat sink; and third, the distance from the sink to the ambient. The micro heat pipe reduces the heat buildup in the resistance from the semiconductor junction to the device, the largest of the heat components.

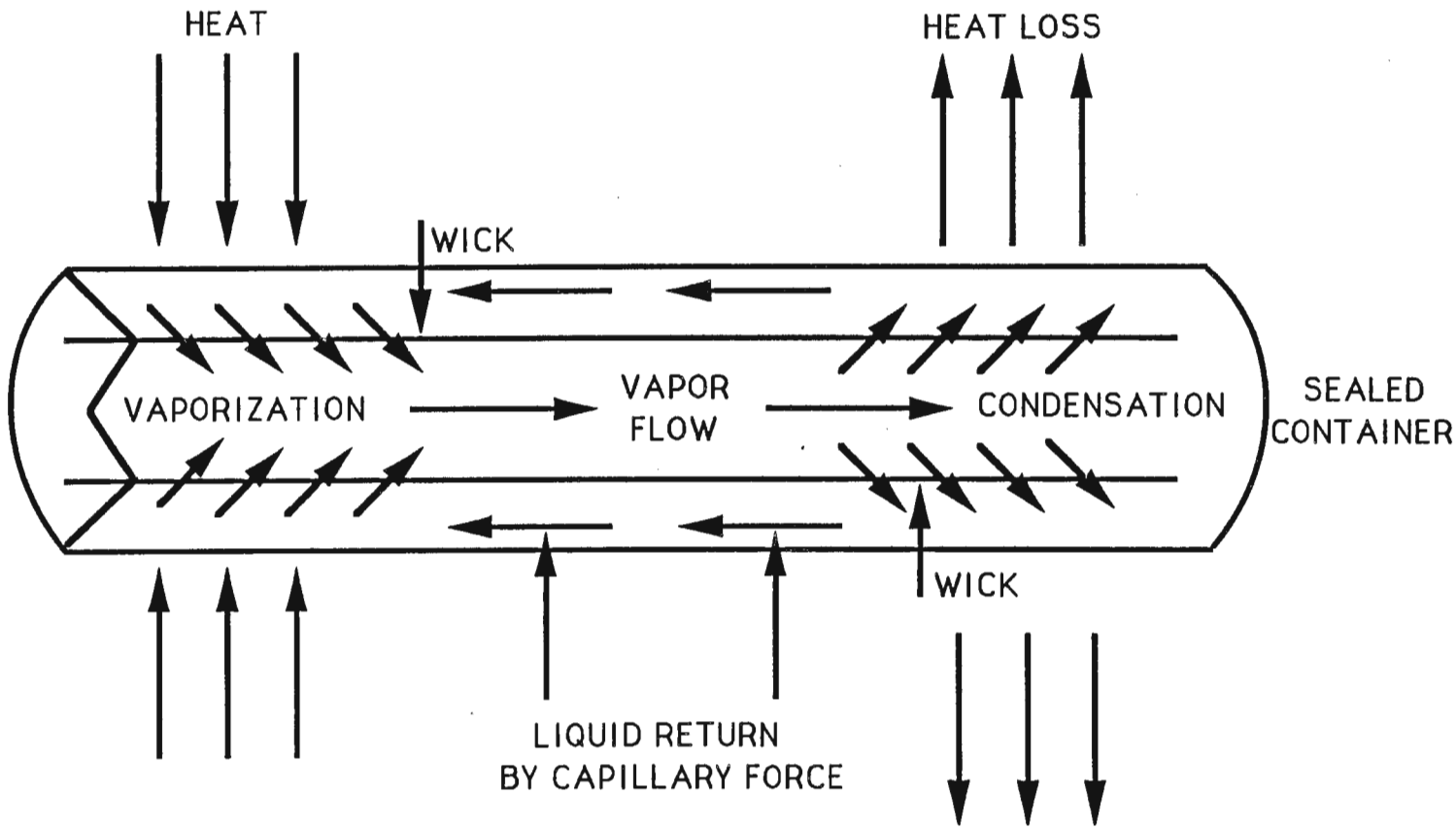



FIGURE 2.4-1. OPERATION OF HEAT PIPE IN MICROCHIP COOLING

One of the enabling technologies for advanced avionic systems, in lieu of the speed, propulsion, and friction anticipated, will be heat removal. Heat pipes are compact and light-weight. If micro heat pipes can become an integral part of 3-D chips, a major advance will have been made.

# 3. CERTIFICATION AND ADVANCING TECHNOLOGY

## 3.1. Challenges for the Certification Process and Regulations

Section 1 discussed the goals that will be driving aerospace technology for several decades. Conjecture was also given on the general technological advances that are required to support the achievement of those goals. The supersonic and transatmospheric aircraft required to meet their respective goals lie somewhat in the future; however, it is reasonably certain that there will be a new generation of commercial subsonic aircraft in use before the year 2000. The new technologies found in these aircraft pose a challenge to traditional certification procedures and regulations.

### 3.1.1. The New Generation of Subsonic Aircraft

Airbus Industrie's entry in the new subsonic generation utilizes today's technology; the assumption being that currently available technology is more than adequate to meet the needs of the airlines over the next decade or more. The Airbus Industrie A320 is a narrow body, short range aircraft; the A330 and A340 are the wide body, long range aircrafts. There are plans in Airbus Industrie to have their new aircraft ready for certification and service by 1992.

The FBW technology incorporated in the A320 will tax certification regulations. With FBW technology, heavy mechanical and hydraulic aircraft systems are being replaced by electronic wiring; the result is an "all-electric" airplane. In any Flight Control System (FCS) using FBW there is a set of sensors which produce electrical inputs into a flight control computer. There are also a number of actuators on the end of the communications link which drive control surfaces. The flight computers take signals from all sensors, cross monitor them, produce consolidated signals, and identify them as the best information. The computers then process the aircraft's control laws, determining what the control surfaces should be doing. Control signals are either forwarded directly to the control surfaces or to actuator drive units.

On the A320, FBW will be used on all surfaces for the rudder and tailplane trimming reversionary mode. Hydraulic activation is used on all surfaces. The FBW system incorporates two elevator aileron computers and four spoiler elevator computers. These computers drive left/right combinations of aileron, ground spoiler/speed brakes, and elevator left/right surface segments. The pitch axis control includes an automatically trimmed horizontal stabilizer. The ailerons and spoilers also receive inputs from a load alleviation system. For pilot control, side-sticks are used in lieu of control columns. The side-stick includes the autopilot disconnect feature, the radio communication/interphone switches, and a datum adjust switch for varying selected headings and vertical speeds.

Boeing plans to introduce its next generation aircraft, currently called the 7J7, by the mid-1990's. It has been announced that the 7J7 will take advantage of emerging technologies in propulsion, aerodynamics, structures, materials, aircraft systems, and cabin design. It will be powered by ultra-high bypass engines.

### 3.1.2. A New Application for Vertical Takeoff and Landing Aircraft

In addition to the new subsonic aircraft, the year 2000 may also see VTOL craft playing a major role in commercial aviation. This craft will most likely utilize either a tilt-rotorcraft or X-wing design. The military success of the Bell XV-15 and its larger version, the V-22 Osprey, give the tilt-rotorcraft design the advantage. Bell has issued a design proposal for the Bell D326 Clipper, a commercial version of the V-22. Numerous turboprop aircraft are now used by "third level" commercial carriers. Such aircraft as DHC-7's (DeHavilland Twin Otters) and Shorts 330's dominate this market. A civilian version of the V-22 would probably carry 44 passengers. The United States Marine Corps version carries 24 fully armed assault troops.

The potential advantages of VTOL craft replacing turboprop craft are significant. As mentioned earlier, travel between cities serviced by third level carriers would be between center-cities, not between major airports. To the traveller, this may mean greater cost since VTOL craft are less fuel efficient; however, the time saving may be great. VTOL travel in this marketplace has been estimated to be three times faster door to door than any other combination of vehicles. When the cost of ground transportation is considered, the cost differential may lessen, or even disappear. To the air travel community as a whole, the reduction in airport and airway congestion may be major; air travel may become considerably safer. Airline deregulation has increased the problem of congestion. Conventional jet aircraft now fly on many of the well travelled routes much more frequently with deregulation. Third level carriers have grown in number to service the routes deserted by the trunk lines; these carriers usually fly lower performance aircraft. The third level carrier aircraft use the same runways as the high performance aircraft flown by the trunk carriers. If the third level carrier used VTOL craft, simple airport facilities could be used that were remote from major airports and convenient to center-cities. An example of this is a feasibility study conducted by the Port Authority of New York and New Jersey on using the Manhattan Battery Park heliport for XV-15 landings and takeoffs.

Another potential market for tilt-rotorcraft is the corporate air travel market. This market is currently serviced by planes such as the Learjet. The advantages to the traveller and the air travel community are similar to those previously mentioned.

### 3.1.3. The Experimental Aircraft - X-29

The new generation commercial subsonic plane(s) and the tilt-rotorcraft are two new aircraft fostering new technologies that will impact certification. Another aircraft that is an indicator of even more advanced technological change is the X-29. The A320 incorporated new technology but at a moderate pace. Performance improvements of 5 to 10 percent over existing aircraft may be expected in

aerodynamics, structures, propulsion, navigation, communication, and FCS's. The A320 will utilize the CFM56-5 or IAE V2500 turbofan engines. The 7J7 is a bit more radical. The initial use of polymer material in the airframe, a flight management system with stability augmentation, and an ultrabypass engine is estimated to improve performance another 10 percent beyond the latest turbofan aircraft. A generic subsonic aircraft using laminar flow technology and ultra high bypass propfan engines, as well as active stability augmentation and very high aspect ratio supercritical wings, is estimated to improve overall performance by 60 percent over today's aircraft. The X-29 is considered most radical.

The X-29 is an advanced technology experimental aircraft developed under the guidance of the DARPA and the National Aeronautics and Space Administration (NASA). The X-29 incorporates the following new technologies:

- Forward Swept Wing: provides aero efficiency.
- Close Coupled Canard: low-speed control.
- Thin Supercritical Airfoil: control at high angle of attack.
- Discrete Variable Camber: enhanced maneuverability.
- Relaxed Static Stability: favorable volume distribution.
- Three-surface Control: increased agility.
- Aeroelastic Tailoring: solves aero divergence.
- Composite Wing Covers: reduces drag and weight.
- Triplex Digital Design: flexible/multiple modes.
- Fly-By-Wire: tailor handling qualities.

The X-29 possesses an inherent 35 percent negative static stability margin: a design feature enabling extensive maneuvering abilities, yet offering a severe control system problem. The FBW DFCS stabilizes the aircraft, permitting normal, acceptable handling. The control laws for the NASA's Space Shuttle, NASA's F-8, and the Air Force AFTI/F-16 could not stabilize this degree of negative static margin. The pitch instability of the X-29 is four times that of the AFTI/F-16. Therefore, the development of the X-29's control laws and a DFCS capable of stabilizing the aircraft is a major accomplishment.

The DFCS was designed to exploit the aerodynamic advantages of the aircraft, e.g., the forward swept wing and the close coupled canard. The DFCS is multi-mode and has three channels. The system uses seven F-16 National Waterlift actuators to drive the two canards, six segmented-trailing-edge flaperons, and a single rudder. A newly designed actuator drives each of the two strake flaps. Flight control computers, air-data sensors, and an attitude reference system are the remaining components of the DFCS. Each of the three digital computers has an analog backup in parallel. Exercise of the several modes provides a graceful

degradation feature when components begin to fail. Normal operations use the full digital mode and the associated automatic camber control. A normal approach mode controls takeoff and landing. There are two reversion modes:

- A digital reversion mode using different software than that used in the normal mode. This mode does not utilize noncritical sensors.

- An analog reversion mode. This mode is only activated when there is a generic digital control fault.

The up and away, power approach, and ground contact for weight on any wheel operational regions are available with each mode. There is also a precision approach control mode which enables the pilot to directly control flight path angle by means of a pitch stick. When switching between modes, precautions have been taken to minimize transients.

While the immediate beneficiary of X-29 technology is the ATF program, it is a matter of time before advanced civil aircraft will likewise reap benefits. A successful experimental aircraft is expected to increase performance by five to ten times that of existing aircraft. Experimental aircraft are being studied jointly by NASA and the Navy for an oblique wing design, and by DARPA and NASA for an X-wing rotorcraft design. The future holds promise for an experimental aerospace plane. The technologies fostered in these experimental craft extend beyond subsonic flight and into supersonic and hypersonic flight.

### 3.1.4. New Aircraft - Summary Comments

The requirement for change is the same whether the certification domain is transport, rotorcraft, or small general aircraft. New systems oriented to today's technological categories will affect changes in certification regulations and procedures. Examples of these are Upper Surface Blowing (USB) propulsive lift and the propfan engine. FAA certification procedures, as of 1984, did not allow for the advantages of powered lift technology as incorporated in USB designs. NASA has found the USB design to be fuel-efficient for low-speed, Short Takeoff and Landing (STOL) flight; current research is testing its performance in high-speed flight. If the technology is adapted to civil use, FAA certification must follow. The propfan propulsion design is a resurrection of the turboprop engine. The propfan, a result of NASA research, incorporates new propeller technology in an energy efficient engine. Paragraph 2.4.4 of this tutorial introduced microelectronics to a propulsion system on a theoretical basis. Most propfan designs utilize microelectronic digital devices for propeller control functions, devices not yet FAA certified for that purpose. The electronic devices will be located in the vicinity of the corrosive exhaust system of the engine, a high temperature environment. While the microelectronics will be used for propeller control, e.g., pitch change control, it will not be used for adaptive control of the engine elements. The propfan is an example of revolutionary engine/propeller design, augmented by digital avionics.

## 3.2. A Specific Problem for Certification - The Propfan

Current air-worthiness regulations do not directly cover the design features present in the latest technological advances. Propfan or ducted/unducted

variable pitch fan propulsion system designs are examples. As a response to this, the FAA is taking steps to assure that certification programs will accommodate the latest technologies. "New Propulsion Concepts and FAA Certification" (Aerospace Engineering, 1988) describes the difficulties in certifying the propfan.

The intention of the Federal Acquisition Regulation (FAR) is to classify aeronautical products into types and to define the minimum requirements for certification. Each product type is covered by a unique regulation called a Part. The product type is evaluated against its respective regulation. As examples, Part 25 contains the criteria against which transport category aircraft are evaluated; Part 33 contains the criteria for aircraft engines; and Part 35 covers aircraft propellers. In the past, engines and propellers with the propfan design have been classified as two distinct product types. This classification must be reconsidered for accuracy. FARs permit the stipulation of special conditions when the existing regulation does not seem to cover the particular product type. The intention of the FAR is to promote public safety, not define limitations of aircraft design. Special conditions are required for the propfan. These conditions must guarantee that the aircraft containing the propfan propulsion system achieves the safety standards demanded by Part 25, while the propfan satisfies the standards established in Parts 33 and 35.

One of the problems with propfan certification is the requirement for containment of the propeller blade. Part 33, the engine regulation, stipulates that any failed or released fan blade must be contained within the engine housing. This requirement is based on traditional turbofan engine design. In a turboprop-based design, the blades are not shrouded and do not have any means of containment. With safety as the underlying motive, it is then necessary to provide the design measure that minimizes the chance that a blade will fail or be released. Part 25, the transport aircraft regulation, stipulates that the aircraft must be able to withstand the impact of a failed or released blade with no structural damage that would pose a hazard to the safety of the aircraft. Applicants for certification must demonstrate that the aircraft can sustain the structural damage caused by a failed or released blade, and that the aircrafts other critical systems will continue to operate correctly during the period of blade loss. The propfan applicant should demonstrate that in his aircraft the probability of blade loss is acceptably low, and that the structure of his aircraft is such that blade loss would not jeopardize the safe continuance of the flight.

A second problem is that the propeller hub found in a propfan design may differ significantly from that found in traditional propeller aircraft. The differences are in the areas of size, number of blades mounted, and temperature environment. Propfan propeller hubs are large, have several blades mounted, and operate in a high temperature environment. Blade retention strength is the prime safety consideration for the propfan hubs; the FAA requirement is that the hub's blade retention strength be sufficient to ensure that the loss of a blade is a virtual impossibility. If the hub is part of the turbine rotor, then its shattering may be a greater threat to the structure of the aircraft than blade loss. Adequate margin between load and material strength should be a design attribute to eliminate this safety concern. All propfan components, the hub included, must demonstrate that they do not deteriorate measurably when exposed

to the exhaust stream of the engine (high temperature vulnerability). To summarize, the propeller hub (blade retention facility) cannot be sufficiently validated by the precise criteria contained in Part 35. The criteria must be supplemented with special conditions to show that the same safety objective is achieved.

A third problem is the propfan pitch control system. It is a digital electronic control system. Digital avionics have now been incorporated into the propulsion system. Digital electronic control devices have not yet been certified by the FAA for propeller control. The pitch control mechanism for the propfan also includes mechanical or hydraulic linkages and a two-stage counterrotating rotor. All of these are not traditionally found in the pitch control system. The safety goals are as follows:

- No failures that result in excessive propeller speeds that may result in fragmentation.
- No failures that will result in a high drag condition.
- No control system failure that results in an undesirable command to autofeather (automatically move the propeller to zero drag position).

Current industry design practices have resulted in pitch control mechanisms that satisfy these safety requirements. The problem of certification is to assure that the corresponding propfan mechanism demonstrates the same degree of airworthiness.

A fourth problem is the design of the propfan propeller blade. These blades may be more susceptible to time and usage than traditional blades. Propfan blades are characterized by swept platforms, wide chords, and thin airfoils. They are made of layers of composite material, enclosing a light filler material. As such, they may be more vulnerable to chemicals, foreign object damage, heat, and humidity. Because of this, it will be necessary to track each blade and hub to determine its performance history. Periodic inspections for durability, flutter, vibration, and stress will also be required. This is not required for normal propeller blades and hubs. These are usually certified for an unlimited life, given that certain maintenance levels are sustained. As a result, detailed records of time-in-service are not maintained. Another safety concern over the propfan blade is that composite material does not conduct electricity as well as metal; therefore, the effect of lightning strikes must be determined.

Other potential problems with propfan certification lie in the areas of potential gearbox failure due to the high horsepower ratings, noise, and safety requirements for aft-fuselage mounted engines. As examples of the latter, minimum clearance standards are different for aircraft with engines mounted behind the fuselage and ingestion of foreign objects is of greater concern when propellers are located at the rear of the craft.

The propfan is an example of a product of a growing technology that has novel and unusual design features. These design features are not addressed in current

certification regulations. However, the issuance of and compliance with special conditions will assure that proper controls are in place for assuring public safety.

### 3.3. A Specific Problem for Certification - The Rotorcraft

Certification rules for helicopters have evolved as helicopter technology has advanced. As avionic systems utilizing microprocessor technology are increasingly used in helicopters, these certification procedures will be challenged. There is a major concern with respect to the impact of digital electronic controls on civil helicopters. Technologies such as mechanical systems, pneumatic systems, and hydraulic systems are mature and are covered by the "specific rules", of air-worthiness regulations which state specific requirements. As electromagnetic and electromechanical technologies were increasingly used in helicopter systems, certification procedures migrated from specific rules to "general rules". With a general rule, more emphasis is placed on achieving a desired safety level and less on how that safety level is achieved. One of the purposes in doing this was to maintain safety standards, but not constrain design. Some of the difficulities of certifying helicopters using advanced technologies are described in "Certification Issues in Civil Helicopters" (Brahney, 1988). The information contained in this section is based on "Certification Issues in Civil Helicopters" (Brahney, 1988) and information obtained from John D. Swihart, Jr., and Richard Vaughn of the FAA.

#### 3.3.1. The Full Authority Digital Engine Controller

The FADEC was one of the first systems introduced in a helicopter that utilized advanced control technology. The FADEC uses microelectronic technology instead of the traditional hydromechanical technology; the former offers increased accuracy, improved control modes, better maintenance, and substantially reduced life cycle costs. The FADEC must operate in a hostile environment for thousands of hours without failure or maintenance. The components of the FADEC use leadless chip carriers and include advanced 16-bit microprocessors with error correcting memory, analog/digital converters, pressure/thermal sensor signal converters, and digital data bus interfaces. The Sikorsky S76B is equipped with the FADEC.

There are two general rules in FAA regulations that are applicable to the certification of helicopters which use systems based on advanced control technologies. FAR 29.1301 covers the operation of the systems in the aircraft; FAR 29.1309 specifies the interaction between the system and the helicopter environment, and the acceptable ramifications of system failures. Software validation for the FADEC requires demonstrating through testing in the helicopter environment that the system is not only logically correct but that the user's requirements are satisfied. Errors may exist that result from software deviations from requirements and from timing considerations during program execution. The software must be evaluated by testing that begins with a sequence of stand alone tests, proceeds through simulated engine tests, continues with testing on an engine test stand, and culminates with flight test. A difficulty with software testing is that general rules in certification require that the testing procedures and results be verified. This means that the FAA must review documented test results, as well as software development

documentation. In the absence of a standard, this documentation will be unique to each manufacturer. Until recently, there was no movement for development of standards for software documentation.

### 3.3.2. Fly-By-Wire Technology

The incorporation of microprocessor technology in helicopter FCS's is more recent than its use in power plants. This innovation has come in the form of FBW. The V-22 Osprey tilt-rotorcraft is an example of a tilt-rotorcraft that utilizes FBW technology. Since FCS's are more relevant to flight safety than FADEC, this innovation is viewed differently by the FAA. In FBW technology, flight safety will be almost completely in the hands of microprocessor controls. The certification of these FBW subsystems for safety is of critical importance. The specific and general rules that are applicable to these subsystems must be sufficient for assuring public safety. There are three major concerns in this area:

- The reliability of the electrical power supply system.
- The security of the FBW subsystems against electromagnetic hazards.
- The reliability of embedded software.

While these concerns are mentioned in the context of certifying FBW technology in rotorcraft, they are equally applicable to certifying FBW in fixed wing aircraft, e.g., Airbus A320.

#### 3.3.2.1. Electrical Power System Reliability

With the introduction of digital electronic control systems in FCS's, the importance of the electrical power supply systems has been elevated. Since the FBW subsystems are critical to flight safety, the electrical power systems supplying energy to them must always operate at a level sufficient for sustaining the FBW subsystems' functioning. Consequently, the validation and certification of these electrical power systems is of major importance to the FAA. The power systems must be tested thoroughly under both normal conditions and conditions in which system errors occur.

#### 3.3.2.2. Security Against Electromagnetic Hazards

Prior to FBW, a helicopter's avionic systems did not perform flight-critical functions. Usually the avionic systems controlled only instrumentation not essential for the safe landing of the helicopter. In this situation, electromagnetic hazards were of little concern. With the proliferation of microprocessors throughout flight-critical systems, this is no longer the case. Since High-Energy RF (HERF) fields (generated by commercial radio broadcast transmissions and radar installations) and lightning strikes occur with some frequency, their effect on critical electronic rotorcraft subsystems must be considered. For the purpose of certification, although there are some similarities in the protection against HERF fields and lightning strikes, the major differences to be considered include:

- Amperage: Amperage from RF fields is in the milli- to microampere range while amperage from lightning strikes is in the kiloampere range.

- Frequency: Frequencies from HERF are mainly greater than one megahertz. Frequencies from lightning are mainly less than one megahertz.

- Interaction: Lightning attaches directly to the aircraft in isolated spots. A HERF field envelopes the entire aircraft.

- Effects: A major difference in effects is that lightning can cause burnout or destruction while HERF can only upset or cause glitches in systems.

Because of the use of electronic devices in flight-critical subsystems, the FAA is issuing voluntary testing standards for certifying that the electronics of an aircraft are adequately shielded against electromagnetic hazards. The need is acute because not only are flight-critical subsystems becoming more susceptible to electromagnetic hazards, but protective metal airframe skins are being replaced by nonconductive composites. Since graphite composites are not good conductors, they cannot shield interior circuits. Currents resulting from high-energy fields now move directly along interior conductors. Advances in circuitry also aggravate the problem for the following reasons:

- Increases in microcircuit bandwidths result in increases in the bandwidth against which they must be protected.

- Miniaturization of circuitry decreases the operating voltage which increases the amount of protection required.

Protection from HERF fields can be achieved by either shielding the desired component or subsystem with a conducting material or by isolating the component or subsystem electrically from induced charges. A combination of the two methods may be most desirable for reasons of weight and cost.

The FAA standards initially only covered structural damage to aircraft; but the latest standards, modeled after those developed by the Society of Automotive Engineers (SAE), are more comprehensive. These standards are documented in the SAE Committee Report (SAE Committee AE4L Report No. AE4L-87-3). They were the basis for early approvals of FADEC systems. An aircraft meeting the new standards is expected to be able to withstand lightning strikes with currents in the 200 kiloamp (kA) range. These strikes usually occur only once every few hundred aircraft-years of flying, but the standards should be adapted to protect aircraft from the maximum currents and maximum rates of rise of lightning.

While the FAA standards may be in place, the avionics technology required to meet them will not be easily achieved.

3.3.2.3. Embedded Software Reliability

The standard accepted by the FAA for certifying software in flight-critical systems is Radio Technical Commission for Aeronautics (RTCA) document DO-178A. However, this document states that the procedures contained therein are not sufficient for assuring an acceptable level of safety for flight-critical

systems. The FAA has recommended that manufacturers providing systems with embedded software conform to DO-178A. Exceptions have been made when redundant software is present. In this instance, the FAA has stipulated that:

- There must be at least two different software systems.
- There may be no shared software modules.
- Each software system must be capable of performing all critical functions independently of another software system.

3.3.2.4. A Backup for FBW - Fluidic Flight Control

Some of the concerns expressed in paragraphs 3.3.2.1 and 3.3.2.2 would be alleviated if there were redundancy to the FBW system. Test results from experimentation with a fluidic FCS serving as backup to FBW is discussed in "Fluidic Flight Control: Early Test Results" (Aerospace Engineering, 1988).

In hostile environments, such as high temperature, vibration, and Electromagnetic Interference (EMI), fluidic flight control is being considered as a functionally redundant, nonelectric backup alternative for FBW. The motivation behind this research is the failure probability that exists in FBW systems that is non-existent in traditional hydromechanical FCS's. The composite FCS is a full authority, two-axis, lateral-directional control system for a Navy T-2C aircraft; the tests are being conducted at the Naval Air Development Center.

The selection of either FBW or the fluidic flight control modes is under pilot control. There are no transients when the modes are switched because the idle system is always in active standby status. Early test results indicate that using fluidic technology as a backup for FBW in aircraft subject to high temperature buildups, vibration, and EMI is a viable alternative.

3.4. The General Problem for Certification

In paragraph 3.2 the propfan propulsion system design was critiqued from the certification point-of-view. In paragraph 3.3, the difficulty of certifying new rotorcraft because of its use of recent advances in microprocessor technology was detailed. In this section, difficulties of a broader nature will be discussed.

3.4.1. Intelligent Components, Subsystems, and Systems

Intelligent components, subsystems, and systems that are applied to many systems involving microprocessor elements are emerging as individual technologies. Their emergence as technologies has paralleled the technological growth made in digital computers and storage devices on the microelectronic level.

3.4.1.1. Adaptive Components and Subsystems

Adaptive components and subsystems are intelligent components that are characterized by digital controls containing logic that adapts to meet the needs

of different inputs or environments. These adaptive components and subsystems create problems for type certification. Questions arise including:

- How are requirements defined?
- How is a system containing one or more self-adapting subsystems type certified?
- Does each system require independent certification?

These questions must be answered if certification regulations and procedures are to accommodate advancing technology.

The cost of fuel is a major factor in acceptance of an aircraft design. For this reason, such measures as seat-statute-miles-per-gallon have gained acceptance as figures of merit for judging propulsion system design. Figure 3.4-1 (Lupinetti, 1987) demonstrates an estimated tripling in performance between the introduction of the first generation turbofan engines through the anticipated propfan engine. One element in the current trend is to gain further engine performance improvement through "smart engines". At this point, digital avionics impacts propulsion systems. Paragraph 2.4.4 of this tutorial discusses a "smart" engine in detail. Aircraft gas turbine engines have historically consisted of an integrated set of open-loop subsystems with a feedback fuel control. Only a few sensors are used to provide inputs, through a feedback process, to a simplified engine model; the controller is based on this model. Restricted onboard computing capacity, insufficient sensor reliability, and insufficient sensor adaptability to harsh environments have dictated this simple control scheme. Advances and anticipated advances in microcomputer technology remove much of this restriction. It has been proven that the use of real-time intelligence (available today with the processing power of microprocessors and data stores) in closed-loop control can be used to improve performance and operability of propulsion system components and subsystems. Propulsion system components and subsystems include, but are not limited to, inlets, fans, compressors, turbines, and nozzles. The theory is that advances in onboard processing capability and sensor technology will enable the incorporation of feedback control within engine components, thus optimizing engine performance and life expectancy.

At this time, research activities are underway identifying the ways to use real-time intelligence for improving propulsion system controls. The specific areas expected to benefit from this identification are compressor stall alleviation, active clearance control, secondary airflow modulation, and active pattern factor control based on blade stresses and temperatures. The enabling technologies involve improving sensor systems, computer systems, effectuator systems, and component models. The continuing revolution in microelectronics suggests that control over factors determining fuel consumption will continue to grow. Among the technological elements of this control are adaptive control, nonlinear models, model fidelity, and fault tolerance. This improvement should alter fundamentally the nature of gas turbine components and subsystems from an open-loop to a closed-loop operation, resulting in optimal performance.

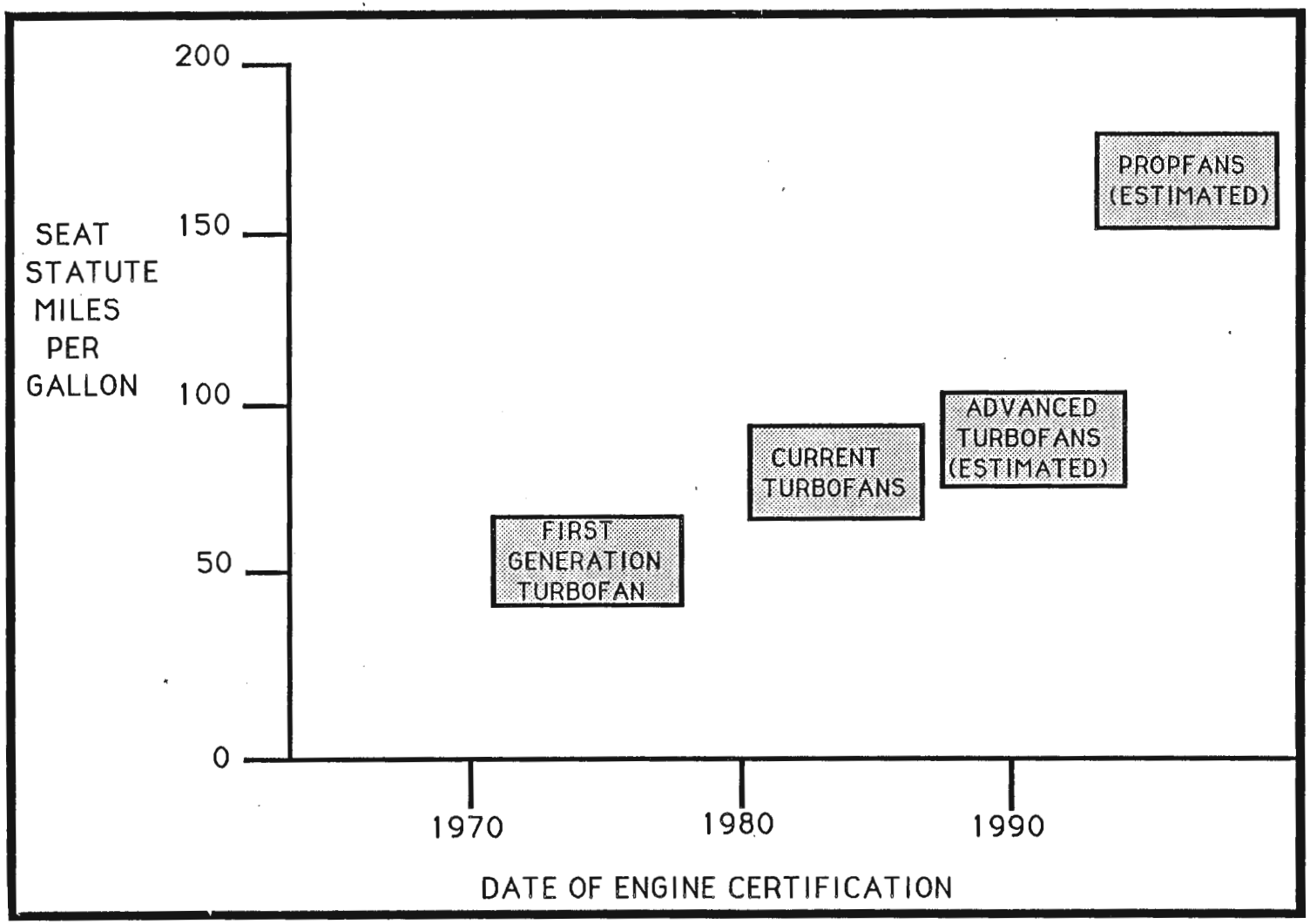



FIGURE 3.4-1. AIRCRAFT PERFORMANCE BY PROPULSION SYSTEM
(Lupinetti, 1987)

3.4.1.2. Artificial Intelligence and Expert Systems

Artificial Intelligence and expert systems were introduced in paragraph 2.4.1 as being an elemental discipline for emerging avionic technologies. AI is a branch of computer science that creates "intelligent" computer systems that understand, learn, reason, and solve problems. Expert systems are systems that use knowledge and construct inferences regarding problems that would otherwise require human expertise; expert systems result from an application of AI. Rapid advances in AI have only been possible because of the increases in computing power and data manipulation enabled by today's digital processors and data stores. Information processing technology has been advancing approximately by a factor of ten every five years; this pace has been maintained for the past thirty-five years. The problem posed for certification of a digital avionic system utilizing AI is similar to that discussed in paragraph 3.4.1.1; the system does not conform to a rigid set of specifications. Its response to a given stimulus is dynamically determined and time dependent. At a different point in the learning process, the system may respond to the same set of stimuli differently than before. Adaptive control, as discussed in paragraph 3.4.4.1, may incorporate learning, but is more likely not to. For this reason AI-based systems' certification rationale must be based more on a demonstration of the manufacturer's design and the integrity of the manufacturer's performance verification techniques, than on conformance to rigid specifications.

Artificial Intelligence, while having been an academic subject for many years, is still in its infancy. The technology for implementing quasi-intelligent subsystems should be in place by the year 2000. At this time it should be mature enough to support its application in aeronautical system products and processes. An expert system is an example of a quasi-intelligent system. Using a set of facts and rules, the expert system is able to reason. Figure 3.4-2 is a schematic depicting the structure of an expert system. Facts represent inputs to a situation interpreter and actor, the "inference engine". The set of facts is referred to as a "situation base". The inference engine also processes rules from a "knowledge base". The result of processing the facts against the rules is an action or conclusion. Facts are inputs derived from particular sensing mechanisms. Rules are complex data structures that define what relationship of inputs (facts) leads to a certain conclusion (action). The inference engine is an interpretive computer program that applies inputs to rules, and initiates an action when a rule is satisfied.

The major difficulty with expert systems is the need for complete specification. In the design of a large system, a designer must provide a complete definition of all permissible states together with a number of state transition diagrams which specify all transitions between states. An expert system uses a complete set of rules that explain how to find all permissible states, and how to determine which transitions are permissible between states. An expert system requires that all knowledge on a system's functioning be gathered and encoded into a knowledge base. The operation of the system is modified by changing the rules that constitute this knowledge base.

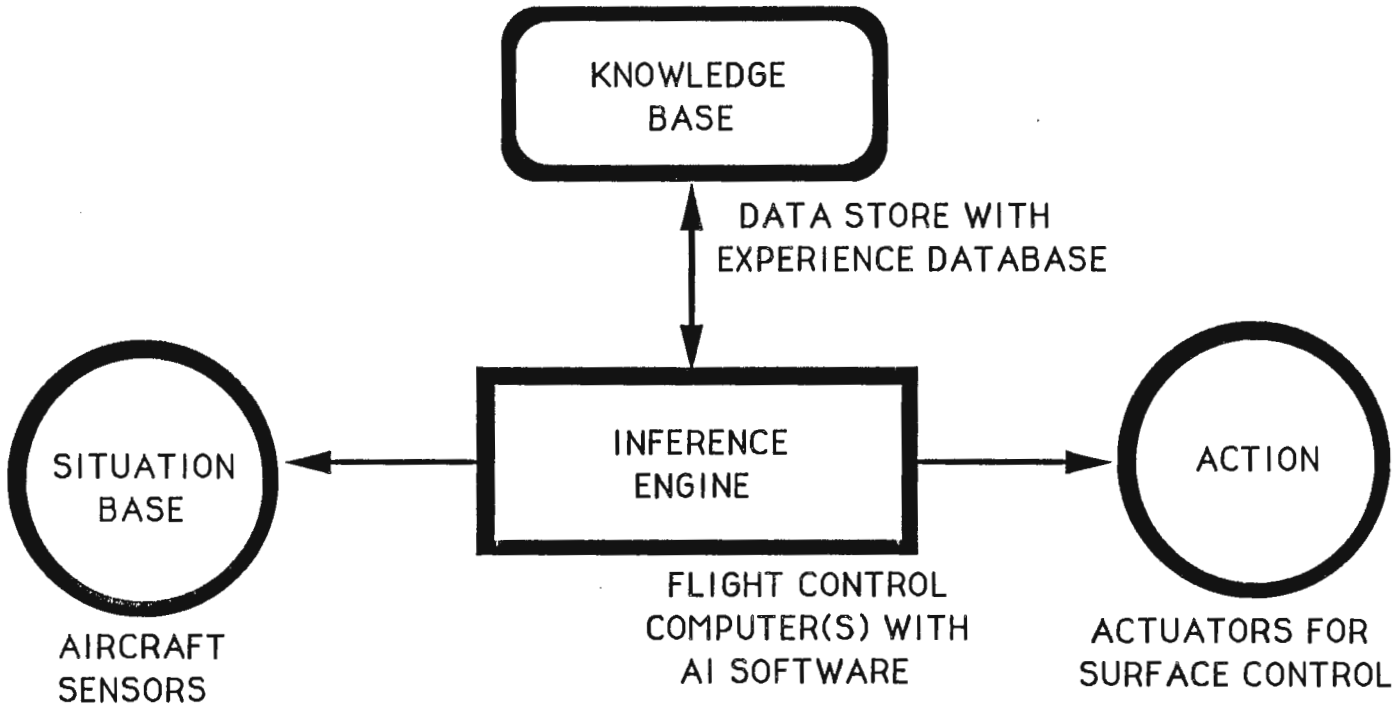



FIGURE 3.4-2. EXPERT SYSTEM

Improvements in this technology are not tied to the inference engine, but to the acquisition and organization of knowledge in the domains that are relevant to the system. The manual acquisition organization and the structuring of the knowledge base are the major bottlenecks in this process. A knowledge base of 1000 rules may require many man-years of effort to develop. By the year 2000 it is anticipated that there will be specialized expert subsystems in common use that will utilize knowledge bases containing several thousand rules. These subsystems will be limited in scope and applied within a distributed, multifunctional airborne environment. While a limited number of such systems exist today, they are restricted to non-time-critical applications. By the year 2000, expert systems with moderately sized knowledge bases will be used in real-time applications.

As with any system, an expert system is only capable of accomplishing what its designer plans. When we say a system "learns", we mean a knowledge base is improved. The difficulty of translating a discipline into a knowledge base is not to be underestimated. In doing this, structure and quantification are being applied to subjects unaccustomed to it. Frequently, these subjects behave in the planned manner for reasons other than those anticipated by the designer.

In addition to expert systems, AI is also an elemental discipline for Theorem Provers, Robotics, Goal Seeking Systems, Vision, Speech Recognition, and Natural Language Understanding. These are expected to result in new technologies that will be beneficial to the aeronautics community.

### 3.4.2. System Integration

System Integration was introduced in paragraph 2.4.2 as being an elemental discipline for emerging avionics technology. Traditionally, avionic systems have consisted of a collection of individual components, each performing a function independent of other aircraft systems. With the advent of powerful microprocessors, avionic system designers were able to fully integrate navigation, display, and control systems. Integration implies sharing of the data produced by several different sensors and computers. This sharing has the cooperative effect of improving the quality of decisions that can be made by either human or electronic intelligence. When Automatic Control Systems (ACS) are used to stabilize the aircraft, conventional aerodynamic stability requirements can be relaxed. Modern tactical aircraft have low, and even negative, levels of longitudinal static stability; thus, maneuvering capability and cruise performance are enhanced. It is only possible to fly an aircraft with this level of longitudinal static stability because of a full-time, full-authority automatic stability augmentation system that stabilizes the aircraft. Integration also implies the physical consolidation of system functions, which offers its own advantages.

Guidance, Navigation, and Control (GNC) systems will be a forcing function in aircraft design during the coming decades. With today's microprocessors, the designer can exploit the interaction and integration of aerodynamic, structural, and propulsion system controls to provide better, more efficient aircraft.

The advantages of integrated design have been demonstrated in experimental aircraft. The following are two examples:

- Integrated digital control of engine inlet and autopilot systems have resulted in a significant range increase.

- Integration of wing extensions with an active control system have resulted in significant cruise performance improvements.

These are considered to be first generation improvements resulting from system integration. More significant improvements should be realized by the year 2000. The Aeronautical Policy Review Committee (Review of National Aeronautics Policy) states that:

> "Systems integration will play an increasingly important, if not dominant role in the development of advanced technology aircraft... advances in the traditional aeronautical disciplines will no longer ensure a superior product."

While hardware advances in the next two decades will provide greater computational capability, improved reliability, and reduced component cost, these will not be the pacing technology developments. The application of a multidisciplinary design approach to these developments is the challenge of research and development.

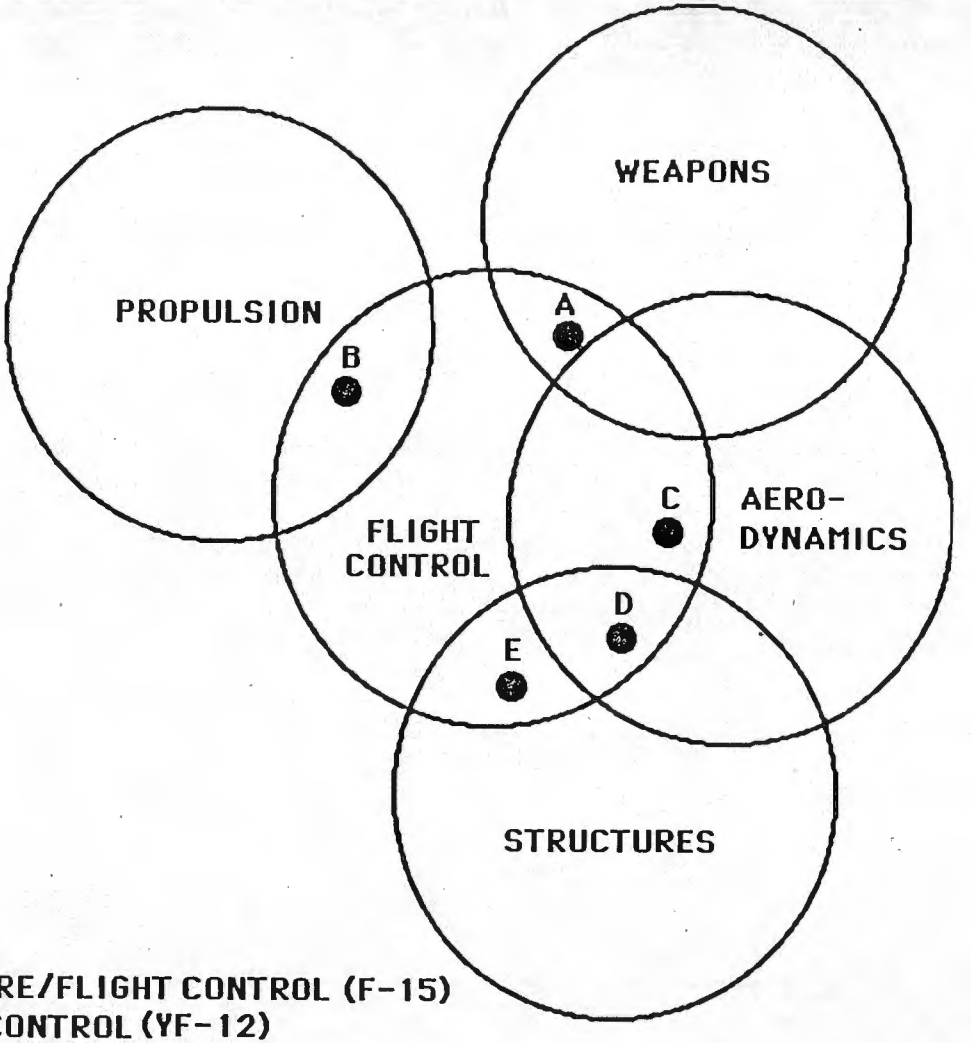
The following is an example of the demands that will be placed on a tactical aerospace vehicle by the year 2010 (Aeronautics Technology Possibilities for 2000: Report of a Workshop, 1984).

> "... it will have to deal with massive amounts of information such as: targets, ground and airborne threats, navigation data, route options, weather, weapons, aircraft system status, and commands. This information will have to be collected, stored, processed, correlated, integrated, analyzed, and acted upon in a very short period of time."

The aircraft computer systems must be integrated to accurately process this magnitude of data. Equivalent integration requirements will exist for civil aircraft.

3.4.2.1. Current Efforts in System Integration

As mentioned earlier, system integration is now at the first generation level. Intermediate level integration involves the linking of selected functions or subsystems for the purpose of achieving improved performance and has been accomplished. An example of this is the coupling of the throttle, flight control, and navigation systems to provide automatic flight path control. Figure 3.4-3 depicts the state-of-the-art schematic for system integration above the intermediate level. The five circles represent the five major disciplinary areas of aeronautics. The intersections represent functions that have been integrated as a result of a design decision. The solid circles within intersections represent those integrated functions that have been accomplished in production or research aircraft.




A INTEGRATED FIRE/FLIGHT CONTROL (F-15)
B COOPERATIVE CONTROL (YF-12)
C SUPERMANEUVERABILITY (HIMAT)
D FORWARD-SWEPT WING (X-29)
E STRUCTURAL MODE CONTROL (B-1)

FIGURE 3.4-3. SYSTEMS INTEGRATION

The potential advantages of these integration efforts are enormous. The following results are expected:

- An integrated GNC will permit engineers to trade inherent stability for weight and drag reductions.

- An integrated control and structural system will permit engineers to design structures to achieve minimum weight by load control at key points, maximizing load factor capability over the complete dynamic range of the aircraft.

An example of technological advance for the purpose of enabling system integration is the Advanced Fuel Management (AFM) program. This program is under Navy sponsorship; its purpose is to improve aircraft performance by a full integration of the propulsion control and FCS's. A thorough discussion of this program is given in "Moving Closer to Fully integrated Control" (Brahney, 1986). Engine control technology must be improved to make this system integration possible. The unit that controls the propulsion system must be as reliable as the FCS's. The design guidelines ensure that the control unit continues to operate the engine, without performance degradation, following the failure of any single mechanical or electrical component. Selective redundancy was included on the control unit to achieve reliability. The inclusion of selective redundancy in the control unit was the method by which reliability was to be achieved.

The decision to use selective redundancy as a means of achieving desired reliability levels eliminated "single string" electronics architecture as a possible approach to control unit design. The FADEC and Full Authority Fault Tolerant Electronic Engine Control (FAFTEEC) established that dual redundant system architecture is a viable approach to providing high reliability in controlled propulsion systems. However, neither of these designs was adequate for tactical aircraft of the 21st century. For this reason the AFM program was initiated. According to the design specification, the AFM system must integrate fuel, hydraulic, electrical, and mechanical components into a dual channel, full authority, fault tolerant, engine control system. It is responsible for monitoring engine status, reading appropriate data, and scheduling selected parameters for the purpose of achieving a desired engine performance level. Power Level Angle (PLA), certain speeds, temperatures, and pressures are the primary inputs. The outputs are scheduled parameters that relate to engine efficiency, e.g., gas generation and augmenter fuel flow parameters. Figure 3.4-4 is a schematic of the AFM.

The AFM system is composed of three subsystems: the fuel subsystem, the airflow subsystem, and the electronics subsystem. The major components of the fuel subsystem are flow sensors, fuel pumps, and valving. This subsystem is responsible for pumping and metering the fuel to the gas generator and augmenter. The major components of the airflow subsystem are vane and nozzle actuators. This subsystem is responsible for controlling the variables that affect engine airflow. The major components of the electronics subsystem are engine harnesses, generators, and controllers. This subsystem monitors the fuel and airflow subsystems. Figure 3.4-5 demonstrates how the AFM system interfaces with the FCS and the propulsion system (engine).

### 3.4.2.2. Barriers to Implementing System Integration

Experience with integration efforts, like the AFM, shows that the most significant barrier to integrated control is the difficulty of incorporating integrated systems in aircraft. Incorporating systems is difficult and very risky due to the uncertainty of the performance of the aerodynamic, structural, propulsion, and environmental systems under normal and failed flight conditions. Additional problems to be considered are implementing safety precautions to ensure reliable and safe operation and operating without concise measures for rating performance levels.

Current technology is not designed to facilitate system integration. There is emphasis on the Input-Output concept of system operation. The system is not designed to process all the different combinations of inputs and outputs, and there is not enough understanding about how the different technologies interface. Systems engineering is attempting to solve this interface problem, but is handicapped by the absence of higher-level design tools, accurate model definitions, and experience in applying tools in real-world situations.

Some specific design needs are as follows:

- Development of measurements for successful/unsuccessful flight systems performance in aircraft that incorporate state-of-the-art technology, in particular, high-authority DFCS.

- Development of design methodology and techniques for designing in an environment of distributed, parallel processor architectures.

- Development of system architectures that provide high degrees of reliability and dynamic reconfigurations for flight-crucial systems. There is a supporting need for the development of test and assessment aids for ensuring that systems developed satisfy the requirements.

- Development of practical optimization algorithms to accurately extract data from large databases. This processing will be in near real-time mode to assist the pilot in flight path prediction and flight decision making. Distributed and parallel computer architectures, and supporting software may be required.

- Development of software validation and verification methods that are less subjective than the methods currently in use and that maximize existing automation technology.

- Development of techniques for task allocation among crew and vehicle systems based on mission requirements and human error theory.

- Development of models for use in multidisciplinary design for highly interactive systems. Specific examples are as follows:

  - Dynamic response of aeroelastically tailored structures.

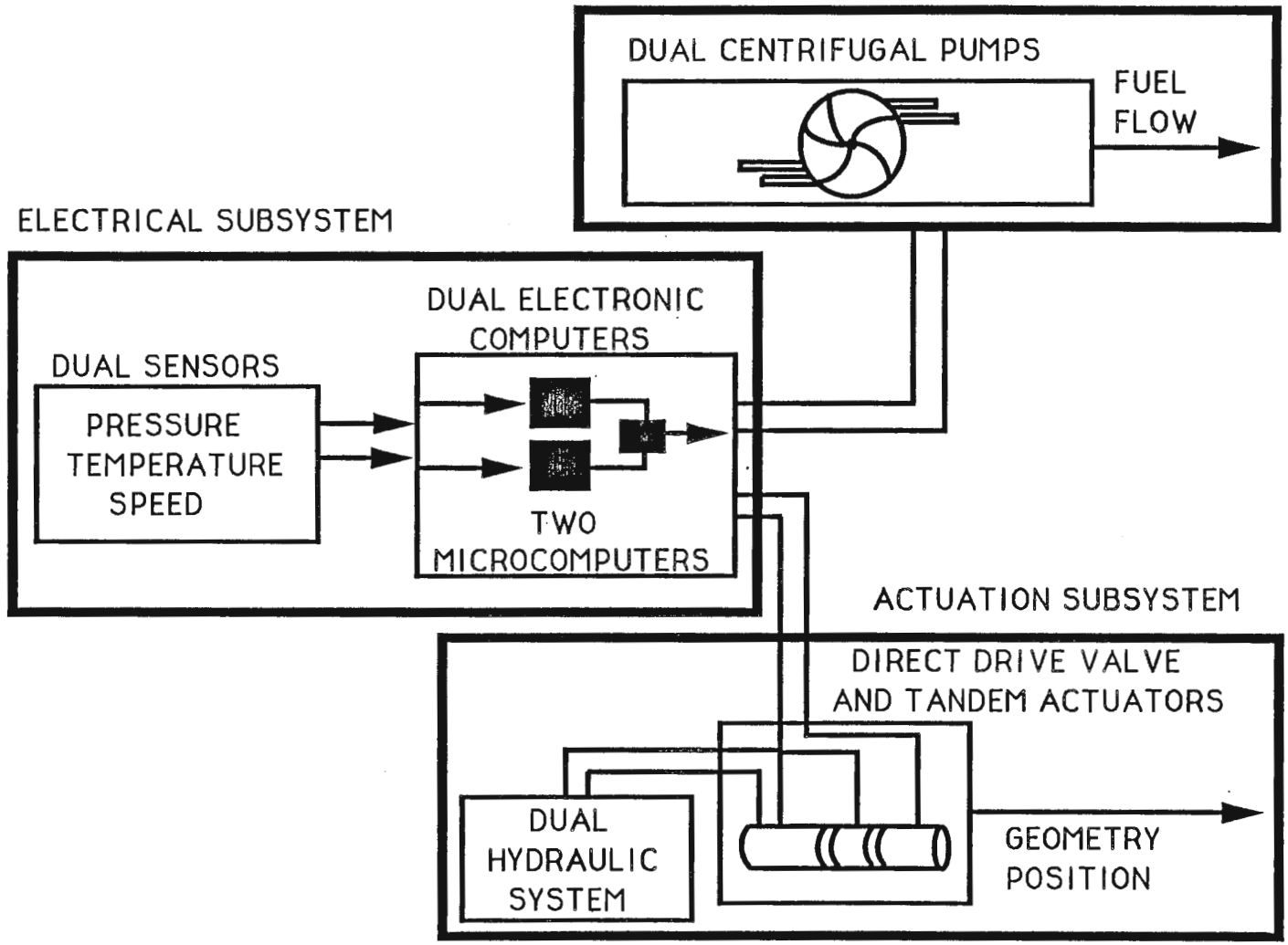



FIGURE 3.4-4. ADVANCED FUEL MANAGEMENT SYSTEM (TOTALLY REDUNDANT)

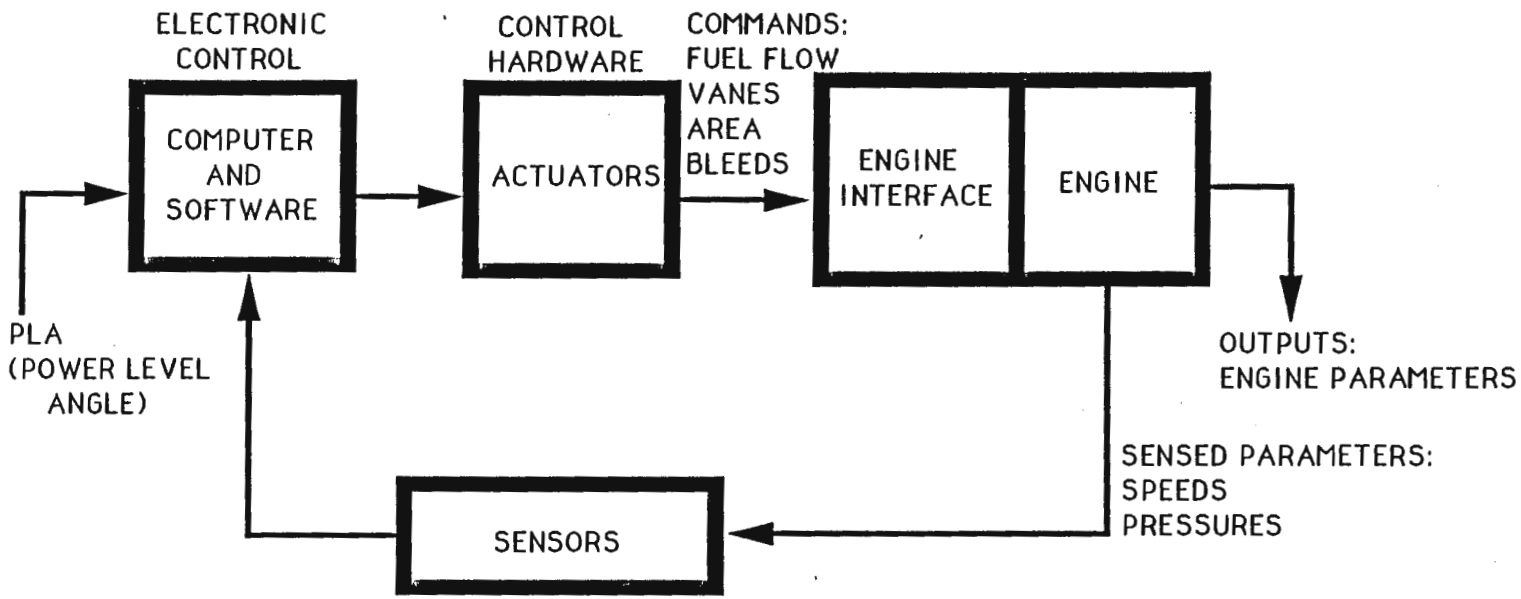



FIGURE 3.4-5. ADVANCED FUEL MANAGEMENT SYSTEM INTERFACES

- Unsteady dynamics at high angle of attack.

- Interaction of engine, inlets, and nozzles.

- Pilot-command/external-disturbance environment for advanced aircraft with state-of-the-art systems.

- Development of computer-assisted design techniques for efficient multi-input and multi-output control for a large number of control parameters. (Practical optimization algorithms for simplifying complex feedback mechanisms are also required.)

- Development of better hardware simulation methodology for building simulator interfaces that permit a more realistic test situation.

3.4.2.3. Advantages to be Gained from System Integration

System integration at the highest level will not be achieved without the development of efficient design and analysis tools. The GNC system will be the impetus for aircraft design integration. The key technology will be the multidisciplinary design of interactive aircraft systems that will meet both civil and military performance requirements for speed, range, maneuverability, navigation, efficiency, and safety. In the case of military aircraft, observability and threat avoidance will also be important. Inherent in this methodology will be the ability to resolve serious design conflicts across supporting technologies. When these multidisciplinary designs are fact, the following benefits are possible:

- Reductions in weight and drag by permitting compensation for unstable aerodynamic, structural, and propulsion conditions using a highly authoritative GNC system. Such factors as maneuver load control, gust load alleviation, active flutter suppression, and relaxed static stability would be involved.

- Providing new mission capability (e.g., low observability or super-maneuverability) by permitting compensation for unstable aerodynamic, structural, and propulsion conditions using a highly authoritative GNC system.

- Increasing capabilities for maneuvering, trimming, and STOL by integrating highly interactive subsystems such as vectored thrust and aerodynamic control.

- Maximizing capacity for military aircraft to conduct complex attack/defense missions by integrating fire, flight, and propulsion system controls.

Aircraft development will benefit from a reduction of costly, time-consuming iterations resulting from adverse interactions between systems and unrealized performance goals. Resultant aircraft systems should require less redesign and refinement over the life of the aircraft. These systems should result in less risk, cost, and time for developing new aircraft.

The multidisciplinary design is viewed as an enabling technology for advanced combat aircraft (fighters, multimission rotorcraft, and VTOL), advanced supersonic transports, hypersonic aircraft, and transatmospheric aircraft. The multidisciplinary design will result in improvements in subsonic aircraft.

# 4. RELATED VALIDATION ISSUES

## 4.1. Fiber Optic Technology

Fiber optics is a branch of optics concerned with the propagation of light along thin fibers. Each fiber consists of a core and sheath; the core and sheath are different transparent materials. In practice, more than 100,000 fibers that are a few microns in diameter have been grouped in a single bundle. Light entering one end of a fiber is transmitted the length of the fiber by internal reflections.

Ground-based telephone and data transmission systems, using twisted-pair and coaxial cable technologies, are now being replaced with fiber optic systems. The primary advantages of the fiber optic technology are the greater band-width (approximately 1,000 times higher), and smaller size. The application of fiber optics to aeronautics offers the advantages of lower cable weight, less space required by conduits, and less vulnerability to the nearness of power cables. In aircraft using composite structures, instead of traditional metallic, the advantage is even greater: less vulnerability to high-level RF interference from power cables, lightning, and other electromagnetic noise.

### 4.1.1. Fly-By-Light

Fly-By-Light is an outgrowth of FBW. In an FBW system, FCCs receive inputs in the form of electronic signals from numerous sources: sensors, avionic systems, pilot control, and maintenance input. In the case of pilot control or maintenance inputs, the FCC may initiate an adjustment to a control surface directly. The control laws for the aircraft exist as a portion of the software for the FCC. Normally, sensor and avionic system inputs are processed through the control law software, and a decision is made that some control surface adjustment is required. The FCC will then send an electronic signal to a control surface or to an actuator controlling a control surface resulting in an adjustment. Usually, heavily shielded coaxial cable is used for the data communications between the FCC, its input sources, and its output destinations. In an FBL system, an optical data bus or bundles of light-sensitive fibers would replace the coaxial cable. Figure 4.1-1 is a schematic of a typical FBL system.

Figure 4.1-1 is a simplified schematic of a system utilizing a single multi-processor control computer with a single channel. In reality there would be multiple systems and channels configured in a redundant architecture. The Primary Control System (PCS) consists of pilot controls, a processor, and actuators. The processor has an electrical power source. The communication between the processor and the pilot controls, and the processor and the actuators is by optical fiber bundles. The PCS is a flight-critical system

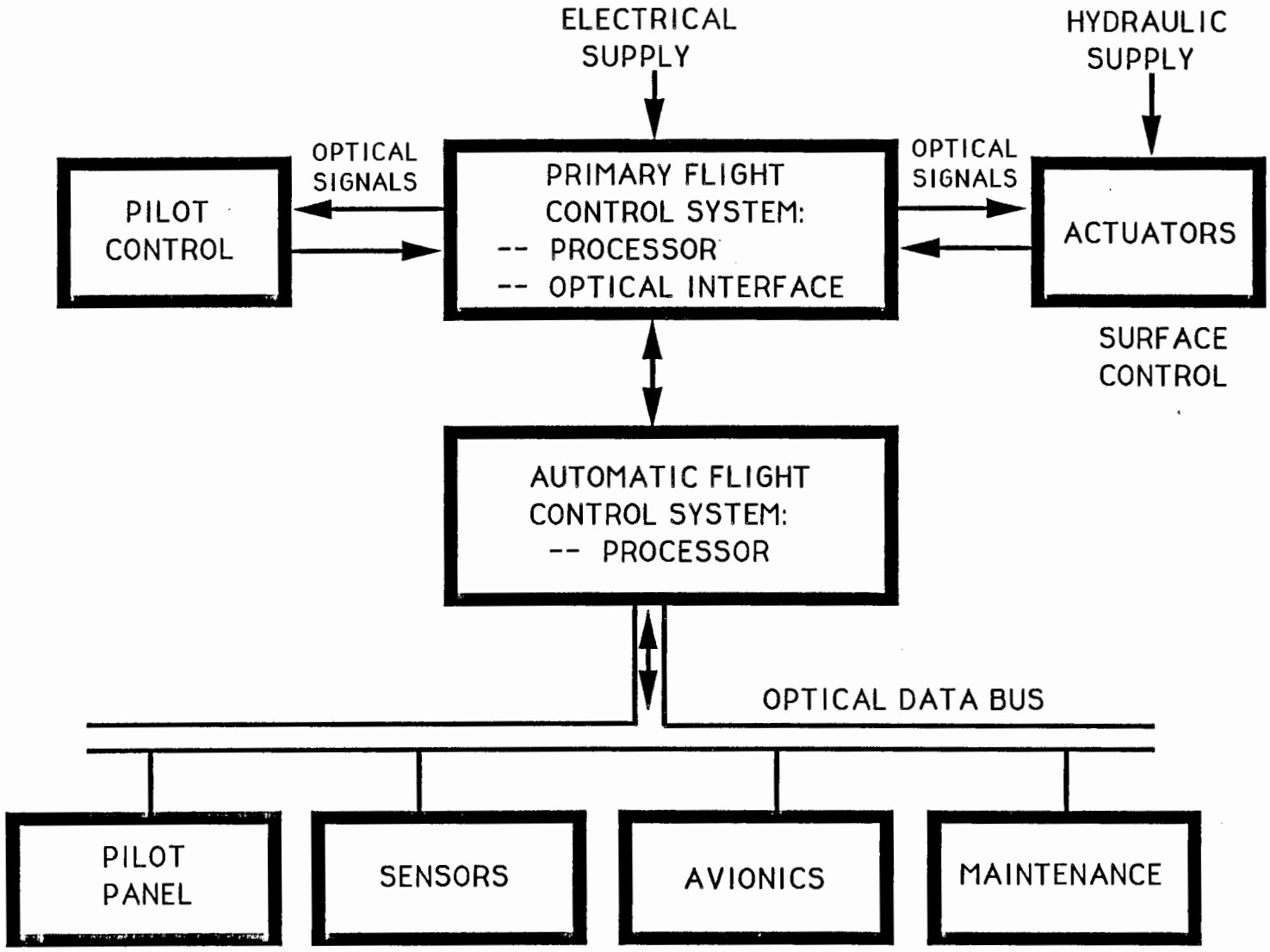



FIGURE 4.1-1. FLY-BY-LIGHT SCHEMATIC

and would therefore be heavily redundant. Flight safety reliability is required of the PCS. The actuators drive control surfaces, so they would most likely require a hydraulic supply.

The ACS consists of a processor, optical data bus, and the following interfacing systems: sensors, other avionic systems, pilot malfunction, and maintenance. The criticality of this system varies. If the aircraft has a high pilot workload or has a high degree of inherent instability, this system may be quite critical. If it is critical, it would involve a high degree of fault tolerance and redundancy. The ACS receives all necessary sensor data through an optical interface with the mission equipment bus in highly augmented/automated/integrated craft. The ACS software is mission-oriented and performs such functions as control law processing and selection of automatic modes. The ACS processor does not communicate with control surface actuators but accomplishes that purpose through the PCS processor.

A major function in a system similar to that defined in figure 4.1-1 is the conversion between optical, electrical, and hydraulic energy forms. An Electronic Interface Unit (EIU) would most likely be used to (1) interface with the processor through a Transistor-Transistor Logic (TTL) compatible buffer, (2) house a transmitter that performs electrical-to-optical conversions, (3) house a receiver that performs optical-to-electrical conversions, and (4) display diagnostic data. A fiber interconnect cable would connect the EIU to a transducer. A digital optical rotary position transducer may be used for pilot input. A digital optical linear position transducer may be used for actuator feedback. Digital optical differential hydraulic pressure transducers may be used to feedback differential pressure between cylinders of a hydraulic actuator or to prevent the force flight between cylinders with different pressures. Controlling an actuator with an optical command may require the conversion of the incoming command to an electrical signal; this signal may then control a conventional electrohydraulic servovalve. A solution to this problem may involve a hydraulic-oriented conversion or a photoelectric conversion. The photoelectric conversion, while seeming obvious, would probably require too much input power. A hydraulic-oriented conversion is probably most practical. This approach does not introduce the need for a new energy form since hydraulic power is already available in the actuator. A turbine-driven alternator or a piezoelectric crystal stack using hydraulic pressure for pulsing the stack ends are state-of-the-art approaches to this conversion.

### 4.1.2. Fiber Optic Sensors

Figure 4.1-2 is an application of fiber optic technology to a wing structure. This wing may contain a large number of sensors that would be subgrouped. Each subgroup would contain a number of binary coded sensors such as code plates, switches, and shaft speed sensors. The sensors of each subgroup would be multiplexed via a singlemode optical fiber to an optical receiver. The receiver would be a component of an EIU which would convert the binary encoded data to an electronic signal and buffer the data to a processor.

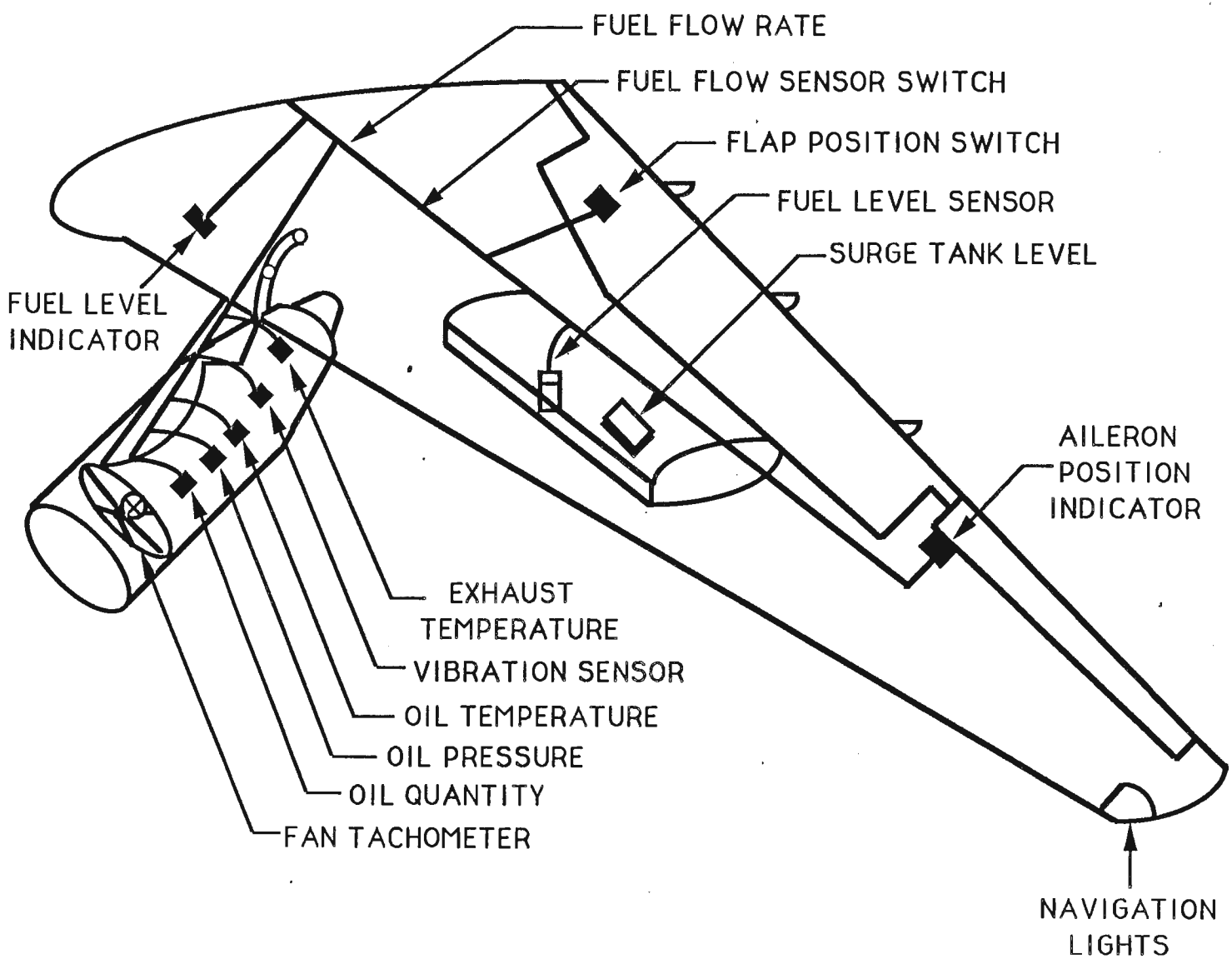



FIGURE 4.1-2. FIBER OPTICS IN WING ASSEMBLY

Due to the relatively large size of fiber optic connectors and the large number of sensors, it is necessary to multiplex data gathered from several sensors over a single fiber. A transducer using multimode fiber optic delay lines with a pulsed laser diode is frequently used as a multiplexor. This multiplexing scheme represents a traditional time division multiplex system. An advantage of this system is the use of a serial binary readout. Disadvantages are the system data losses and delay intervals inherent in time based multiplexing. Research on this type of multiplexor is documented (Farina, Hubbard, and Lefkowitz, 1983). Proposals have been made for a multiplexor utilizing coherence multiplexing. With this approach an encoder is used to read the position of an optical code plate or the switches in a parallel binary format. Each channel is delayed with respect to the others by use of fiber optic delay lines and the data combined onto a single fiber. A phase-carrier, coherent detection scheme with a limited coherence laser is used to multiplex the information onto a single fiber. The advantages of the coherence approach are shorter delay lines and reduced mode system losses. Research on coherence multiplexing is documented (Glomb, 1986).

Fiber optic sensors are increasingly being considered in aeronautical design. Fiber optic length monitors can measure strains (elongations and compression) from 0.1 micrometers to 1.0 millimeters. Fiber optic acoustic area monitors can detect structural damage. These developments compensate for some of the difficulties anticipated as composite materials are increasingly used in aircraft structures. Some of the research and development that justifies the use of fiber optic sensors in aeronautical structures is documented (Otaguro, Michal, and Watanabe, 1986).

Quality control for brittle materials has used proof testing methods and analytical formalism as techniques for validation. These techniques involve subjecting the structure to applied loads much larger than expected service loads for short time periods. Measurements of flaw size can be used to estimate structural life under related conditions. These techniques assume that in-service stress distribution conditions are identical to proof testing conditions. No allowance is made for in-service wear, damage, or age. Safety considerations are factored in by the process of structural over-design. Fiber optic sensors are essentially glass or plastic fiber waveguides with small encircling spools. These sensors can be used to detect low amplitude acoustical signals. While several techniques exist for detecting the occurrence of acoustical signals in the structures containing the sensors, the techniques rely on two basic concepts:

- Measuring the resulting cyclic change in velocity of the optical signals in the fibers of the sensor (interferometric method).

- Measuring the change in propagation intensity in the fiber due to acoustic signal mode mixing.

The interferometric method is used in passive sonar device technology, i.e., submarine detection. Similarly, the interferometric method can be adapted for detection of acoustic emission from strained composite structures.

The acoustic sensor system would be designed as a network of sensors and built into the aircraft's composite structure. The glass fibers would be glued to the internal surfaces of the composite structures. The networks would be connected by fiber communication links to a processor. The net-works would be the vehicle for sending the sensor signals to the processor. Time delay spectroscopy could identify the source of the signal to the processor; acoustic emission analysis would determine the development of flaws. A real-time evaluation of structural integrity would result from this system. The composite structure could be integrated with this "listening blanket" during proof-testing, and the proof-testing period could be extensive. The purpose of this would be to determine the suitability of the composite structure before major assembly. In addition to stress determination, fiber optic sensors can also be used, within certain constraints, as temperature and chemical detectors.

This basic technology is at hand. Fiber optic Sagnac and Mach-Zehnder interferometric sensors are being used to monitor strain. The integration of sensor development, acoustic emission analysis, and computerized evaluation of brittle material integrity is missing. With proper emphasis, an integration of control and structural technologies within the above guidelines could be accomplished early in the coming century.

## 4.2. Artificial Intelligence

Artificial Intelligence has been discussed earlier in this tutorial. In paragraph 2.4.1, AI was identified as one of the two elemental disciplines for developing avionic technologies. In paragraph 3.4.1.2 it was discussed as a catalyst for review of current aircraft certification procedures and regulations. AI is that branch of computer science which attempts to create intelligent computer systems that understand, reason, learn, and solve problems (Buckanin, 1984). AI can casually be categorized into three areas: expert systems, image processing and interpretation, and natural language communication. In paragraph 3.4.1.2, the expert system category was defined and examples were given. In this section emphasis will be on the other two categories. Voice Command Processing (VCP) is an example of the natural language communication aspect of AI. Machine Vision (MV) (See paragraph 4.2.2.1 for definition) is an example of the image processing and interpretation category. Both of these examples have potential application in tomorrow's aircraft. VCP is considered to have a wide range of application; MV is more restricted.

### 4.2.1. Voice Command Processing

VCP will have a major impact on how the pilot of the future commands and controls his aircraft. However, the integration of this technology with a future cockpit poses many difficulties, both in the hardware and system interface areas.

#### 4.2.1.1. The Need for Voice Command Processing

As the complexity of avionic systems increases, so does the workload on the pilot. This is a concern to both the civilian and military aspects of aerospace. The concern is most acute for today's high performance military aircraft that must perform single-seat tactical missions. The hands busy/eyes busy

mission pushes the limits for both pilot and aircraft. VCP is part of the solution to this problem; it should reduce the button and switch effort significantly. Developments in recognizer/synthesizer hardware and in recognition algorithms have made this technology one that should provide near-term benefits.

Boeing's entry in the next generation of subsonic transports, the Boeing 7J7, will utilize a voice recognition system. The USAF is testing voice activated electronics for its C-135C transport and its advanced F-16 fighter. These systems will respond to inquiries regarding fuel level, weapons status, and navigation; they will be responsive to numerous commands, e.g., changing radio channels and target proximity. There are over two-hundred basic information arrangements (pages) in the F-16C; each of these has numerous permutations and combinations. ATF will have hundreds of pages. If a conventional display approach is taken in the ATF, the pilot-aircraft interface will be impossible. Speech and hearing are two highly efficient forms of communicating. Systems oriented to their use must be developed for command and control.

4.2.1.2. The Problems for Voice Command Processing

Speech recognition is the process of correctly translating human speech into digital data. Speech synthesis is the reverse procedure. The latter is a mature technology: digital images of known responses can be stored and used when needed. Speech recognition is less developed. The sound of a given word may vary for the following reasons:

- Voice quality and attributes of the speaker.
- Context in which word is used.
- Inconsistent separators between words.

For a pilot, the sound of a given word may also vary for these reasons:

- Stressful conditions.
- Noise of aircraft.
- Presence of an oxygen mask.
- Altered g forces.

This difficulty is minimized by (1) requiring that speech be in isolated words, (2) limiting the acceptable vocabulary, and (3) programming the computer to recognize a single user. The imposition of the constraint that only isolated words be used will be accomplished with the definition of the command language. The pilot must be thoroughly trained in this language, and some type of video prompt should be present in the cockpit. The acceptable vocabulary is being limited. It is estimated that a vocabulary of between 200 and 500 words will suffice for even the most sophisticated systems. Programming the computer to recognize the pilot's pronunciations will be discussed later.

4.2.1.3. A Procedure for Voice Recognition

Figure 4.2-1 is a schematic for the voice recognition process. The audio input drives this process. There are three steps in voice recognition: extracting words out of the audio input, pattern recognition, and context analysis. The first step is that of extracting words out of the audio input. This consists of (1) converting the audio input to a spectrum and (2) extracting the spectral data. The conversion is normally handled by a Texas Instruments special purpose chip TMS320. This chip is a very fast 32-bit microprocessor that develops an energy distribution of the spoken word(s) by time. The extraction may be handled by such techniques as linear predictive coding, fast Fourier transforms, or digital spectral analysis. Linear predictive coding performs a polynomial fit on the sound spectrum; fast Fourier transforms develop frequency spectrums; and digital spectral analysis results in a series of attributes that characterize the primary frequencies of speech during each segment of time.

The second step is pattern recognition. This step utilizes the output of the first step and a library of relevant words. Depending on the technique used in the first step, one of the following will be produced:

- A set of coefficients representing the waveform.
- A frequency distribution of the waveform.
- A set of attributes of the waveform.

The library of relevant words is encoded in the same format as the output of step one. Assuming that digital spectral analysis is the extraction technique, the output of the first step may consist of attributes such as, the frequencies of each major frequency band or peak in the spectrum. These frequencies are referred to as formants. The library would contain the formants for different relevant words; there would probably be multiple formants per relevant word. Each formant is used as a "template" and is passed over the incoming sound message (output from the first step). The analysis continues until as much as possible of the incoming sound message is identified.

The third step is context analysis. Once the identification of individual words or phrases, is complete, the intent of the entire message is determined. This may be accomplished by graph analysis or encoded tables. The context analysis may be very efficient and permit only known words, or maybe very user-friendly and permit unknown words. Some known words may even be optional. The output of this step is the recognized pilot command. While not a function of voice recognition, the next step is interpretive logic that will obey the command.

The library of relevant words is developed during the programming process. This is a special mode of operation to the VCP. In this mode the pilot will identify a word to the VCP. The pilot will then pronounce the word under different circumstances. The system will then be tested under actual flight conditions. If there are pilot messages that the system does not recognize, the training mode will be reentered in an attempt to reproduce these conditions. Usually, a number of iterations through the training and actual flight cycle will be

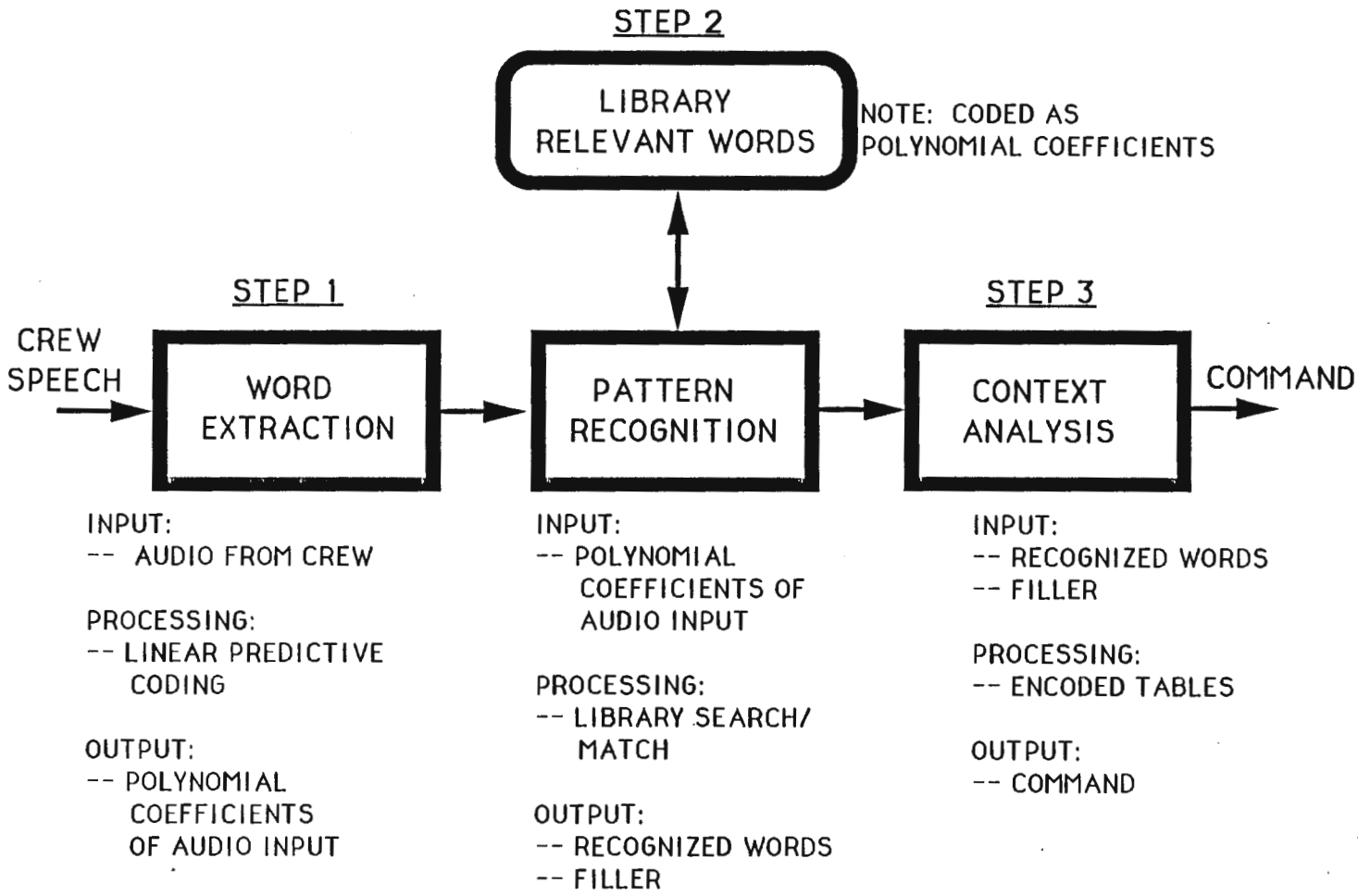



FIGURE 4.2-1. VOICE RECOGNITION SYSTEM

required. It is possible to have multiple libraries for a given aircraft, one for each potential pilot.

4.2.2. How Machine Vision is Accomplished

Machine Vision is used in the early stage (analysis stage) of research and development. It commences with the digitizing of the subject's image. An area is scanned by conventional or structured light beams. A charge-coupled camera or TV camera converts the image of the area being scanned into an array of picture elements (pixels). The intensity of light at each pixel is encoded as a binary number; this number represents a tone on a gray scale. Following the scanning and encoding, the array of pixels must be analyzed; it is at this point that most procedures for vision processing differ.

If the application is simple, such as the inspection of pieces, the pixel array is organized into dark and light segments. This process is called "thresh-holding". From this process, edges will be determined and shapes outlined. A library of standard shapes is used to determine the identity of the subject scanned. If microcircuits are being inspected, the library of shapes will be oriented to connections between circuit lines. If the scanned subject has edges that produce another topology, it may have flaws or breaks.

The analysis stage may be very simple, repetitive calculations on pixels or groups of pixels. Computers with simple Von Neuman architectures may find this an exhaustive job when the pixel density is high and/or the area scanned is large. This is an ideal application for parallel processors. Vision processing will encourage the development of parallel processing computer architectures.

Inspection is a simple example of MV. The hardware would consist of the camera system and a processor. Robotic vision, while having more hardware, is not necessarily more complex. A common application is locating an object that the robot will act on. For example, the vision system may find a particular spot on a circuit board where the robot is to insert a component. In this case, simple silhouettes may be sufficient for a program to identify a part by area, length, number of holes, or other features.

Surface curvature poses a problem to vision systems. Curvature may cause variances in reflection that are normal; this makes identification more difficult. An algorithm has been developed to handle this problem. The variations in light intensity are determined with scans moving in different directions. Reflection intensities resulting from surface curvature will vary from scan to scan; reflection intensities from flaws will not vary. When robotic vision requires great precision, 3-D vision may be required. A procedure used for 3-D vision is to utilize a single camera for illumination of the scanning beam and an additional camera(s) for a triangulation at the scanning spot. Robots have drilled holes in parts to a precision of 0.005 inches using this technique.

4.2.2.1. Machine Vision in Aircraft

Because the flight environment for civil aircraft is reasonably structured, these aircraft have ways of "seeing" that do not conform to traditional concepts

of vision. For tactical military aircraft, the flight environment is not so structured. Most of the aeronautical research and development in MV is for military aircraft. The DOD is currently studying the problem of applying MV to target recognition and navigation. One project calls for the recognition of a target by an aircraft or incoming missile; another project calls for the cross-country navigation of a high speed vehicle. Success in research as ambitious as this may be more than a decade away. On a simpler level, MV is successfully being used in factories to ensure the quality of manufactured electronics components. It is also being utilized to improve the quality of robotic welding, drilling, and cutting.

4.2.2.2. Two Complex Applications of Machine Vision

Target recognition is an advanced application for MV. The major problem is that of distinguishing between a tank and a truck when either vehicle may be viewed from any perspective (angle). The fact that the background for the subject being identified is undefined complicates the problem. Pattern recognition algorithms are being applied to this problem. In general, pattern recognition is to have a predefined image of the subject(s) for which identification is desired. This predefined image is then superimposed throughout the scanned area to determine whether or not it is present. For a single item, it would be necessary to superimpose the images from all perspectives throughout the scanned area. Other difficulties result from the following:

- The predefined images will vary as the elevation and distance of the scanning camera changes.
- The basic subject may not be fixed; e.g., the tank may have its gun pointed in any direction or the truck may be loaded/unloaded.

Synthetic discriminant algorithms are also being considered for target recognition. With this approach, a subject is defined to be a combination of a number of abstract geometric objects. The tank may be identified by the treads. This alleviates the problem of concern for orientation, but intensifies the problem of isolating the object from the background.

Another complex application is ground vehicle navigation. In this research, a vehicle with MV is to be driven over a winding road. The processing commences with a gray scale image of the road. Edges in the image that were parallel and approximately the same distance apart as the preceding image were taken as the road edges. Shadows and trees pose major problems. Color cameras reduce this problem as light intensity is used as a parameter for isolating the road from the remainder of the scanned area.

Obstacle avoidance is another facet of the navigation problem. Since this involves a distance dimension, a 3-D approach using lasers, sonar, or camera triangulation is being attempted. The success of this effort varies as to the ability of the obstacle to reflect sound. In addition to using a single detector (laser, sonar, or camera), research is underway to utilize all three sensors and have a master program compare the data submitted by the processor controlling each sensor (vote-taking). This data is coordinated with background data (images supplied by a satellite) to create the final computer image.

Progress in both target recognition and navigation is slow; however, MV is being used in manufacturing and domestic applications.

## 4.3. Computer Technology

Aircraft have become increasingly dependent on computers for both performance and safety improvements. This technology has grown by a factor of 10 every 5 years for the past 35 years; continued growth is anticipated. This rapid growth has primarily been the result of mammoth breakthroughs in device technology: vacuum tube to transistors to integrated circuits. Numerical software has also been greatly improved.

### 4.3.1. Advances in Microelectronic and Component Technologies

Since advances in these technologies have been a root cause of the evolution of today's digital avionics hardware, this subject has been alluded to in earlier sections of this tutorial. By the early 1990's it is anticipated that microelectronics will achieve today's recognized limits for Very-Large-Scale Integrated Circuits (VLSIC). Currently, Field Effect Transistor (FET) geometry cannot be scaled below approximately 0.25 micron. The reduction in the feature size of microelectronic devices will level off before reaching this limit. New techniques must be developed. As mentioned in section 2.4.5, 3-D architectures are a potential solution. These architectures would permit vertical as well as horizontal element interconnections. If problems such as heat removal can be solved, this technology may help sustain the reduction in feature size. An alternative to vertical interconnections is a reduction in the number of interconnections. This can be accomplished by developing modular digital function devices that accomplish more complex functions. More complex function devices is a second potential solution. A third possible solution is a new digital integrated circuit technology that uses submicron structures and new physical principles to solve the interconnect problem.

On the component level, rapid advances in processing and storage device technology are expected to continue. Logic and memory component density is expected to continue to increase by a factor of ten every five years for the next decade. As feature size converges to the 0.25 micron range, manufacturing chip cost will level off and, perhaps, increase. The cost per unit capability should continue to decline because of the increased functionality of the chip. Mass storage density has increased by a factor of four every five years. Current optical storage technology increases storage by a factor of 10 over current magnetic storage per unit volume. The utilization of video storage is also being studied for its adaptability to digital systems.

The "splitting" problem of optical bus systems is being researched. Optical bus systems that support in excess of 64 taps per kilometer are envisioned. In paragraph 4.1.2 of this tutorial, frequency division multiplexing techniques for optical fibers (permitting up to 50 channels per optical fiber) were discussed. Each channel has a bandwidth of 0.5 to 1 gigabit per second.

Single devices housing solid-state sensors, processors, and memory are anticipated within the next decade. As mentioned earlier, new chip-level and

wafer-level interconnection technologies will be used to alleviate the component interconnection problem. Holography is an example of a new interconnection technology.

4.3.2. Advances in Parallel Processor Organization

The traditional processor architecture, the von Neumann architecture, is characterized by sequential operation. The processor's cycle time is the limiting factor on performance. The cycle time is bounded by the propagation time of signals. The propagation time of signals inhibits clock periods shorter than one nanosecond. Theoretically, the maximum achievable performance is one billion ($1.0^{+9}$) instructions per second. A parallel architecture is required if this limit is to be significantly passed. By the year 2000, effective parallel architectures consisting of thousands of processors, high bandwidth communications, and parallel structured software is anticipated.

Parallel processing is simple. In sequential processing, all data is accessed from a central data store. A processor that is working on several applications will spend most of its time retrieving data instead of calculating. In paragraph 4.2.2.1 of this tutorial, MV was seen to be an application that consisted of many disjoint tasks. This application is very conducive to implementation in a parallel environment. Each task would be performed on a subprocessor with its own memory.

If a microprocessor can perform thousands or tens of thousands of operations per second, a parallel machine with thousands of microprocessors could perform millions or tens of millions of operations per second. If the microprocessor costs $50, a parallel architecture of 1000 microprocessors would cost $50,000. A supercomputer with similar capacity would cost $10 million.

There are three major difficult steps to implementing parallel processor architectures. The first step is to develop the capability for a meaningful, efficient communication between many processors. This includes the responsibility of controlling several processors working on different tasks on the same problem. The second step is to simplify programming in a parallel environment. The third step is to develop parallel-oriented solutions to problems. The following two approaches for resolving the difficulties of these steps are being studied:

- Utilizing a single instruction/multiple data stream architecture (one sequence of instructions carried out in parallel on data).

- Utilizing a multiple instruction/multiple data stream architecture (many sequences of instructions carried out in parallel on data).

The multiple instruction/multiple data stream architecture is the purest form of parallel processing but offers the greatest challenge. Within these two broad groupings, parallel architectures can be subgrouped by grain size, degree of connection, shared memory, and the technique for decomposing a problem into subproblems that may be solved in parallel. Grain size is defined to be the degree of parallelism. An architecture with a few large processors (up to one hundred) has a large grain. An architecture with thousands of small processors

has a small grain. The degree of connection relates to the interconnectivity of the processors. An architecture that has all processors communicating with all other processors has a high degree of connection. If processors can only communicate with a small, restricted set of processors, a low degree of connection exists. The shared memory attribute relates to whether all processors share a single memory store or whether each processor has its own memory store. The problem decomposition attribute relates to whether the machine decomposes the problem into tasks that may be executed in parallel or whether a programmer must do this.

NASA's Massively Parallel Processor (MPP) is an example of the first generation of parallel architectures. The MPP was developed by Goodyear Aero-space. The MPP is a small grain machine: 16,268 processors arranged in a 128 X 128 square. It has a low degree of connectivity. Each processor is connected to its four nearest neighbors. Every processor has its own 1,024 bits of memory. Problem decomposition is the responsibility of a programmer. A processor consists of a logic unit, an arithmetic unit, 35 shift registers, and the memory. A processor works on a single bit of data per executed instruction. All processors simultaneously could process 16,000 bits of data per cycle; a conventional mainframe processes 64 bits per cycle. Since each processor can execute 10 million instructions per second, the MPP maximum computing rate is 200 Million floating-point operations per second (Mflops). This rate is comparable to the Cray supercomputer.

The MPP is primarily performing signal and image processing. These applications are conducive to the nearest-neighbor connection scheme. The MPP outperformed the Cray by a factor of five. In the area of physics, plasma and gravitational field simulations are being attempted. These simulations are not as conducive to the nearest-neighbor connection scheme as are aerodynamic simulations such as image and signal processing.

NASA-Ames and Yale University have received two iPSC parallel machines. The iPSC was developed by Intel. The iPSC contains 128 processors, and would, therefore, be classified as a large grain machine. Each processor has over 512 kilobytes of memory. A hypercube design was used that interconnects each processor to seven neighbors. Thirty-two and sixty-four processor models of the iPSC permit each processor to connect with five and six neighbors, respectively. Problem decomposition is again the responsibility of a programmer.

Programming a parallel processing computer is very difficult. The four approaches being studied as potential solutions to this problem are as follows:

- Developing functional programming languages that automatically translate into a parallel algorithm.
- Developing process definition languages that permit a machine to dynamically allocate resources.
- Retrofitting parallel-oriented statements into a traditional algorithmic language like FORTRAN.

- Developing tools that permit a programmer to allocate resources in a simple manner.

A dataflow approach is the partitioning of a problem into a sequence of serial and parallel tasks which may be performed by separate processors. Dataflow processors are dedicated processors that allocate these tasks to their subprocessors. A program incorporates certain statements that define the route that data must follow and the transformations that occur to that data enroute. Using this program, the dataflow processor then assigns work to sets of processors. A nondataflow processor will only perform the work assigned to it.

### 4.3.3. Advances in Distributed Processor Organization

Distributed processor organization involves the arrangement of multiple processors into a network interconnected by some telecommunications medium. The growth of distributed computer system architectures has parallelled the growth in microelectronic technology. The result of microelectronic technological advances increased the importance of small computers. A network of small computers with a distributed workload could perform certain applications more reliably, efficiently, and economically than a single large computer with point-to-point communication to terminal devices. FCS's can be developed using distributed architectures. Paragraph 4.1.1 of this tutorial discusses the components of a FCS. The following discussion uses the FCS as an example of a distributed computer system. The architecture presented here is based upon the Bendix Multicomputer Architecture for Fault Tolerance (MAFT). "Fault-Tolerance in Distributed Digital FBW FCSs" (Gluch and Paul, 1986) covers this subject in great detail.

Figure 4.3-1 is a schematic of a FCS. From this schematic it is obvious that multiple processors are interconnected by a telecommunications medium, a trunk, or bus. The Primary Flight Computer, Secondary Flight Computer, Actuator Control Electronics, Data Sensor Units, and Autoflight Computer are all intelligent devices. They are all microprocessor-based units including memory, data store, and software. The bus may be electrical or fiber optic. Competition for the bus between intelligent units must be regulated by a protocol. A Carrier Sensed Multiple Access (CSMA) or token passing protocol may be used. With the CSMA, each of the nodes on the bus has a guaranteed window within a predetermined Access Time Interval (ATI). A node is a connection from an intelligent unit to the bus. In order for the FCS to perform adequately, the ATI must be small, perhaps on the order of 10 milliseconds. A token passing protocol requires that a token constantly circulate on the bus; only the node with the token may transmit.

Three types of data sensor units represent nodes on the bus. The pilot interface unit receives analog inputs from pilot controls, converts these to digital signals, constructs the appropriate communication layers, and places the resultant message on the bus during the alloted ATI. The appropriate actuator control electronics unit will take the message off the bus and act on it. Other nodes, such as the primary and secondary flight control computers, may also receive copies of this message, but for other purposes. An appropriate response may be returned to the originator and/or other interested parties by

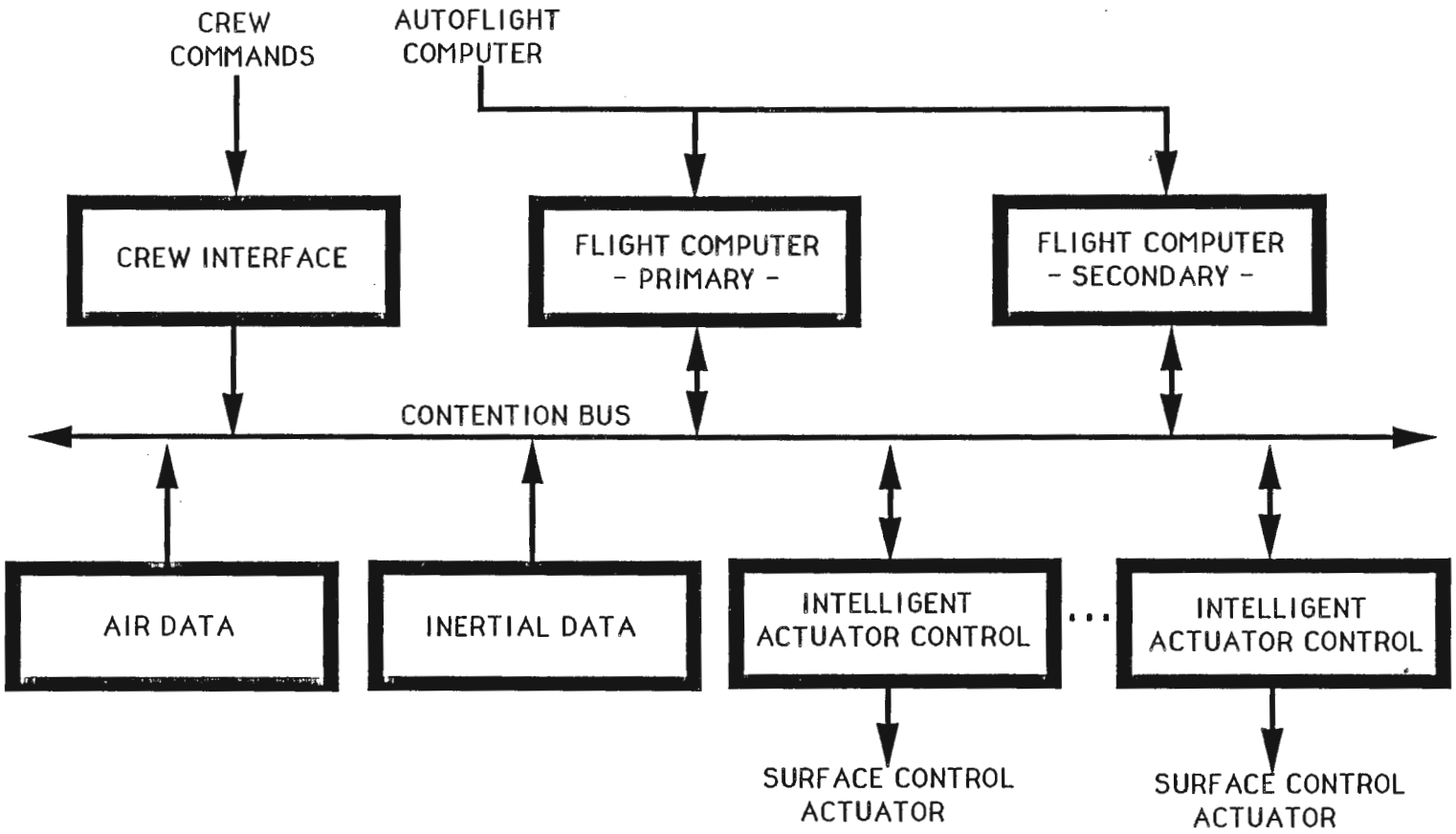



FIGURE 4.3-1. DISTRIBUTED PROCESSOR FLIGHT CONTROL SYSTEM

any recipient. In a similar manner, air data and inertial data sensors will broadcast data across the bus during their ATI. This data will be received by the autoflight computer, the primary flight computer, and the actuator control electronics units. The autoflight computer is responsible for determining whether flight sensors are recording flight parameters that meet projections. If not, surface control commands may be issued to correct the situation.

Actuation control of the surfaces and interface to the bus is accomplished by the actuator control electronics units. These intelligent units may interface to either semi-electromechanical or hydraulic actuation systems. These units perform closed loop control, low frequency equalization, fault detection, and reconfiguration services for each of the control surfaces.

The architecture depicted here represents a highly distributed approach to the implementation of a digital FBW system. Intelligent processing units located throughout the aircraft work collectively through communications provided by a multi-access bidirectional bus. While the overall system functions are distributed, the hardware and software of the primary flight computer is the central ultra-reliable LRU in the system. Reliability concerns dominate the system. The probability of system failure must be virtually zero. Multiple hardware redundancy is employed. This redundancy involves multiple communication busses, sensors, flight control processors, and actuators. Additionally, there will be redundancy in all software modules. All intelligent systems will incorporate fault-tolerant logic.

### 4.4. Advances in Control and Display Technologies

As the number and complexity of avionic systems grow, it becomes increasingly difficult for the aircraft crew to assume all past responsibilities, as well as the workload resulting from the new systems. With computer-driven surface controls providing stable flight in inherently aerodynamically unstable aircraft, the pilot is no longer the operator of the aircraft, but is rapidly becoming a flight monitor/manager who will intervene in the event of system failure. There are two directions that future systems may take. First, systems may be designed that would augment the operator's information acquisition and processing capabilities. Second, the systems may be designed to eliminate the need for operator intervention on the flight deck.

#### 4.4.1. Improved Man-Machine Interface

The explosion in numbers of avionic systems has resulted in extremely large volumes of data being present on the aircraft computers. The crew may not view all of this data; however, sufficient data to permit the crew's full awareness of the aircraft's situation is important. Information from onboard sensors, stored knowledge, and observation/interaction of the crew must be digested and presented in a manner that is not overwhelming. The crew must focus on the decision process, strategy development, and strategy implementation.

##### 4.4.1.1. Short Range Cockpit Improvement

The A320 cockpit bears limited resemblance to its predecessors, but one that may be a future trend setter. Roll and pitch controls are allocated to "side

sticks" that are installed on lateral consoles. The conventional control column is no longer present. In conventional aircraft the control column function required large mechanical assemblies that occupied much of the subfloor compartment. The sidestick controller uses analog displacement sensors to permit large grip motion, instead of force sensors that would restrict motion. The sidestick controller includes roll and pitch sensor boxes and centering "feels". A solenoid-operated detent locks the control lever at neutral when in autopilot mode. The stick does not respond to autopilot commands. The canted hand grip has maximum angular deflections of 16 degrees in pitch and 20 degrees in roll. The sensors detect displacements of the hand grip; this results in the transmission of analog signals to the FCS computers. All roll and pitch control is under the guidance of the fully electronic FCS.

Minimal mechanical backup is provided by a mechanically controlled rudder and standby tailplane trim. The basic yaw control uses a single rudder controlled from the cockpit pedals. A trim switch is located at the rear of the central pedestal and controls rudder trim.

Another recent innovation in today's cockpit is the presence of CRT displays. Usually, one set of CRTs displays traditional flight control mode information, as well as consolidations of artificial horizon, horizontal situation indicator, and airdata instrumentation. Another CRT displays integrated plan and profile views of the projected flight plan.

Smart flat panel displays are being considered as potential replacements for CRTs. Examples of the technologies being considered are gas plasma, liquid crystal, and thin film electroluminescence. The Liquid Crystal Display (LCD) is getting particular attention because of its low power and low voltage requirements. It has the disadvantages of poor temperature performance, low dynamic range, and limited viewing angle; however, these problems are solvable. Research with thin-film transistor LCD matrix devices has established the LCD technology as the most likely successor to the CRT.

#### 4.4.1.2. Long Range Cockpit Improvement

Whenever a new avionic device is placed in the cockpit, more visual workload is placed on the pilot. Researchers working on the virtual panorama display, a precursor of the USAF Supercockpit Project, are attempting to estimate the workload produced by a new display before the display appears in a cockpit. This involves creating semantic networks from the information content in an image. A path is created in the network connecting a certain piece of information in the image with information known to the pilot. The number of semantic nodes that must be traversed are used as a measure of the workload placed on the pilot by the image. This technique has been tried in simulated missions, and the simulation confirmed the technique. The more complex the semantic network rating, the greater the deterioration in pilot performance in the simulated mission.

In the supercockpit the pilot will wear a helmet weighing approximately three and one-half pounds. Two tiny video screens will be mounted in front of the pilot's eyes. These screens will enable the pilot to view avionics displays, as well as magnified and illuminated real-world scenes. The data to be viewed

will be selected by the pilot, or the FCS will display data on an exception basis.

In the supercockpit, the screens will be translucent, so the images will be projected on the real world. A flight simulator helmet from which the supercockpit helmet will evolve utilizes two tiny CRTs; each CRT projects a scene one thousand pixels wide and one thousand pixels high, each pixel has a width of approximately twelve microns. The concept of the translucent display, instead of the CRT, is being tried in the Army's developmental LHX helicopter. There is concern over pilot confusion resulting from superimposing the display images on the real world. Research has shown that simulated images cause little confusion; whereas, actual images (such as magnifications) may cause considerable confusion. A 14-bit precision system is used to detect the orientation of the pilot's head. If the pilot turns his head, the display image will rotate with his head in relatively the same position. The system uses a set of three orthogonal radio-frequency coils mounted in the cockpit and three tiny magnetic detectors mounted in the helmet. Signals from the detectors are used to determine the exact position of the helmet. Special purpose electronics were developed to compensate for distortions in the optical system. These electronics result in (1) the blending of edges when there are two overlapping optical fields, and (2) the creating of the effect of tunnel vision.

If the scene becomes overly complicated, some of the data may be fed into the pilot's ear. The speakers would be located in the pilot's helmet. Locations of nearby aircraft may be indicated by sounds located in virtual or apparent space around the pilot. Techniques for localizing sound are primitive. Stereo indicates the lateral location of sound (left or right), but doesn't localize it in the two remaining dimensions, up/down or front/back. The cues used in stereo (timing, phase shift, and loudness difference between the ears) are important, but are incomplete. Paragraph 4.2.1 of this tutorial discussed VCP. Audio sensors for picking up the pilots' speech would also be present in the helmet.

Pilots can be overloaded with sounds, as well as sights. With VCP the voice can be used to control the aircraft. The sense of touch can also be used. In the cockpit of the future, the pilot may be able to reach out and touch a "virtual switch" to activate a sensor. The switch may be an image from his helmet. This may be accomplished through fiber optic technology or by having a pneumatic device built into a special glove. Far more sophisticated pneumatic arrays using piezoelectric materials may be able to simulate the touch of any object.

#### 4.4.1.3. Pilot Associate Program

Developments in AI will have a tremendous impact on future cockpits. AI Implications for Advanced Pilot/Vehicle Interface (PVI) Design (Maxwell and Davis, 1985) is an excellent source on the methods and extents to which AI will effect the PVI in future generation aircraft. Avionic systems are proliferating in today's aircraft. This has been particularly a problem in military tactical aircraft with flight crews of one or two persons. In this case, flight crews are facing rapidly increasing workloads which must be accommodated under stressful conditions. As avionic systems increase in commercial and general

aviation, the same effect will be felt for aircraft in these classes. The amount of data to be analyzed and pressures to reduce crew costs will increasingly reduce the role of the human crew in data analysis and decision-making. The PVI is that portion of the cockpit that presents information to the flight crew for their analysis, and receives control information from the crew. As the amount of aircraft data and control requirements have increased, PVIs have been designed to compensate in several ways.

First, multifunctional displays have been designed that do not display all data but only the data that is functionally relevant. Second, integrated displays are designed to provide cross-system data that is functionally related. Third, automated systems are designed that operate quasi-autonomously from the crew; by doing this, the crew needs only to monitor systems intermittently instead of continuously controlling them. Fourth, multifunctional systems are designed that permit crew controls at a high level; they do not have to give attention to subsystem details. The effect of each of these should be to reduce the workload on the crew.

Artificial Intelligence is a technology that offers the possibility of massive improvements in this area. The DOD Pilot Associate program is studying the feasibility of having a computer responsible for the routine tasks of flying, and acting as a copilot. Additionally, the pilot's brain waves and heart muscle electrical rhythms are monitored to determine "black out". If a pilot blacks out, the computer will completely take over flight of the aircraft. Figure 4.4-1 depicts a PVI utilizing an AI computer system. There are three relationships that may exist between a pilot and an AI system. AI Implications for Advanced PVI Design (Maxwell and Davis, 1985) discusses these relationships in detail.

The first relationship is the Pilot Manager/AI Associate model. In this model the pilot is responsible for those decisions at the outer control loop level. The pilot performs a systems manager function; the AI system responds to the directions of the pilot. Decisions in the inner control loop are the responsibility of the AI Associate. This model is acceptable in operational situations where tasks may be partitioned vertically. The pilot performs tasks at the upper end, while the AI Associate performs those at the lower end.

The second relationship is the Pilot/AI Colleague model. In this model the pilot works with the AI Colleague on the basis of equality and shared responsibility. Each oversees the actions of the other and contributes to the performance of virtually every task. This model is best suited to operational situations in which a vertical partitioning is either impossible or inefficient.

The third relationship is the Autonomous Assistant model. In this model the computer may actually assume control of the aircraft, and the pilot will be subordinate to the AI system. This relationship is desirable if the pilot becomes incapacitated to the extent that the pilot cannot function in one of the other two relationships.

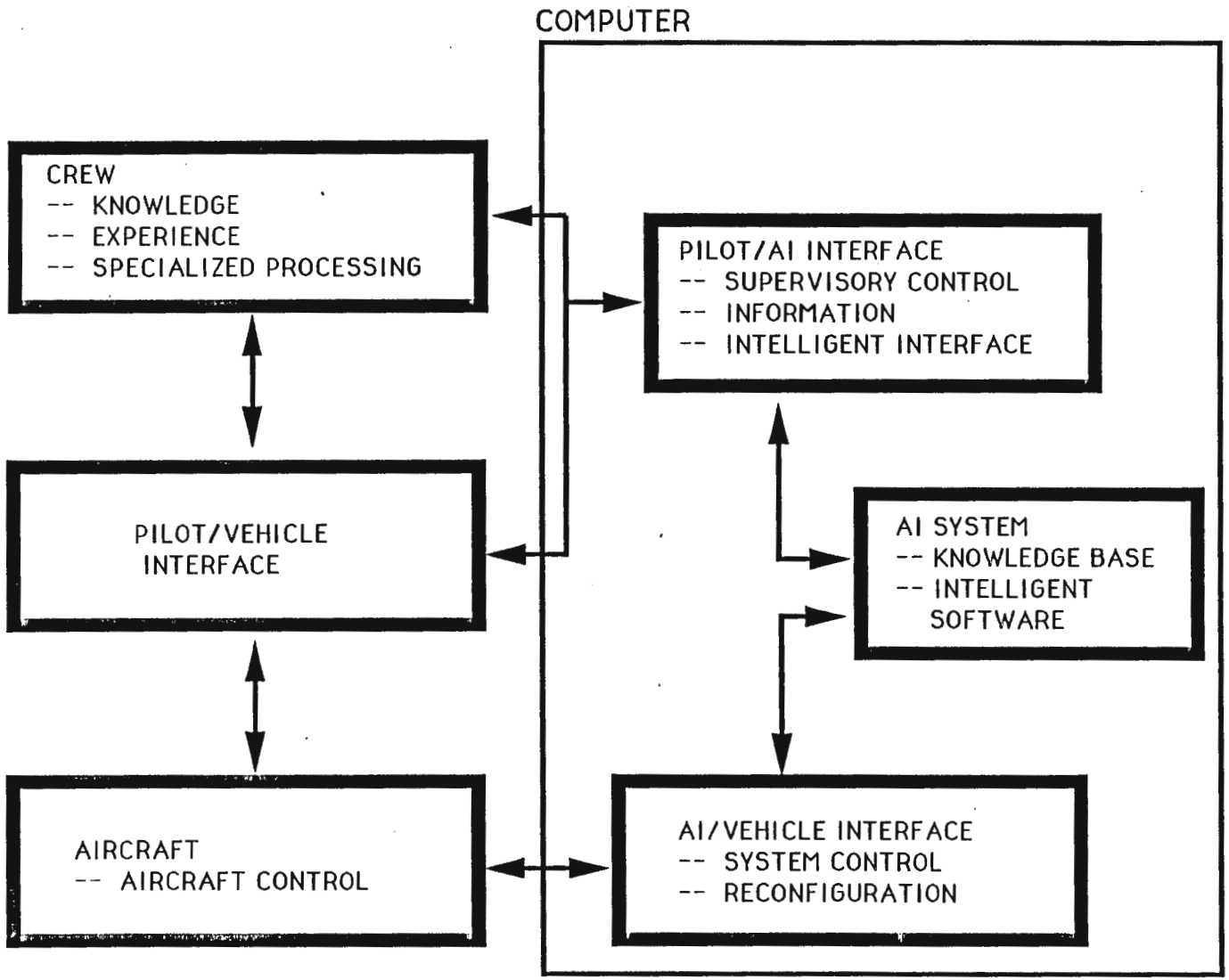



FIGURE 4.4-1. PILOT/VEHICLE INTERFACE WITH ARTIFICIAL INTELLIGENCE SYSTEM

These three relationships are not mutually exclusive. A single PVI may support all three relationships. During a single mission, each of the models may be used when operational situations dictate. The particular blend of the above three models into the design of a single PVI is determined by the characteristics of the tasks being performed and the problems being solved.

The technology for supporting such a PVI is far in the future. In addition to the hardware and software enabling technologies, design algorithms must be developed for resolving disagreement between the pilot and AI system, determining when the AI system should interrupt the pilot from his current task and enabling the PVI to change from one relationship to another.

### 4.4.2. Remotely Piloted Vehicles

With the exception of highly restricted applications, there appears to be no movement to eliminate humans from the flight deck. However, improved computer technology does enable autonomous and semiautonomous aircraft to be remotely piloted. This may be desirable when the presence of a human pilot is undesirable. Examples include vehicle missions that require very long endurance (e.g., space probes, communication relays, and fire controls) and military missions that are too dangerous for manned aircraft (e.g., defense suppression). When communication links may be sustained, the RPV may operate under the direction of a control station. In missions subject to the severance of communication links, the unmanned vehicle may have to operate autonomously. Autonomous operation is far in the future. The technology advances needed to support RPV development are intelligent sensor control, intelligent sensor processing, AI, communications, and remote control related devices.

#### 4.4.2.1. Control of a Remotely Piloted Vehicle

Semiautonomous RPV must be controlled from a "mother" craft. The next generation of spacecraft will include several RPVs. Examples of these are the Orbital Maneuvering Vehicle (OMV) and the Orbital Transfer Vehicle (OTV). The NASA Space Station will be the "mother" craft. These vehicles will serve as supply carriers, service satellites, and propel missions into deep space.

The latest technology is being applied to controlling the OMV and OTV from the space station. Pilot video monitors provide navigational displays to the pilot. The primary video monitor will display the main video camera signal received from the RPV. The secondary video monitor provides overlays to the primary monitor; the selection of a particular overlay is at the discretion of the pilot. Information displays will provide the pilot with work aids and secondary displays. These include set-up programs, checklists, operating procedures, real-time flight trajectory graphs, and alarms. The information displays will most likely be a color microcomputer display. A control panel will also be present in the form of a microcomputer display; touchscreen technology will provide graphic representation of push-button panels and associated status information.

Hand Controllers provide the facility for manually maneuvering the RPV. Usually two controllers will be present: one for control of the X-, Y-, and Z-axis maneuvering and one for roll, pitch, and yaw maneuvering. These two controllers

permit a six degree of freedom vehicle control. Touchscreen is the primary mode of control for function selection, but a keyboard and mouse may be used as a backup. An alternate mode of control is speech recognition and synthesis.

4.4.2.2. Avionics for a Remotely Piloted Vehicle

The preceding section discussed RPV control for a vehicle in a space application. Other RPVs may have to navigate the near-earth atmosphere at varying degrees of altitude. The avionics for such a vehicle would include flight control, inertial attitude reference, air data terminal, and payload/datalink systems. These are shown in schematic form in figure 4.4-2. At present, these RPVs are primarily used in military applications. As a result, the payload may consist of sophisticated weaponry with digital components or electronic countermeasures. The datalink is a secure communications system that supports the RPVs remaining at a safe distance from enemy defenses and maneuvering in a highly evasive fashion. The total cost of the RPV avionics would be concentrated in these two systems.

The FCS consists of a microprocessor with Random Access Memory (RAM) and/or Erasable Programmable Read-Only Memory (EPROM) memory. This processor handles the software functions associated with attitude stabilization, guidance, navigation, command execution, payload interface, datalink interface, and miscellaneous computational tasks. This microprocessor receives inputs from the flight sensor instrumentation and generates outputs to various servos and actuators. The inertial attitude reference is a sensor that provides accurate data on instantaneous vehicle rates. The air data terminal is the end of the data link on the RPV; it communicates with the control station.

Avionic systems represent approximately 80 percent of the cost of an RPV; the payload, itself, may represent 60 percent of the cost. Research and development effort is directed toward reducing the payload and ground control station costs. Since a major application for this type of RPV is target recognition, MV is an enabling technology for future RPV advances. (See paragraph 4.2.2.)

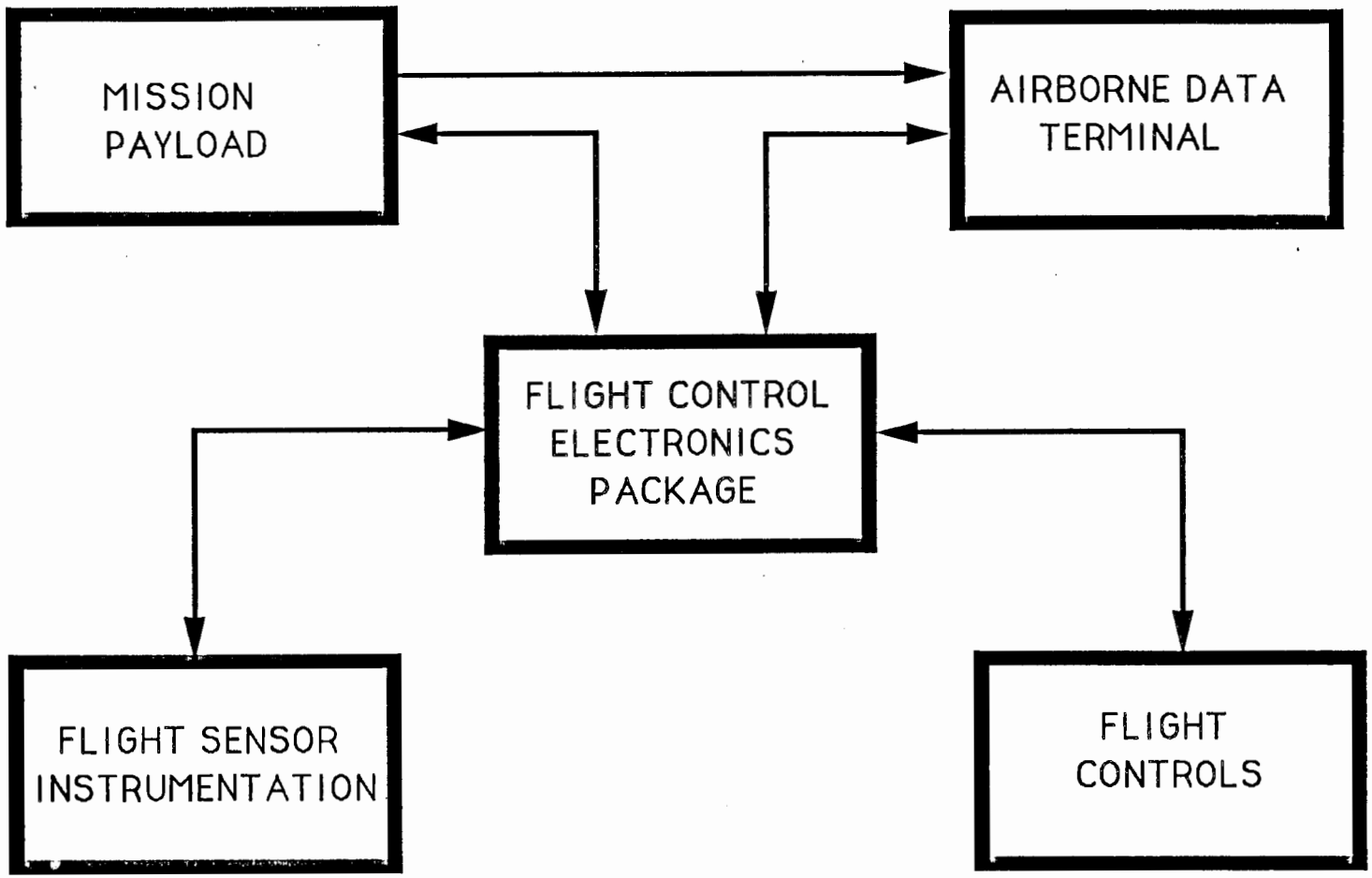



FIGURE 4.4-2. REMOTELY PILOTED VEHICLE AVIONICS

## BIBLIOGRAPHY


Aeronautical Policy Review Committee, Review of National Aeronautics Policy, Committee Report, November 1983.

Aeronautics and Space Engineering Board, Aeronautics Technology Possibilities for 2000: Report of a Workshop, 1984.

Brahney, J. H., "Certification Issues in Civil Helicopters," Aerospace Engineering, June 1988.

_____, "Moving Closer to Fully Integrated Control," Aerospace Engineering, October 1986.

_____, "Guidance, Navigation, and Control for 21st Century Aircraft," Aerospace Engineering, April 1986.

Buckanin, D. L., Artificial Intelligence: Overview and Annotated Bibliography," DOT/FAA/CT-TN84/5, March 1984.

Editor, "Fluidic Flight Control: Early Test Results," Aerospace Engineering, June 1988.

Editor, "Helping the Pilot Handle the Supercockpit," Aerospace America, February 1987.

Editor, "New Propulsion Concepts and FAA Certification," Aerospace Engineering, February 1988.

Epstein, A. H., ""Smart" Engine Components: A Micro in Every Blade?," Aerospace America, January 1986.

Executive Office of the President, Office of Science and Technology Policy Report, National Aeronautical R&D Goals - Technology for America's Future, March 1985.

Farina, J., R. Hubbard, and P. Lefkowitz, "Development and Test of a Digital/Optical Rotary Position Transducer, USAAVRADCOM TR 83-D-15 United States Army Research and Technology Laboratories, 1983.

Federal Aviation Administration Report, Aviation Technology - A View of the Future, Spring 1987.

Glomb, Jr., W. L., Passive Fiber Optic Coherence Multiplexing for Aircraft Sensors, IEEE/AIAA 7th Digital Avionics Systems Conference, 1986.

Gluch, D. P., and M. J. Paul, "Fault-Tolerance in Distributed Digital Fly-By-Wire Flight Control systems," IEEE/AIAA 7th Digital Avionics Systems Conference, 1986.

Hartley, C. S., and R. Pulliam, Use of Heads-Up Display, Speech Recognition, and Speech Synthesis in Controlling a Remotely Piloted space Vehicle, IEEE/AIAA 7th Digital Avionics systems Conference, 1986.

Lerner, E. J., "Many Processors Make Light Work," Aerospace America, February 1986.

Lupinetti, A., "Aviation Technology: A View of the Future," a viewgraph presentation, 1987.

_____, "Talking to Your Aircraft," Aerospace America, January 1986.

Martin, J., Design and Strategy for Distributed Processing, Prentice-Hall, Inc., 1981.

Maxwell, K. J., and J. A. Davis, Artificial Intelligence Implications for Advanced Pilot/Vehicle Interface Design, IEEE/AIAA 6th Digital Avionics Systems Conference, 1985.

Moore, C. A., R. D. Moore, and J. C. Ruth, Application of Voice Interactive Systems-Military Flight Test and Future, IEEE/AIAA 6th Digital Avionics Systems Conference, 1985.

National Aeronautics and Space Administration, "1986 Long-Range Program Plan," August 1985.

Otaguro, W. S., R. J. Michal, and S. F. Watanabe, Fiber Optic Monitors for Space Structures, IEEE/AIAA 7th digital Avionics Systems Conference, 1986.

Peterson, G. P., "Heat Removal Key to Shrinking Avionics," Aerospace America, October 1987.

Society of Automotive Engineers, SAE Committee AE4L Report No. AE4L-87-3, February 1987.

Terry, J. L., Tomorrow's Flight Controls: Helicopter Fly-By-Light, IEEE/AIAA 5th Digital Avionics Systems Conference, 1983.

Vandersteen, A. D., Avionics for the Small Remotely Piloted Vehicle, IEEE/AIAA 7th Digital Avionics Systems Conference, 1986.

## GLOSSARY

Ambient. The substance which absorbs heat from the heat sink.

Artificial Intelligence. The characteristics of a machine programmed to imitate human intelligence functions.

Autofeather. To automatically and swiftly feather the propeller when the engine fails to drive it.

Canard. A tail-first aerodyne, usually with auxiliary horizontal surface at the front and a vertical surface at the back.

Fly-By-Glass. Flight control system where fiber optics carry the signal.

Fly-By-Light. Flight control system where fiber optics carry the signal.

Fly-By-Wire. Flight control system with electric signaling.

Gigabit. One billion bits.

Kilobyte. One thousand bytes.

Micron. One-millionth of a meter.

Nanosecond. One-billionth of a second.

Thyristors. Solid-state devices that convert alternating current to direct current.

ACRONYMS

| | |
|---|---|
| 3-D | Three-Dimensional |
| ACS | Automatic Control System |
| AERA | Automated Enroute Air Traffic Control System |
| AFM | Advanced Fuel Management |
| AI | Artificial Intelligence |
| ATF | Advanced Tactical Fighter |
| ATI | Access Time Interval |
| CRT | Cathode Ray Tube |
| CSMA | Carrier Sensed Multiple Access |
| DARPA | Defense Advanced Research Projects Agency |
| DFCS | Digital Flight Control System |
| DOD | Department of Defense |
| EIU | Electronic Interface Unit |
| EMI | Electromagnetic Interference |
| EPROM | Erasable Programmable Read-Only Memory |
| FAA | Federal Aviation Administration |
| FADEC | Full Authority Digital Engine Controller |
| FAFTEEC | Full Authority Fault Tolerant Electronic Engine Control |
| FAR | Federal Acquisition Regulation |
| FBL | Fly-By-Light |
| FBW | Fly-By-Wire |
| FCS | Flight Control System |
| FET | Field Effect Transistor |
| GaAs | Gallium Arsenide |
| GNC | Guidance, Navigation, and Control |
| HERF | High-Energy Radio Frequency |
| ITT | (Consultative Committee for) International Telegraphy and Telephony |
| kA | kiloamp |
| LCD | Liquid Crystal Display |
| MAFT | Multicomputer Architecture for Fault Tolerance |
| Mbps | Million bytes per second |
| Mflops | Million floating-point operations per second |
| MPP | Massively Parallel Processor |
| OMV | Orbital Maneuvering Vehicle |
| OTV | Orbital Transfer Vehicle |
| PCS | Primary Control System |
| PLA | Power Level Angle |
| PVI | Pilot/Vehicle Interface |
| RAM | Random Access Memory |
| RCA | Radio Corporation of America |
| RF | Radio Frequency |
| RPV | Remotely Piloted Vehicle |
| RTCA | Radio Technical Commission for Aeronatics |
| SAE | Society of Automotive Engineers |
| STOL | Short Takeoff and Landing |

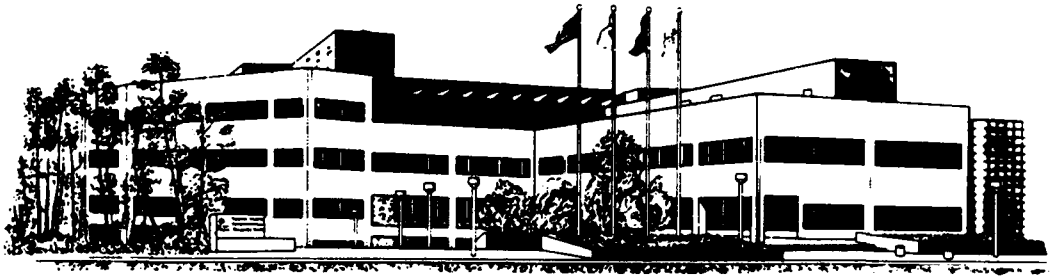
| | |
|---|---|
| TCAS | Traffic Alert and Collision Avoidance System |
| TTL | Transistor-Transistor Logic |
| USAF | United States Air Force |
| USB | Upper Surface Blowing |
| VHSIC | Very-High-Speed Integrated Circuits |
| VLSIC | Very-Large-Scale Integrated Circuits |
| VTOL | Vertical Takeoff and Landing |

U.S. Department of Transportation
Federal Aviation Administration

DOT/FAA/CT-88/10

# HANDBOOK-VOLUME II
# DIGITAL SYSTEMS VALIDATION

## CHAPTER 17
## SOFTWARE QUALITY METRICS



FEDERAL AVIATION ADMINISTRATION
TECHNICAL CENTER
ATLANTIC CITY INTERNATIONAL AIRPORT, NEW JERSEY 08405

NOTICE

This document is disseminated under the sponsorship of the U.S. Department of Transportation in the interest of information exchange. The United States Government assumes no liability for the contents or use thereof.

The United States Government does not endorse products or manufacturers. Trade or manufacturers' names appear herein solely because they are considered essential to the objective of this report.

**Technical Report Documentation Page**

| 1. Report No. | 2. Government Accession No. | 3. Recipient's Catalog No. |
|---|---|---|
| DOT/FAA/CT-88/10 | | |

| 4. Title and Subtitle | 5. Report Date |
|---|---|
| SOFTWARE QUALITY METRICS | July 1992 |
| | 6. Performing Organization Code |
| | 8. Performing Organization Report No. |
| 7. Author(s) | DOT/FAA/CT-88/10 |
| N. VanSuetendael and D. Elwell | |
| 9. Performing Organization Name and Address | 10. Work Unit No. (TRAIS) |
| Computer Resource Management, Inc.<br>200 Scarborough Drive, Suite 108<br>Pleasantville, New Jersey 08232 | 11. Contract or Grant No.<br>DTFA03-86-C-00042 |
| | 13. Type of Report and Period Covered |
| 12. Sponsoring Agency Name and Address | Tutorial<br>Handbook Chapter 17 |
| U.S. Department of Transportation<br>Federal Aviation Administration<br>Technical Center<br>Atlantic City International Airport, NJ 08405 | 14. Sponsoring Agency Code<br>ACD-230 |

15. Supplementary Notes

Pete Saraceni, FAA Technical Center, Program Manager

16. Abstract

When digital technology is employed to perform some function aboard aircraft, the designer documents the technology and the applicant presents a package to the Certification Engineer (CE). Typically, the package might include design and test specifications, test plans, and test results for the system. This package assures the CE that the designer has properly developed and validated the system. Software Quality Metrics (SQM) may be used during the software (SW) development and testing.

SQM technology attempts to quantify various quality-oriented factors, such as reliability and maintainability. The SW developer determines the quality factors that are important to the application. SW Metrics that correlate to these factors are used on the code to determine to what extent these factors have been reached. Based on the results, the developer determines whether the SW meets the requirements set for it and how well the SW will perform.

This technical report documents the results of a study conducted to analyze SQM as they apply to the code contained in avionic equipment and systems. This report is intended to provide an indepth explanation of how SQM may be applied and interpreted.

| 17. Key Words | 18. Distribution Statement |
|---|---|
| Software Quality Metrics, Software Complexity Metrics, Program Level, Analysis, Data Structure, Effectiveness Metric, Structural Complexity | Document is available to the U.S. public through the National Technical Information Service, Springfield, VA 22161 |

| 19. Security Classif. (of this report) | 20. Security Classif. (of this page) | 21. No. of Pages | 22. Price |
|---|---|---|---|
| Unclassified | Unclassified | 264 | |

Form DOT F 1700.7 (8-72) Reproduction of completed page authorized

## ACKNOWLEDGEMENT

The Federal Aviation Administration Technical Center and Computer Resource Management, Incorporated, wish to acknowledge Honeywell, Incorporated, Sperry Commercial Flight Systems Group of Glendale, Arizona, for their assistance in writing this document. Honeywell provided the FAA with flight control software code they developed. The code was used in evaluating and comparing different complexity metrics methodologies. In addition, Honeywell provided continuing technical support for their software during the research. All of this assistance was given at no cost to the Government.

TABLE OF CONTENTS


| Section | | Page |
|---|---|---|
| 1. | INTRODUCTION | 17-1 |
| 1.1 | Software Quality Metrics | 17-1 |
| 1.2 | Scope of this Chapter | 17-1 |
| 1.3 | How this Chapter is Organized | 17-2 |
| 2. | INTRODUCTION TO METRICS | 17-3 |
| 2.1 | Background of Software Quality Metrics | 17-3 |
| 2.2 | Definition of Software Quality Metrics | 17-4 |
| 2.3 | The Software Quality Metrics Model | 17-6 |
| 2.4 | Limits of Measurements | 17-6 |
| 2.5 | Metrics and Standards | 17-8 |
| 3. | PRACTICAL SOFTWARE QUALITY METRICS | 17-11 |
| 3.1 | Study Results | 17-11 |
| 3.2 | Software Quality Metrics Application and Analysis | 17-12 |
| 3.3 | Worked Example Problem Statement | 17-13 |
| 3.4 | Case #1: Software Quality Metrics Based on Halstead's Software Metrics | 17-14 |
| 3.4.1 | Metric Application and Analysis | 17-14 |
| 3.4.2 | Certification Conclusions | 17-33 |
| 3.5 | Case #2: Software Quality Metrics Based on McCabe's Software Metrics | 17-36 |
| 3.5.1 | Metric Application and Analysis | 17-36 |
| 3.5.2 | Certification Conclusions | 17-46 |
| 3.6 | Case #3: Software Quality Metrics Based on RADC's Software Metrics | 17-48 |
| 3.6.1 | Metric Application and Analysis | 17-49 |
| 3.6.2 | Certification Conclusions | 17-63 |

TABLE OF CONTENTS
(Continued)


| Section | Page |
|---|---|
| APPENDICES | |
| A - SOFTWARE METRIC DATA SHEETS | 17-65 |
| B - SOFTWARE QUALITY FACTOR DATA SHEETS | 17-131 |
| C - SOFTWARE METRIC DATA | 17-163 |
| BIBLIOGRAPHY | 17-179 |
| GLOSSARY | 17-193 |
| ACRONYMS AND ABBREVIATIONS | 17-195 |

LIST OF TABLES


| Table | | Page |
|---|---|---|
| 3.1-1 | VIABLE SOFTWARE QUALITY METRICS | 17-12 |
| 3.1-2 | PROMISING SOFTWARE QUALITY METRICS | 17-13 |
| 3.4-1 | OPERATOR AND OPERAND COUNTS, BY CODE CATEGORY | 17-23 |
| 3.5-1 | CYCLOMATIC COMPLEXITY, BY CODE CATEGORY | 17-44 |
| 3.5-2 | ESSENTIAL COMPLEXITY DATA | 17-45 |
| 3.6-1 | QUALITY FACTOR GOALS, BY CODE CATEGORY | 17-50 |
| 3.6-2 | MAINTAINABILITY CRITERIA | 17-52 |
| 3.6-3 | EXPANDABILITY CRITERIA | 17-52 |
| 3.6-4 | REUSABILITY CRITERIA | 17-52 |
| 3.6-5 | SELECTED METRIC ELEMENTS | 17-54 |
| 3.6-6 | METRIC ELEMENT SCORES, BY CODE CATEGORY | 17-58 |
| 3.6-7 | METRIC SCORES, BY CODE CATEGORY | 17-58 |
| 3.6-8 | CRITERIA SCORES, BY CODE CATEGORY | 17-60 |
| 3.6-9 | SOFTWARE QUALITY METRICS SCORES, BY CODE CATEGORY | 17-60 |
| 3.6-10 | FINAL SOFTWARE QUALITY METRICS SCORES, BY CODE CATEGORY | 17-63 |

# 1. INTRODUCTION

## 1.1 Software Quality Metrics

Technological advances have led to smaller, lighter, and more reliable computers that are used increasingly in flight applications. Federal Aviation Administration (FAA) Certification Engineers (CEs) are faced with certification of aircraft that depend on this digital technology, and on its associated software.

When digital technology is to be used to perform some function aboard aircraft, the designer documents the technology and the applicant presents a package to the CE. Typically, the package might include design and test specifications, test plans, and test results for the system. This package assures the CE that the designer has properly developed and validated the system. Software Quality Metrics (SQM) may be used during software development and testing.

SQM technology attempts to quantify various quality-oriented factors, such as reliability and maintainability. The software developer determines the quality factors that are important to the application. Software metrics that correlate to these factors are used on the code to determine to what extent these factors have been reached. Based on the results, the developer determines whether the software meets the requirements set for it, and how well the software will perform.

If SQM results are submitted to support a system to be certified, the CE should understand SQM and their results and implications. This chapter shows the step-by-step procedures for applying major SQM to avionic code. It shows the CE how SQM are applied, analyzed, and evaluated. This chapter can therefore be used as a reference to aid the CE when certifying avionic equipment or systems where SQM were used.

## 1.2 Scope of this Chapter

This chapter only includes metrics that are based on code, and that apply to the types of code used in avionic equipment. Metrics that are well substantiated by empirical evidence receive an in-depth discussion. In particular, this chapter applies and analyzes SQM based on three prominent families of software metrics: Halstead's, McCabe's, and Rome Air Development Center's (RADC).

Other SQM are included, but are addressed indirectly in appendices A and B. For a full discussion on these other metrics, see the Technical Report, Software Quality Metrics (N. VanSuetendael and D. Elwell 1991).

History metrics and metrics that cover all phases of the software development process are not within the scope of this chapter. Both categories, however, do warrant further investigation. In cases where an SQM falls in either of these

categories, the analysis is limited to that portion of the SQM that addresses the code phase.

Because SQM rely on statistics to indicate which software metrics correlate with a quality factor, this chapter does not cover all possible combinations leading to good correlations. The most accepted SQM are discussed. Acceptance is based on what SQM experts have concluded from their correlational studies. Future combinations of software metrics and quality factors are also addressed, since the SQM components are presented independently in appendices A and B.

## 1.3 How this Chapter is Organized

Section 2 discusses the background of SQM, defines them, and defines the model presented for analyzing them.

Sections 3.1, 3.2, and 3.3 summarize the results of the study and introduce the certification problem addressed by the SQM worked examples.

Section 3.4 discusses SQM based on Halstead's software metrics as they apply to developed avionic code. The SQM are calculated and analyzed step-by-step; then their application to the certification problem is evaluated. Sections 3.5 and 3.6 follow the same format as section 3.4 using SQM based on McCabe's Software Metrics and RADC's SQM methodology, respectively.

Readers interested only in how to apply the metrics of section 3 can read the steps and skip the analysis boxes. Those readers who have an interest in analyzing SQM should read the boxes that provide analyses for each step.

Appendices A and B contain data sheets that summarize the basic data a CE needs to understand, analyze, and evaluate these and other SQM. Appendix A addresses software metrics and appendix B addresses quality factors. Future SQM may correlate the software metrics of appendix A with new quality factors. Conversely, the quality factors of appendix B may be correlated with new software metrics. These two appendices are organized so that new SQM can still be analyzed.

Appendix C contains the software metric data on which the SQM calculations are based.

# 2. INTRODUCTION TO METRICS

## 2.1 Background of Software Quality Metrics

Computers play a primary role in industry and government. Because the hardware of modern systems relies heavily on the supporting software, software can critically affect lives. Many applications, e.g., flight critical systems, mission-critical systems, and nuclear power plants, require correct and reliable software. The increased importance of software also places more requirements on it. Thus, it is necessary to have precise, predictable, and repeatable control over the software development process and product. Metrics evolved out of this need to measure software quality.

In the early 70s, Halstead maintained that an attribute of software, such as complexity, is directly related to physical entities (operators and operands) of the code, and thus could be measured (Halstead 1972 and 1977). Halstead's metrics are fairly easy to apply. They are based on mathematical formulas and are backed by over a decade of refinement and follow-on empirical study. SQM based on Halstead's software metrics compose case 1 of the worked examples.

In the mid-70s, another researcher, McCabe, devised the Cyclomatic Complexity Metric, a measure based on the number of decisions in a program. McCabe's metric has been used extensively in private industry (Ward 1989). SQM based on McCabe's software metrics compose case 2 of the worked examples.

At approximately the same time, researchers at TRW became interested in measuring software qualities (Boehm et al 1978). The TRW team was responding to earlier research on the subject by Rubey and Hartwick. This approach considered software quality factors to be based on elementary code features that are assets to software quality. This work was adopted by RADC of the Air Force Systems Command. In 1977, RADC determined that software metrics were viable tools. Software acquisition managers could use them to determine accurately whether requirements were satisfied for new, delivered systems (McCall, Richards, and Walters 1977). RADC's SQM methodology is used in case 3 of the worked examples.

SQM based on these three software metric families are widely accepted. Recent research focuses on refining some of the initial assumptions and on creating new SQM.

In developing new metrics, theory often precedes empirical evidence. Today, numerous companies are marketing SQM. Some do not back their measurement claims with much independent research or empirical substantiation. Other companies, however, have empirically validated their metrics. Many companies use their metric systems to address all phases of the software life cycle.

Metric researchers are split broadly into two camps: those who claim software can be measured, and those who state that the nature of software does not lend itself to metrical analysis. The latter group includes researchers who have been disillusioned with the field, or who have mathematically "proved" that software cannot be measured. To further split the former group, some are busy refining existing metrics, others disproving parts of them. In any case, the majority of researchers are concerned about software quality and the need to quantify it.

Metric research is aimed at reducing software cost, keeping development on schedule, and producing software that functions as intended. Additionally, metrics are being developed to evaluate whether software accomplishes its functions reliably and safely. In the present context, metrics can be used to aid in the certification of avionic equipment and systems.

## 2.2 Definition of Software Quality Metrics

Before SQM can be understood, the terms need to be defined. A computer-based system consists of a machine and its associated programs and documentation. Hardware consists of the physical components of a computer. Software consists of computer programs and the documentation associated with the programs.

Code is a subset of software, and exists for the sole purpose of being loaded into the machine to control it. Code that can be read by people is source code. The translation of the source code that is loaded into the machine is called object code. This chapter addresses metrics that measure aspects of source code. In this chapter, any reference to code is to be understood as a reference to source code.

As used in the term, "SQM", quality modifies software. SQM measure whether software is quality software. This measure is based on quality factors. A quality factor is any software attribute that contributes either directly or indirectly, positively or negatively, toward the objectives for the system where the software resides. A particular quality factor is defined, or singled out, when it is determined that it significantly affects those objectives. Two examples are software reliability and complexity.

Metrics is another word for measures. The most basic measure is a repeatable, monotonic relationship of numbers to a magnitude of a property to be quantified. As soon as such a relationship is defined, the one set is a measure of the other. Repeatable means anyone will get the same results at any time. Since the relationship is a monotonic function, a certain change in the measure always represents a certain change in the property being measured, where either change is simply an increase or decrease in magnitude. (Increasingly negative numbers represent a decreasing magnitude.) Although such a function is consistently non-increasing or non-decreasing, it is not proportional. In addition, it is desirable that the monotonic measure have a one-to-one correspondence to the magnitude of the measured property so that it is not possible for differing property magnitudes to give the same measure.

A more desirable measure is one where the magnitude of the property is reflected in the measure by a smooth, mathematical relationship. This measure is

considered well-behaved. The most desirable and common well-behaved measure is a linear one, since any change in the magnitude of a property is proportionally represented by the measure.

The magnitude of properties of objects can usually be measured, since a property magnitude can almost always be expressed by a repeatable, monotonic relationship. Further, these measures are usually well-behaved, and are often linear. With respect to software, the objects are the printed symbols that represent ideas. Well-behaved measures exist for these objects (see Curtis 1980, for more details on measurement theory).

If a repeatable, monotonic relationship could be established that relates measures of objects (a set of numbers) to subjective qualities, the result would be, by definition, a quality measure. SQM is the name given to a measure that relates measures of the software objects (the symbols) to the software qualities (quality factors).

All practical SQM establish rules for relating the software symbols to the software quality factors. These rules are usually expressed as well-behaved mathematical relationships. However, few of the SQM produce a completely repeatable and monotonic measure of the software quality. Nearly all SQM in use exhibit variability and lack of monotonicity in their results.

Nevertheless, practical, reliable SQM do exist. Their validity is based on the statistical inference that can be applied to systems under development, drawn from the analysis of systems in operation. Experiments must be performed to analyze how well the values produced by the SQM correlate to the quality it claimed to measure. The strength of the relationship is expressed in terms of statistical correlation or regression. Thus, each SQM has a population or a sample space for which the measurement is valid, as well as a level of confidence that can be attributed to it.

When the relationship between two randomly varying quantities is measured in terms of correlation, the correlational coefficient, $\rho$, indicates how close the relationship, if any, comes to being linear. It does not give any indication of the coefficients of the linear relationship. A perfectly linear positive relationship is indicated by $\rho=1$. A perfectly linear negative relationship is indicated by $\rho=-1$. A value of $\rho=0$ indicates that the two quantities are not linearly correlated. Two quantities that are not correlated are not necessarily unrelated. They may follow some nonlinear relationship (Hays and Winkler 1970). Therefore, the correlational coefficient can establish the presence of a relationship, but not the absence.

A particular SQM may claim that a software quality is linearly related to a software metric. Correlational experiments may verify this with a value of $\rho=0.9$. However, the proposed linear SQM relationship is still not necessarily valid. In order to be useful as a relative measure, the slope of the linear relationship must be confirmed. In order to be useful as an absolute measure, the zero intercept of the linear relationship must also be determined. Thus, "If the correlation significantly differs from zero and if the estimates are generally close to their actual counterparts, then one could convincingly argue the metric is valid" (Dunsmore 1984).

When the relationship between two quantities is measured by regression analysis, the relationship is addressed more directly. A slope, an intercept, and the coefficient of determination, $r^2$, are calculated. This coefficient is a measure of the amount of variation in the dependent variable (quality factor) that may be explained by the variation in the independent variable (software metric). "A high coefficient of determination leaves less variability to be accounted for by other factors" (Basili and Hutchens 1983).

In practice, the word "metric" is used loosely. It must be discerned from the context whether the word is short for software metric or SQM. This is not always clear, because many practitioners assume that the software qualities are directly linked with the software objects. They use the word metric as though the measure of the software object were the same as measuring the software quality. In this chapter, the measure of the software objects will be referred to as software metrics, and the measure of the quality factors as SQM.

## 2.3 The Software Quality Metrics Model

There are two categories of elements measured by an SQM. The software object category contains directly measurable items: lines of code, conditional statements in a program or module, and number of unique operators and operands. The software quality factor category contains qualities software should possess to some degree: reliability, maintainability, simplicity, and ease of use. The two categories contain components for creating an SQM. Either category contains only the raw elements which, by themselves, signify little. When measures of the elements in the first category are combined (or "mapped") with elements from the other, an SQM is formed.

Thus, a particular SQM consists of three components: software metrics, software quality factors, and the mapping between the two. This relationship allows scientists, programmers, and engineers to measure the quality of the software being evaluated.

Practitioners emphasize the importance of agreeing on selecting quality factors appropriate to the application. Selecting appropriate quality factors is subjective. In the worked examples, each project's needs are identified first; the quality factors are then chosen based on these needs.

This model is flexible because new SQM are produced simply by combining new or existing software metrics with new or existing quality factors. The decision whether a software measure successfully "correlates" with a quality factor is subjective and depends on the needs of the application, the company sponsoring the research, and other concerns.

## 2.4 Limits of Measurements

SQM can be applied to any readable set of symbols. However, the results of a particular SQM will be meaningful only when the SQM is applied to the particular subset of symbols for which it has been prequalified. Thus, a metric can be thought of as a function which has a certain domain over which it is

defined. Outside that domain, the results are unpredictable, and therefore it should not be applied.

Within the domain, the measure of a software quality is only as good as the proposed correlation of the software quality to the software object measured, as determined by experimentation. The domain is typically no larger than the sample space of the experiments, although, if a sufficient number of experiments are performed, the sample space can reliably project the properties of the population. Within the domain, when a highly-correlated SQM is applied, it typically does not measure absolute levels of the software quality.

An SQM is generally a ratio that is expressed as a unitless number. For example, a high correlation is found between software maintenance time and the number of decision statements in a program. If Program A has twice as many decision statements as Program B, the time required to maintain A probably will be double that of B (assuming previous experimental studies show a linear relationship between the two entities). This type of result cannot specify a unit, such as particular number of hours of maintenance. It is a relative measure, only measuring trends. It does not assess software quality, but improvements in quality. The improvement suggested by a doubling of an SQM may actually represent an inconsequential improvement in absolute terms. The measurement could simply indicate the software has improved from unacceptable to barely passable.

If, however, the relationship has been calibrated by making comparisons to software whose qualities are established by after-the-fact experience, then the SQM does provide an absolute measure of software quality and will have units attached to it. Such a metric can be used to assess levels of software quality.

Software metrics, in and of themselves, predict nothing. They simply measure the state of the software at the time of measurement. For example, a particular code metric may indicate that, at the time of measurement, a module contained a specific number of decisions.

On the other hand, when a software metric is correlated to a quality factor, the SQM inherently predict some future state of the software. The quality factors are only meaningful when the code is applied in a context. Code, as printed on paper, exhibits no software quality. But in the context of a programmer having to read the code in order to change it, the presence of the quality factor, maintainability, becomes evident. Thus, when an SQM measures a software quality, it predicts that when the code is viewed in that context, the code will exhibit that quality.

SQM indirectly address testing adequacy. Testing adequacy assumes that the level of software quality desired is produced by testing for failures and fixing them until the level is achieved. Clearly, SQM are helpful here. They provide a quantitative assessment of the quality level of the software produced. This assessment can be compared to the quantitative level specified. Testing is objectively adequate when the specification is fulfilled. Without SQM, testing adequacy is a qualitative decision.

Specifying test coverage requirements is one way of assuring testing adequacy. RTCA Standard DO-178A recommends certain types of testing for software in each of the criticality categories. If all the test cases required by each type are performed, testing adequacy is assured. While SQM do not address test coverage, some software metrics address test coverage by implying the number of test cases necessary. Adequate coverage would require that the count produced by these metrics be met. This would not be sufficient to establish test coverage, however, since the software metric does not address particular test cases. Other software tools are specifically designed to generate test cases.

## 2.5 Metrics and Standards

SQM can be used to satisfy the requirements of some standards that apply to avionics. Title 14 of the United States Code of Federal Regulations (CFR) contains the Federal laws that apply to aeronautics and space. Chapter 1, Subchapter C, Federal Aviation Regulation (FAR) Part 25, specifies the airworthiness standards for transport category airplanes. In particular, Section 25.1309 specifies that airplane systems and associated components must be designed in such a way that the probability of their failure is not excessive. Compliance with this design requirement must be demonstrated by an analysis. SQM can be used in this analysis. This airworthiness standard currently contains no Special Conditions to which SQM technology would apply.

The requirement of Section 25.1309 is stated in qualitative terms. The FAA publishes formal guidelines for meeting the requirements of the CFR. One such guideline, Advisory Circular (AC) 25.1309-1A, gives specific design and analysis procedures and techniques which can be used to meet the requirements of Section 25.1309 of the CFR. Of particular interest for SQM is that this AC assigns quantitative thresholds to the probability of failure that is acceptable. This may make SQM even more useful for demonstrating compliance with Section 25.1309. The FAA has also adopted the Society of Automotive Engineers' (SAE) Aerospace Recommended Practice (ARP) 1834 as an informal guideline for the fault and failure analysis of digital systems and equipment, as introduced in AC 25.1309-1A. Some day SQM may be incorporated into these guidelines.

The certification process consists of determining whether avionic equipment and systems comply with Section 25.1309. To support this determination, the FAA has adopted the Radio Technical Commission for Aeronautics (RTCA) standard, RTCA/DO-178A, "Software Considerations in Airborne Systems and Equipment Certification" (1985), as another informal guideline. This standard recommends several software design and analysis procedures that lend themselves to SQM application.

The software development standard, DOD-STD-2167A, incorporates the use of SQM into the software development process. The requirements for reliability, maintainability, availability, and other quality factors are to be specified in the System/Segment Specification (SSS), as required by paragraphs 10.1.5.2.5ff of the associated Data Item Description, DID-CMAN-80008A. Furthermore, the software quality standard, DOD-STD-2168, recognizes that software development tools will be used in the software quality program. The tools are to be identified in the Software Quality Program Plan (SQPP) according to the specification of paragraph 10.5.6.2 of the associated Data Item Description, DID-QCIC-80572.

SQM have become so widely used that they are now being incorporated into national and international standards. Standardization of their application and interpretation will make them more useful to the software engineering community. The Institute of Electrical and Electronics Engineers (IEEE) has published standards on several commercial software metrics in IEEE Standard 982. This standard defines and explains the use of measures that can be used to produce reliable software. Furthermore, the IEEE Computer Society has produced a draft standard on an SQM methodology for the International Standards Organization (ISO). The ISO plans to release this standard in 1991.

# 3. PRACTICAL SOFTWARE QUALITY METRICS

## 3.1 Study Results

This study of current practical SQM revealed three families of software metrics upon which most viable SQM are based. In order to be a basis for a viable SQM, these software metrics had to meet the following criteria:

1. The metric must apply to code.

2. The metric must apply to code used in avionic equipment and systems. This code is usually written in Assembly, FORTRAN, Pascal, C, or Ada. Hence, any metrics that apply to these languages were generalized to apply to avionic code. Other metrics may also apply based on their universal application to code.

3. The metric must be related to a known software quality factor. In some cases, metrics are not correlated with quality factors, but can still help programmers develop software. For example, a performance metric shows the number of seconds a program takes to execute. The measurement determines to what degree the performance deviates from an accepted value, but does not measure the quality of the program.

4. The metric must be substantiated by independent experimental evidence.

Three additional families of software metrics include promising prospects for viable SQM, but the software metrics and their relationships to software quality factors require further substantiation. Each of the most established software metrics of these six families is summarized on a Software Metric Data Sheet in appendix A. Many other software metrics were encountered, but very little has been published about them. They are not addressed here.

In addition, this study revealed numerous quality factors that represent the concerns of software developers and users. Many of these are summarized on the Quality Factor Data Sheets in appendix B.

Most importantly, this study identified practical SQM which consist of experimentally verified relationships between some of the software metrics of appendix A and the quality factors of appendix B. These SQM are listed in table 3.1-1.

The study also identified SQM which suggest a relationship between some of the software metrics of appendix A and the quality factors of appendix B, but the relationships require further substantiation. These promising SQM are listed in table 3.1-2.

TABLE 3.1-1. VIABLE SOFTWARE QUALITY METRICS

| Software Metric Family | Software Metric | Quality Factor |
|---|---|---|
| Halstead | N | Maintainability<br>Number of Bugs<br>Reliability |
| | L | Complexity<br>Simplicity |
| | V | Complexity<br>Maintainability<br>Number of Bugs<br>Reliability<br>Simplicity |
| | $V^*$ | Conciseness<br>Efficiency |
| | I | Conciseness<br>Efficiency |
| | E | Clarity<br>Complexity<br>Maintainability<br>Modifiability<br>Modularity<br>Number of Bugs<br>Reliability<br>Simplicity<br>Understandability |
| | $\hat{B}$ | Maintainability<br>Number of Bugs<br>Reliability |
| McCabe | v(G) | Complexity<br>Maintainability<br>Modularity<br>Modifiability<br>Number of Bugs<br>Reliability<br>Simplicity<br>Testability<br>Understandability |
| | ev(G) | Complexity<br>Conciseness<br>Efficiency<br>Simplicity |
| RADC | AC.1 through VS.3<br><br>(see Bowen, Wigle, and Tsai 1985) | Completeness<br>Consistency<br>Correctness<br>Efficiency<br>Expandability<br>Flexibility<br>Integrity<br>Interoperability<br>Maintainability<br>Modularity<br>Portability<br>Reliability<br>Reusability<br>Simplicity<br>Survivability<br>Usability<br>Verifiability |

The appendices also contain software metrics and quality factors that are not part of any of the SQM identified in the tables. They represent components that are likely to be incorporated into subsequently proposed and developed SQM. By including them, this chapter better prepares the CE for evaluating whatever SQM an applicant may submit in a certification package.

## 3.2 Software Quality Metrics Application and Analysis

The CE should be able to understand, analyze, and evaluate any SQM that are presented. Three examples of the application and analysis of some of the SQM are given below. They present SQM based on each of the three most prominent families of software metrics. The examples are step-by-step worked examples of how a developer might have applied SQM to demonstrate the certifiability of some computerized avionics. They are presented as three different solutions to the same problem statement. An analysis of the application is interjected where appropriate. The analysis sections are set off from the metric application process by boxes. The data for the calculations can be found in appendix C.

TABLE 3.1-2. PROMISING SOFTWARE QUALITY METRICS

| Software Metric Family | Software Metric | Quality Factor |
|---|---|---|
| Albrecht | Function points | Complexity<br>Maintainability<br>Modularity<br>Performance<br>Simplicity<br>Understandability |
| Ejiogu | $S_c$ | Complexity<br>Maintainability<br>Modularity<br>Performance<br>Simplicity<br>Understandability |
| Henry | Information flow | Complexity<br>Maintainability<br>Modifiability<br>Modularity<br>Performance<br>Reliability<br>Simplicity<br>Understandability |

In the problem statement and the worked solutions of this section, the following qualifications apply:

- The context of the problem and solutions is fictitious.
- The code analyzed is real flight control code, consisting of 143 modules, written in Z80 assembly language.
- The flight control code was categorized by the developer into 59 essential modules and 84 nonessential.
- Two of the essential modules were copied, renamed, and arbitrarily categorized as critical modules, for the sake of providing a more complete example.

3.3 Worked Example Problem Statement

A flight control system submitted for certification utilizes software consisting of 145 modules, 84 of which are classified as nonessential, 59 as essential, and 2 as critical. The developer calculated SQM during the implementation of the code to measure the extent to which certain design and performance requirements

had been met (RTCA/DO-178A, paragraph 6.2.4.2). The developer also used SQM in the requirements coverage analysis and test coverage analysis (RTCA/DO-178A, paragraph 6.2.5.3). The applicant submitted the SQM results to the FAA as part of the implementation and testing assurance documentation.

From this documentation, the CE should determine whether

- the software was developed according to a disciplined approach (RTCA/DO-178A, paragraph 1.1),
- the software was developed and tested by a process appropriate to the software level (RTCA/DO-178A, paragraph 6.2.1),
- the SQM-based requirements analysis substantiates that the associated requirements are fulfilled (RTCA/DO-178A, paragraph 6.2.5.3.1),
- the SQM-based test coverage analysis substantiates test coverage (RTCA/DO-178A, paragraph 6.2.5.3.2), and
- the software will operate at the required level of performance.

Each of the worked examples demonstrates the process a developer followed to use SQM in the software development project described. Each developer used SQM that are based on a different family of software metrics. By observing the process, the CE will understand how to evaluate SQM reports submitted in certification documentation.

## 3.4 Case #1: Software Quality Metrics Based on Halstead's Software Metrics

In this case, the developer used Halstead's software metrics to assess the code quantitatively. To do this, the developer followed the steps described in the next section.

### 3.4.1 Metric Application and Analysis

Step 1. Specify the quality factors of concern and select the correlating software metrics for code in each category of criticality.

| Halstead Metric Application Analysis 1 |
|---|
| • The correlation of the selected quality factor to the selected software metric should be substantiated by previous experimentation. Otherwise the SQM is based solely on intuition. |

a. This developer was only contracted to supply the software for this system; another developer supplied the hardware. Therefore, the Software Requirements Specification (SRS) required that the code be written as efficiently as possible. It was estimated that if this were done, the software would not demand more memory resources than the hardware contractor

was to supply. The developer chose the Halstead Estimated Program Level, $\hat{L}$, as indicative of code size efficiency. It was specified that all modules must be programmed to exceed a Halstead Estimated Program Level of 0.05.

| Halstead Metric Application Analysis 2 |
| --- |
| • Although it was a good idea to quantify the efficiency specification, the developer did not refer to any experiments that correlated the Estimated Program Level to object code size. Furthermore, the level chosen as acceptable is based on previous experience that was not stated in the certification documentation. The CE should request further information to substantiate the validity of using this SQM. |

b. Since the code was expected to have about a 10-year operational phase, maintenance would be an ongoing expense. Furthermore, it was anticipated that code improvements would be requested on a fast turn-around basis. Therefore, it was decided that a software metric that correlated to maintainability would be used for all three levels of code. The software measures, Lines of Code and Halstead's Volume, V, were chosen. Any module that exceeded the initial calculation of the average value, plus one standard deviation for its category, was to be singled out, possibly to be reduced. The Halstead metric requires all of the basic counts of operators and operands. From experience, the cost of performing these counts was always recouped when maintenance is lessened for such a long operational phase.

| Halstead Metric Application Analysis 3 |
| --- |
| • The relationship of maintainability to Lines of Code and Halstead's Volume is well established by experimentation. However, the CE can accept this SQM as a reliable indicator of maintainability improvement only if the code being developed belongs to the same population of code on which the experiments have been performed.<br><br>The developer properly used this SQM to indicate which modules should be improved because they would require abnormal amounts of maintenance time. The developer did not propose any improper claims of absolute times or levels of maintainability. |

c. It was assumed that the effort to program each module would be a reasonable indicator of the relative effort to debug. Halstead's Programming Effort, E, was used to balance the workload during the debugging phase. Each programmer who was to debug modules was assigned an approximately equal number of Programming Effort units to debug.

| Halstead Metric Application Analysis 4 |
|---|
| • Halstead's Programming Effort metric has been shown to correlate strongly to actual debugging times. Nevertheless, because of variations in program complexity and programmer capability, the developer was right to limit its use to indicating the relative amount of effort required.<br><br>The CE should be sure that the experiments substantiating such a relationship apply to the environment of this project. |

d. Since no particular method is prescribed by RTCA/DO-178A for assessing the testing adequacy of essential modules, Halstead's Number of Bugs metric, $\hat{B}$, was used to determine testing adequacy. Testing was considered adequate when it uncovered the number of bugs estimated by $\hat{B}$.

| Halstead Metric Application Analysis 5 |
|---|
| • Halstead's Number of Bugs metric correlates well to the actual number of bugs in the experiments done on this topic. If those experiments apply to this code environment, use of the metric would indicate a certain amount of testing adequacy. |

e. Since the critical modules would be subjected to more rigorous testing, it was of particular interest that they rate high in testability. In particular, the number of coding errors that would be discovered during testing should be minimized, so that testing could focus on functional performance. Therefore, before testing began, all critical modules had to be coded so that Halstead's predicted Number of Bugs, $\hat{B}$, was less than one. Number of Bugs was used to measure testability.

| Halstead Metric Application Analysis 6 |
|---|
| • This is an innocuous use of an SQM. It may or may not have saved the developer testing time. Either way, the quality of the code was probably not impacted.<br><br>The CE can at least note that the development process was subjected to the disciplined approach of quantitative control (RTCA/DO-178A, 1.1). |

The use of the Number of Bugs metric required that evaluation be performed on the past work of the programmers working on essential and critical modules. For each programmer, the typical number of discriminations per bug, $E_0$, was obtained from this evaluation to be used in the Number of Bugs

calculation. Furthermore, the programmers who produced bugs at a lower rate were assigned to write the modules anticipated to require more mental discriminations. The cost of the evaluation would be justified by the testing adequacy assurance and testing effort reduction that would result from reducing the number of bugs.

f. In addition to bug reduction, it was considered important that the development process was followed by all programmers of essential and critical modules. If any of these modules varied more than one standard deviation from the mean value of Halstead's Estimated Program Level metric, $\hat{L}$, it was assumed that the programmer had not rigorously followed the programming standards for that module. Compliance with this requirement was taken as a measure of how consistently the programmers followed the standards.

| Halstead Metric Application Analysis 7 |
| --- |
| • This is a reasonable way of using SQM to provide a disciplined development process. Excessive variations in compliance to coding standards would result in some modules being overly-concise and others being too wordy. The Estimated Program Level metric would detect these relative extremes, although the absolute level may not have a particular meaning. |

g. All essential and critical modules were to be swapped from Programmable Read-Only Memory (PROM) into a certain protected area of the system Random Access Memory (RAM) during execution. Since the development computer was often tied up building system load modules, it was decided that Halstead's Intelligence Content metric, I, would be used to indicate whether each module would result in object code that would fit in the protected area, whatever size that area would be.

| Halstead Metric Application Analysis 8 |
| --- |
| • This measure is only intuitively related to object size. The relationship is based solely on the theoretical assertion that the Intelligence Content of a program does not vary when the program is translated. Experiments have not substantiated how well the Intelligence Content correlates to object size. Whether or not it does, the quality of the code was probably not impacted. A poor estimate primarily would impact the cost incurred by the developer. |

h. To help ensure the integrity of the critical modules, they should be kept very simple. It was decided to limit critical modules to a maximum value of Halstead's Program Length, N. The smaller, more numerous modules that would result would incur more system overhead during run-time, but this was considered to be a worthwhile compromise. Exceptions could be granted based

on the peculiar needs of the modules that did not naturally meet this limitation.

| Halstead Metric Application Analysis 9 |
| --- |
| • While the Program Length is strongly correlated to the perceived simplicity of a program, the threshold is set somewhat arbitrarily. Therefore, it is essential that those modules that exceed it receive a special review to determine if the constraint can reasonably be applied to them. The developer did well to allow exceptions to the Program Length maximum, in order to accommodate those modules for which the limitation was unreasonable. |

Step 2. Specify the level of quality considered acceptable for each SQM.

a. For the code size Efficiency quality factor, the contract required 100 percent compliance to the specified Estimated Program Level as the acceptable level of quality for all levels of code.

b. For the Maintainability quality factor, the developer required that at least 75 percent of the modules not exceed the limit, in either Lines of Code or Volume. The values of the two software metrics for each failing module gave an idea of what to fix to achieve compliance.

c,d. The Programming Effort metric and the Number of Bugs in the essential modules were not correlated to quality factors so no goals were set for them. They were simply used by managers to control and improve software development.

e. For the Testability quality factor, the developer required 100 percent compliance of the critical modules to the Number of Bugs threshold.

f. For the Consistency quality factor, the developer required 95 percent compliance. This applied only to the essential and critical modules.

g. The Intelligence Content was not correlated to a quality factor so no goal was set for it. It was simply used by managers to control the size of the product.

h. For the Integrity quality factor, the developer required 98 percent compliance by the critical modules.

With these preliminary selections, development began. After the code was written, the following metric steps were taken.

Step 3. Define the rules for determining that a word of source code is an operator, as defined on the data sheet of appendix A. The rules may be contained in a lookup table or may consist of rule statements. A word that does not satisfy the rules for an operator is taken to be an operand.

| Halstead Metric Application Analysis 10 |
|---|
| • The occurrence of an operator that was not anticipated by the rules will inflate the operand count and deflate the operator count. For instance, if the operator, CALL, occurs, but was not put in the lookup table, it will be counted as an operand.<br><br>• An operator manifested in a way that was not anticipated by the rules will inflate the operand count and deflate the operator count. For instance, if the operator, CALL C, occurs, but was only put in the lookup table in the form, CALL, it will be counted as an operand.<br><br>• An operand that accidentally satisfies the rules will inflate the operator count and deflate the operand count. This would only occur if the rules were less stringent than the language processor.<br><br>• The rules are necessarily different for every computer language or language variation. They are very closely tied to the rules used in the interpreter, compiler, or assembler. It is assumed that the code has been translated, without error, by one of them. Code from a different language processor will produce unreliable results.<br><br>• Different people will probably define the rules differently. The various sets of rules may produce similar results most, but not all of the time. |

For this case, the software metric tool, PC-Metric, was used to calculate Halstead's software metrics. PC-Metric uses a reserved word file as a lookup table of the words which are operators. In addition, other rules are followed. Since the theory of operators and operands is based on algorithm implementation, it does not apply to program comments and declarations. Thus, PC-Metric filters out comments, and declarative keywords are put in the keyword file and flagged that they are not to be counted. Typical entries in the reserved word file used by PC-Metric to calculate Halstead's metrics on Z80 Assembly code would be as follows:

```
2 (
0 )
2 *
2 +
2 ,
2 -
2 /
2 ADC
2 ADD
2 AND
1 ASEG
```

where the "2" flags a countable operator, the "0" flags that the pair of parenthesis operator is counted when the left parenthesis is encountered, and the "1" flags ASEG as a declaration word.

| Halstead Metric Application Analysis 11 |
| --- |
| • The calculations produced by a software metric tool reflect all the user-defined rules and all the decisions built into the program code. The only way to ensure a repeatable, predictable measure for comparison to some previous measure, is to maintain configuration control of the software metric tool and of the code being analyzed. Then, comparisons can be safely made only between two measurements that claim to be produced by, for example, PC-Metric, Version 1.3, on code successfully assembled by the Z80 Assembler, Version 2.6.<br><br>To further ensure the accuracy of the measurements, it is essential to use the software metric tool to analyze a test case for which the values can be manually calculated for comparison. This will ensure that the proper attribute is measured, and is measured accurately. |

Step 4. Define the segment of code that constitutes a module.

| Halstead Metric Application Analysis 12 |
| --- |
| • A module can be defined trivially as a single statement or as the next smallest independent function, or a subroutine, program, subsystem, or entire system of programs. In general, the larger the module is defined, the smaller will be the proportion of the number of unique words to the total number of words. |

A file was taken to be a module, since each file constituted a partition categorized, for certification purposes, as nonessential, essential, or critical. In general, this meant that a module was the same as an Assembly language subroutine. However, many files contained internal subroutines and/or additional external subroutines. In all cases, these were lumped together as a single module. The code consisted of 145 modules, thus defined.

Step 5. Define the rules for detecting the beginning and ending of each module.

| Halstead Metric Application Analysis 13 |
|---|
| • A module whose ending is not delineated, or is delineated in a way other than as specified in the rules, might be combined with the next module. This combination would inflate the counts for the first module and would cause the following module to go undetected.<br><br>• A module whose beginning is not delineated, or is delineated in a way other than as specified in the rules, might be combined with the previous module or might be ignored. In either case, it goes undetected.<br><br>• If any of the code accidentally satisfies the rules for indicating the beginning or ending of a module, the counts will likely be inaccurate.<br><br>• The rules are necessarily different for every computer language or language variation. They are very closely tied to the rules used in the interpreter, compiler, or assembler. It is assumed that the code has been translated, without error, by one of them. Code from a different language processor will produce unreliable results.<br><br>• Different people will probably define the rules differently. The various sets of rules may produce similar results most, but not all of the time. |

The first line of each file was assumed to be the beginning of the module. The end was generated by the EOF (End-of-File). There was, therefore, little ambiguity for meeting the chosen module definition.

Step 6. Segregate the modules into categories of nonessential, essential, and critical so that metric results can be used to aid the different development processes required for each category of code.

Following the guidelines of RTCA/DO-178A, the developer divided the 145 modules into 84 nonessential, 59 essential, and 2 critical.

Step 7. For each module, count the total number of operator occurrences, $N_1$, and the total number of operand occurrences, $N_2$. Also count the number of unique operators and the number of unique operands, $\eta_1$ and $\eta_2$, respectively. The data sheets of appendix A give the definitions of each of these quantities. Also, accumulate the measures for all modules in each category of criticality so that metric results can be used to aid the different development processes required for each category of code.

| Halstead Metric Application Analysis 14 |
|---|
| • The rules used to determine whether an operator or operand is unique must take into account the possibility that a word of code is to be interpreted as a different operator or operand, as determined by the context.<br><br>Whenever functionally unique operators or operands are represented by the same word of code, their respective count is deflated. For instance, BASIC usually uses a comma as an argument separator, but in parts of a PRINT statement, it is also a format specifier.<br><br>• The rules used to determine whether an operator or operand is unique must take into account the possibility of a single operator or operand having more than one form.<br><br>In this case, two slightly varying forms of the same operator or operand will be taken as the occurrence of two unique words, thereby inflating the unique count. This occurs in BASIC when a variable name may have up to 20 characters, but only the first eight are recognized. The full name and the first eight characters constitute two acceptable forms of the same operand.<br><br>• The values, $\eta_1$ and $\eta_2$, are not additive quantities. Adding the counts of unique operators or operands for sub-modules does not give accurate counts for the module that they constitute. |

PC-Metric accounted for most of the problems associated with the first point of analysis, mentioned above, by distinguishing between comments, declarations, and executable code. Within executable code, all unique operators and operands in a module are almost always constrained to be unique in form by the interpreter, compiler, or assembler.

In addition, PC-Metric had to distinguish between an asterisk in the first character position as designating a comment, but in subsequent positions as being the multiplication operator.

The second point of analysis in the previous analysis box was addressed by the "0" flag used on the right parenthesis. This flag indicates that a right parenthesis is not a unique operator, but is simply part of the left-right parenthesis pair.

The counts and the derived measures for each module of the code are listed in appendix C. The counts for each category of code are shown in table 3.4-1.

TABLE 3.4-1. OPERATOR AND OPERAND COUNTS, BY CODE CATEGORY

| | Nonessential | Essential | Critical |
|---|---|---|---|
| $\eta_1$ | 129 | 157 | 28 |
| $\eta_2$ | 1,418 | 1,059 | 51 |
| $N_1$ | 15,141 | 11,393 | 205 |
| $N_2$ | 11,602 | 8,676 | 160 |

Step 8. Calculate the various derived measures.

a. Calculate the Vocabulary, $\eta$, as follows:

$$\eta = \eta_1 + \eta_2 \text{ words}$$

| Halstead Metric Application Analysis 15 |
|---|
| • The vocabulary is not an additive quantity. |

The values for each category of code were as follows:

Nonessential:

$$\eta = 129 + 1,418$$
$$= 1,547 \text{ words}$$

Essential:

$$\eta = 157 + 1,059$$
$$= 1,216 \text{ words}$$

Critical:

$$\eta = 28 + 51$$
$$= 79 \text{ words}$$

b. Calculate the Implementation Length, N, as follows:

$$N = N_1 + N_2 \text{ words}$$

The values for each category of code were as follows:

Nonessential:

$$N = 15,141 + 11,602$$
$$= 26,743 \text{ words}$$

Essential:

$$N = 11,393 + 8,676$$
$$= 20,069 \text{ words}$$

Critical:

$$N = 205 + 160$$
$$= 365 \text{ words}$$

c. Alternatively, if the number of unique operators and operands is known, the Estimated Length, $\hat{N}$, can be calculated without counting all the occurrences of operators and operands. It is calculated as follows:

$$\hat{N} = \eta_1 \log_2 \eta_1 + \eta_2 \log_2 \eta_2 \text{ words}$$

| Halstead Metric Application Analysis 16 |
|---|
| • This estimate assumes that every combination of operators and operands of length $\eta$ occurs only once in any program. This means that repeated segments of code, as long as $\eta$ words, are put into subroutines rather than in redundant code. If such redundancy is present in the code, the Implementation Length will be underestimated by the Estimated Length Equation.<br><br>The Estimated Length Equation also assumes that operators and operands alternate without variation. Any code that does not follow this convention will likely underestimate the Implementation Length.<br><br>Whether or not these assumptions are reasonable, experiments show that $\hat{N}$ yields a close estimate of the Implementation Length, N.<br><br>• Because the Estimated Length Equation is nonlinear, adding the Estimated Lengths of sub-modules would not be expected to give an accurate Estimated Length for the module that they constitute. Nevertheless, experiments indicate that adding the lengths does give a fairly accurate estimate. |

The values for each category of code were as follows:

Nonessential:

$$\hat{N} = 129 \log_2(129) + 1,418 \log_2(1,418)$$
$$\approx (129 \times 7.011) + (1,418 \times 10.470)$$
$$\approx 904 + 14,846$$
$$\approx 15,750 \text{ words}$$

Essential:

$$\hat{N} = 157 \log_2(157) + 1{,}059 \log_2(1{,}059)$$
$$\approx (157 \times 7.295) + (1{,}059 \times 10.0485)$$
$$\approx 1{,}145.3 + 10{,}641.4$$
$$\approx 11{,}787 \text{ words}$$

Critical:

$$\hat{N} = 28 \log_2(28) + 51 \log_2(51)$$
$$\approx (28 \times 4.807) + (51 \times 5.672)$$
$$\approx 135 + 289$$
$$\approx 424 \text{ words}$$

d. Calculate the Volume, V, as follows:

$$V = N \log_2 \eta \text{ bits}$$

| Halstead Metric Application Analysis 17 |
|---|
| • This metric calculates the number of binary digits required to uniquely represent all of the operators and operands that occur in a module of Length, N, and Vocabulary, $\eta$. It is not primarily a measure of the size or complexity of the function programmed, but a measure of the size of a particular implementation of the function.<br><br>• Volume is not an additive quantity. |

The values for each category of code were as follows:

Nonessential:

$$V = 26{,}743 \log_2(1{,}547)$$
$$\approx 26{,}743 \times 10.5952575$$
$$\approx 283{,}348.971 \text{ bits}$$

Essential:

$$V = 20{,}069 \log_2(1{,}216)$$
$$\approx 20{,}069 \times 10.2479275$$
$$\approx 205{,}665.657 \text{ bits}$$

Critical:

$$V = 365 \log_2(79)$$
$$\approx 365 \times 6.30378$$
$$\approx 2{,}300.88 \text{ bits}$$

e. **Calculate the Potential Volume, $V^*$, as follows:**

$$V^* = (2 + \eta_2^*)\log_2(2 + \eta_2^*) \quad \text{bits}$$

where $\eta_2^*$ is the sum of the number of unique inputs and outputs for the module.

| Halstead Metric Application Analysis 18 |
|---|
| • This metric calculates the number of binary digits required to represent the function of a module in its most efficient form, namely two operators (i.e., one that says "do the function", and another that groups the operands with the operator) and all the operands that the function requires.<br><br>• Potential Volume is not an additive quantity. |

The number of inputs and outputs for the code is not known, so no calculation of Potential Volume can be made.

f. Calculate the unitless ratio, Program Level, L, as follows:

$$L = \frac{V^*}{V}$$

| Halstead Metric Application Analysis 19 |
|---|
| • This metric calculates how efficiently the module is implemented. As the Vocabulary and/or Implementation Length decrease (s), the Program Level increases, i.e., the module is written at a higher level because each word carries more weight.<br><br>• The Program Level is not an additive quantity. |

The Potential Volume could not be calculated, so no direct calculation of Program Level was made.

g. Alternatively, if the number of unique inputs and outputs is not known, the Estimated Program Level, $\hat{L}$, can be calculated from the basic counts, as follows:

$$\hat{L} = \frac{2\ \eta_2}{\eta_1\ N_2}$$

The values for each category of code were as follows:

Nonessential:

$$\hat{L} = (2 \times 1{,}418)/(129 \times 11{,}602)$$
$$= 2{,}836/1{,}496{,}658$$
$$\approx 0.00189488848$$

Essential:

$$\hat{L} = (2 \times 1{,}059)/(157 \times 8{,}676)$$
$$= 2{,}118/1{,}362{,}132$$
$$\approx 0.00155491538$$

Critical:

$$\hat{L} = (2 \times 51)/(28 \times 160)$$
$$= 102/4{,}480$$
$$\approx 0.0227679$$

h. Calculate the Intelligence Content, I, as follows:

$$I = \frac{2\ \eta_2}{\eta_1\ N_2} \times (N_1 + N_2)\log_2(\eta_1 + \eta_2) \quad \text{bits}$$

| Halstead Metric Application Analysis 20 |
|---|
| • This metric simply calculates the product of the Estimated Program Level (how efficiently the module is implemented) and Volume (the size of the implementation) of a module. It represents that constant quantity of information that is present in any implementation of a particular function, in any language, at any level. This quantity is basically an estimated Potential Volume of the program; I and $V^*$ can often be used interchangeably.<br><br>• Intelligence Content is not an additive quantity. |

The values for each category of code were as follows:

Nonessential:

$$I = 0.00189 \times 283{,}349$$
$$\approx 536.9 \text{ bits}$$

Essential:

$$I = 0.00155 \times 205{,}666$$
$$\approx 319.8 \text{ bits}$$

Critical:

$I = 0.02277 \times 2{,}301$
$\approx 52.4$ bits

i. Calculate the Programming Effort, E, as follows:

$E = V/L$ discriminations

| Halstead Metric Application Analysis 21 |
|---|
| • This metric calculates the number of elementary mental discriminations done by a programmer to reduce a preconceived algorithm to a module of source code in a language in which the programmer is fluent.<br><br>It is based on the assumption that, when writing a program, a programmer selects each word of the program by mentally searching a list of words from which to choose. Specifically, the programmer performs a mental binary search of the vocabulary of $\eta$ words in order to select the N words used in the implementation. Furthermore, each comparison (or mental discrimination) in the selection process requires an effort related to the difficulty of understanding the program. The program difficulty is supplied by the reciprocal of the Program Level.<br><br>This relationship indicates that the greater the Volume, or the lower the Program Level, the greater the effort required to write a program.<br><br>• Programming Effort is not an additive property. |

Using $\hat{L}$ for L, the values for each category of code were as follows:

Nonessential:

$E = 283{,}348.971/0.00189488848$
$\approx 149{,}533{,}323$ discriminations

Essential:

$E = 205{,}665.657/0.00155491538$
$\approx 132{,}268{,}070$ discriminations

Critical:

$E = 2{,}300.88/0.0227679$
$\approx 101{,}058$ discriminations

j. Calculate the Estimated Programming Time, $\hat{T}$, as follows:

$$\hat{T} = \frac{E}{3600S} = \frac{\eta_1 N_2(\eta_1 \log_2 \eta_1 + \eta_2 \log_2 \eta_2)\log_2 \eta}{7{,}200\eta_2 S} \text{ hours}$$

| Halstead Metric Application Analysis 22 |
|---|
| • This metric calculates the time it takes to make the number of elementary mental discriminations indicated for a program by the Programming Effort. The Stroud Number, S, is the total number of elementary mental discriminations done per second by a person.<br><br>This calculation assumes that the programmer's attention is entirely undivided and that the range of values found from psychological experimentation, $5<S<20$, applies to programming activity.<br><br>• Estimated Programming Time is not an additive quantity. |

Assuming a value for S of 18, the values for each category of code were as follows:

Nonessential:

$\hat{T}$ = 149,533,323/(3600 x 18)
≈ 2,308 hours

Essential:

$\hat{T}$ = 132,268,070/(3600 x 18)
≈ 2,041 hours

Critical:

$\hat{T}$ = 101,058/(3600 x 18)
≈ 1.6 hours

k. Calculate the Language Level, $\lambda$, as follows:

$$\lambda = L^2 V$$

| Halstead Metric Application Analysis 23 |
|---|
| • This metric calculates the efficiency with which algorithms can be implemented in a particular language. For any module written in the particular language, the Language Level should be a constant, regardless of the Volume of the module or the Program Level at which it is written. |

Using $\hat{L}$ for L, the values for each category of code were as follows:

Nonessential:

$$\lambda = (0.00189)^2 \times 283,349$$
$$= 1.02$$

Essential:

$$\lambda = (0.00155)^2 \times 205,666$$
$$= 0.50$$

Critical:

$$\lambda = (0.02277)^2 \times 2,301$$
$$= 1.19$$

1. Calculate the Number of Bugs, $\hat{B}$, as follows:

$$\hat{B} = V/E_0 \text{ bugs}$$

where $E_0$ is the number of discriminations per bug. $E_0$ is determined by an evaluation of a programmer's previous work.

| Halstead Metric Application Analysis 24 |
|---|
| • Experimentation has found that 3,200 discriminations per bug is a typical value. |

The values for each category of code were as follows:

Nonessential:

$$\hat{B} = 283,349/3,200$$
$$\approx 89 \text{ bugs}$$

Essential:

$$\hat{B} = 205,666/3,200$$
$$\approx 64 \text{ bugs}$$

Critical:

$$\hat{B} = 2,301/3,200$$
$$\approx 0.7 \text{ bugs}$$

Step 9. Calculate the SQMs defined in step 1 of this case.

a. Calculate the Efficiency quality factor, as follows:

$$\text{Efficiency} = \frac{\text{\# of modules with } \hat{L} > 0.05}{\text{total number of modules}} \text{ \% compliance}$$

| Halstead Metric Application Analysis 25 |
|---|
| • This metric is not a measure of efficiency, but the extent to which the desired efficiency was present. This makes it a measure of quality. |

The value for the code can be calculated from the data in appendix C.

Efficiency = 42/145
= 29.0% compliance

Since 100 percent compliance was specified, the development process clearly needed refining, if the code size efficiency goal was to be met.

b. Calculate the Maintainability quality factor, as follows:

$$\text{Maintainability} = \frac{\text{\# of modules that met the goal}}{\text{total number of modules}} \text{ \% compliance}$$

The value for the code can be calculated from the data in appendix C. (The noncompliant data are flagged with an asterisk.)

Maintainability = 119/145
= 82.1% compliance

Thus, the maintainability goal was met on the first check. Furthermore, the 26 noncompliant modules were reviewed and several of them were reduced in size.

c. Based on the total Programming Effort units for the code, the 28 programmers were assigned as follows:

Nonessential: 15 programmers

Effort per programmer = 149,533,323/15
≈ 10.0 million units

Essential and critical: 13 programmers

Effort per programmer = 132,369,128/13
≈ 10.2 million units

This assignment gave less than a three percent variation in programmer load. Similar calculations were made to determine how many modules to assign to each programmer.

d. Generally, for each module of essential code, testing was performed until the predicted number of bugs was found. If less than one bug was predicted, no testing was done. For some modules predicted to have bugs, none were found. These were tested until path coverage was obtained.

e. Calculate the Testability, as follows:

$$\text{Testability} = \frac{\text{\# of modules with } \hat{B} < 1}{\text{total \# of modules}} \text{ \% compliance}$$

The value for the critical modules of the code can be calculated from the data in appendix C.

Testability = 2/2
= 100% compliance

f. Calculate the Consistency, as follows:

$$\text{Consistency} = \frac{\text{\# of modules with } \hat{L} \text{ within } \sigma}{\text{total \# of modules}} \text{ \% confidence}$$

The initial value for the essential and critical modules of the code can be calculated from the data in appendix C. (The noncompliant data are flagged with an asterisk.)

Consistency = 54/61
= 88.5% compliance

Four modules were refined to meet the 95 percent compliance required.

g. The Intelligence Content metric indicated that the module, ESSENT34, would result in the largest object module. Therefore, the protected area of memory was made 10 percent larger than the actual size of this module's

object code. This size successfully contained all the other essential and critical object modules.

b. Calculate the Integrity as follows:

$$\text{Integrity} = \frac{\text{\# of modules with } N < 250}{\text{total \# of modules}} \text{ \% compliance}$$

The value for the critical modules of the code can be calculated from the data in appendix C.

$$\text{Integrity} = 2/2$$
$$= 100\% \text{ compliance}$$

3.4.2 Certification Conclusions

For certification purposes, it should be determined whether the metrics, as applied, were indeed reliable indicators of the properties measured. It should also be determined whether such measures equivalently fulfill the requirements of the certification guidelines. This determination can be based on the experience of the developer, the experimentation performed by the developer, or the experimentation that the developer references, as documented in the certification package. It can also be based on the experience of the CE or on reference by the CE to the handbook data sheets.

Based on these inputs, the SQM performed are evaluated to see if they help the CE answer any of the certification determinations posed in section 3.3. This evaluation addresses each SQM in the order in which they were specified in step 1.

a. The code size efficiency SQM was used to substantiate that the software requirements had been met. However, the relationship of $\hat{L}$ to object code size has not been established by experimentation. Furthermore, it has not been established that a program written with an Estimated Program Level of 0.05 is considered a very efficient program, or that the much lower values measured indicate that the modules are very inefficient.

| Halstead Metric Application Analysis 26 |
|---|
| • The CE would likely conclude that this SQM does not fulfill the requirements analysis recommended by RTCA/DO-178A, paragraph 6.2.5.3.1, for essential and critical software. This SQM does not establish whether the efficiency requirement was or was not met. |

b. The use of the Maintainability SQM indicates that the developer subjected the code to a responsible, quantitatively controlled development process.

| Halstead Metric Application Analysis 27 |
| --- |
| • First, the CE should compare this software development project to those of the experiments on Maintainability. The CE would then likely conclude that this SQM offers some assurance that the software was developed by a process appropriate to the software level, as recommended by RTCA/DO-178A, paragraph 6.2.1. |

c. Since the debugging load was balanced using the Programming Effort software metric, the developer maintains that it is more likely that the modules are consistently debugged and that they operate properly. Otherwise, some programmers would have been overloaded and their search for bugs would have been truncated in order to keep or schedule.

| Halstead Metric Application Analysis 28 |
| --- |
| • First, the CE should compare this software development project to those of the experiments on debugging times. The CE would then likely conclude that this SQM offers some assurance that the software will operate at the required level of performance. |

d. Since the testing of essential modules was regulated with the Number of Bugs metric, the developer maintains that the modules are likely to have fewer bugs than if this criterion had not been established.

| Halstead Metric Application Analysis 29 |
| --- |
| • First, the CE should compare this software development project to those of the experiments on the number of bugs found during testing. The CE could then conclude that this SQM offers some assurance that the software will operate at the required level of performance. However, this metric is not as assuring as others since it claims to be an absolute, rather than a relative measure. |

e. Since the critical modules met the Number of Bugs threshold, the developer maintains that they are more easily tested than the other modules. Testing will not be inhibited by undetected coding errors.

| Halstead Metric Application Analysis 30 |
| --- |
| • First, the CE should compare this software development project to those of the experiments on the number of bugs found |

during testing. The CE could then conclude that this SQM offers some assurance that the critical code has been subjected to a more demanding development process. However, this metric is not as assuring as others since it claims to be an absolute, rather than a relative measure. Particularly, the critical code satisfied the absolute level, as coded. It is possible that the nonessential and essential code would also have satisfied the absolute level, as coded.

f. The developer used the Estimated Program Level to regulate the consistency of the essential and critical modules with the programming standards. Thus, their consistency was checked but the consistency of the nonessential modules was not. Furthermore, this check resulted in improvements.

| Halstead Metric Application Analysis 31 |
| --- |
| • First, the CE should compare this software development project to those of the experiments on Estimated Program Level. The CE would then likely conclude that this SQM offers some assurance that the software was developed by a process appropriate to the software level, as recommended by RTCA/DO-178A, paragraph 6.2.1. Regardless of the applicability of the experiments, the SQM resulted in inconsistencies being detected and corrected. |

g. The Intelligence Content metric was not used in a way that relates to any certification concerns.

h. Since the critical modules met the Program Length threshold, the developer maintains that they have more integrity than the other modules.

| Halstead Metric Application Analysis 32 |
| --- |
| • First, the CE should compare this software development project to those of the experiments on Program Length. The CE could then conclude that this SQM offers some assurance that the critical code has been subjected to a more demanding development process. However, this metric is not as assuring as others since it was used as an absolute measure, rather than a relative measure. Particularly, the critical code satisfied the absolute level, as coded. It is possible that the nonessential and essential code would also have satisfied the absolute level, as coded. |

This developer could not use SQM to substantiate test coverage, as defined in RTCA/DO-178A, paragraph 6.2.5.3.2, because Halstead's software metrics do not measure anything that substantiates structural test coverage.

In this example, the developer used both software metrics and SQM to analyze the flight control code being developed. The metrics were applied in various ways and for various purposes. In all cases, the metrics provided a quantitative analysis of the software product. This analysis was used to assess, and sometimes improve, the software development product and process.

### 3.5 *Case #2: Software Quality Metrics Based on McCabe's Software Metrics*

In this case, the developer used McCabe's software metrics to assess the code quantitatively. To do this, the developer followed the steps described in the next section.

#### 3.5.1 Metric Application and Analysis

Step 1. Specify the quality factors of concern and select the correlating software metrics for code in each category of criticality.

| McCabe Metric Application Analysis 1 |
|---|
| • The correlation of the selected quality factor to the selected software metric should be substantiated by previous experimentation, otherwise the SQM is based solely on intuition. |

a. This developer was concerned that the code developed would be easy to maintain. Therefore, a software metric that correlated to maintainability was used for all three levels of code. The software measure, Cyclomatic Complexity, was chosen. Any module that exceeded the initial calculation of the average value, plus one standard deviation for its category, was to be singled out, possibly to be reduced. Since this metric is easily calculated by a software tool, it was well worth the effort to calculate it for all modules.

| McCabe Metric Application Analysis 2 |
|---|
| • The relationship of maintainability to McCabe's Cyclomatic Complexity measure is well established by experimentation. However, the CE can accept this SQM as a reliable indicator of maintainability improvement only if the code being developed belongs to the same population of code on which the experiments have been performed.<br><br>The developer properly used this SQM to indicate which modules should be improved because they would require abnormal amounts of maintenance time. The developer did not propose any improper claims of absolute times or levels of maintainability. Since the threshold is set somewhat arbitrarily, it is essential that those modules that exceed it receive a special review to determine if the constraint can reasonably be applied. |

b. Since the Cyclomatic Complexity of each module is a reasonable indicator of the relative number of errors to debug, this measure was used to balance the workload during the debugging phase. Each programmer who was to debug modules was assigned an approximately equal number of Cyclomatic Complexity units to debug.

| McCabe Metric Application Analysis 3 |
| --- |
| • McCabe's Cyclomatic Complexity metric has been shown to correlate strongly to the number of errors found in program modules (Walsh 1979; and Henry, Kafura, and Harris 1981). Nevertheless, because of variations in program size and programmer capability, the developer was right to limit its use to indicating the relative amount of debugging effort required.<br><br>The CE should be sure that the experiments substantiating such a relationship apply to the environment of this project. |

c. Since critical modules require structural coverage analysis for assurance of testing adequacy, the developer chose to perform a branch coverage analysis to determine the test cases to use. The Cyclomatic Complexity of each module was used to specify a number of test cases that would ensure branch coverage. The developer recognized that it is sometimes possible to satisfy branch coverage with even fewer test cases, but the ease of calculating v(G) justified its use.

| McCabe Metric Application Analysis 4 |
| --- |
| • The Cyclomatic Complexity Metric does give a number of test cases sufficient to ensure branch coverage but does not guarantee that the developer chose the proper test cases.<br><br>RTCA/DO-178A, paragraph 6.2.5.3.2, recommends using structured coverage analysis to ascertain which test cases constitute testing adequacy of the elements of critical modules. This SQM cannot be accepted as an equivalent (RTCA/DO-178A, paragraph 1.1) to the full analysis required, since it only establishes the number of cases, not the validity of each case.<br><br>The CE can at least note that the development process is being subjected to the disciplined approach of quantitative control (RTCA/DO-178A, paragraph 1.1). |

d. Since the critical modules would be subjected to more rigorous testing, it was of particular interest that they rate high in testability. In particular, it was desired to minimize the number of coding errors that would be discovered during testing so that testing could focus on functional

performance. Therefore, before testing began, all critical modules had to be coded so that the Cyclomatic Complexity was no more than 10. Cyclomatic Complexity was used to measure testability.

| McCabe Metric Application Analysis 5 |
| --- |
| • This SQM is a particularly useful one. A study found that not only was the number of errors positively correlated to v(G), but that modules with v(G) greater than 10 showed a disproportionate number of errors compared to those with v(G) less than 10 (Walsh 1979). For the AEGIS Naval Weapon System software, the absolute measure, v(G)=10, was a threshold for error prone modules. |

e. The developer reasoned that the module size limitation and the testing proposed were insufficient to uncover the complicated exceptions to simple program control flow that results from unstructured programs. The developer chose to measure the proportion of structured programming in the critical modules with the Essential Complexity metric, and to use this as an indicator of the module integrity. Not only would these modules be limited to 10 branch conditions, but of the 10, the developer required that a minimum of half of them be structured. This requires an Essential Complexity of five, at most.

| McCabe Metric Application Analysis 6 |
| --- |
| • The Essential Complexity certainly measures the amount of structure in the module control flow, but how strongly it correlates to program integrity has not been established. The developer based this choice on intuition. The CE should require that the developer substantiate this relationship. |

Step 2. Specify the level of quality considered acceptable for each SQM.

a. For the Maintainability quality factor, the developer required that at least 80 percent of the modules not exceed the limit.

b,c. No quality goals were set for the debugging effort or the number of test case metrics. These metrics were used as inputs into the management of the software development process, rather than as a specification for the product.

d. For the Testability quality factor, the developer required 100 percent compliance of the critical modules to the Cyclomatic Complexity threshold.

e. For the Integrity quality factor, the developer required 100 percent compliance of the critical modules to the Essential Complexity threshold.

With these preliminary selections, development began. After the code was written, the following metric steps were taken.

Step 3. Define the rules for identifying and counting program control flow conditions. The rules may be contained in a lookup table or may consist of rule statements.

| McCabe Metric Application Analysis 7 |
| --- |
| • The predicate portion of a conditional branch of program control flow may contain more than one condition. The conditions must be identified and counted, not just the predicates.<br><br>• Unconditional branching does not contribute to the count of program control flow conditions.<br><br>• The occurrence of a conditional branch keyword that was not anticipated by the rules will deflate the condition count. For instance, if a CALL-on-carry-true occurs, but CALLC was not put in the lookup table, this branch condition will not be counted.<br><br>• A conditional branch manifested in a way that was not anticipated by the rules, will deflate the condition count. For instance, if a CALL-on-not-zero-true may be abbreviated to CALLN, but the lookup table only contains the expanded form, CALLNZ, an occurrence of the branch condition in the form CALLN will not be counted.<br><br>• In Assembly language, a jump to an indirectly addressed pointer constitutes a case structure which has many branch conditions. The number of unique modifications made to the referenced memory location determines the number of branch conditions produced in this manner.<br><br>• A keyword that accidentally satisfies the rules will inflate the condition count. This would only occur if the rules were less stringent than the language processor.<br><br>• The rules are necessarily different for every computer language or language variation. They are very closely tied to the rules used in the interpreter, compiler, or assembler. It is assumed that the code has been translated, without error, by one of them. Code from a different language processor will produce unreliable results.<br><br>• Different people will probably define the rules differently. The various sets of rules may produce similar results most, but not all of the time. |

For this case, the software metric tool, PC-Metric, was used to calculate McCabe's software metrics. PC-Metric uses a reserved word file as a lookup table of which keywords delineate conditional branch predicates. In addition, other rules are followed. Since this metric is based on program control flow, it does not apply to program comments and declarations. Thus, PC-Metric filters out comments, and declarative keywords are put in the keyword file and flagged that they are not to be counted. Typical entries in the reserved word file used by PC-Metric to calculate McCabe's metrics on Z80 Assembly code would be as follows:

```
1 ASEG
1 DEFB
A CALL
3 CALLC
3 CALLZ
A JP
4 JP(IX)
3 JPC
3 JPZ
A RET
3 RETC
3 RETZ
```

where the "3" flags a keyword/condition combination that delineates a statement with a single branch condition, the "4" flags a keyword/condition combination that delineates a statement with multiple branch conditions, the "A" flags an unconditional variant of a keyword that often delineates a statement containing a conditional branch predicate, and the "1" flags declaration words.

A case structure is the only compound predicate that Z80 code contains. PC-Metric maintains an Extended Complexity Count for these statements. The Extended Complexity count is equal to the Cyclomatic Complexity count, which counts all the branching statements, plus the number of those branching statements that contain multiple branch conditions.

| McCabe Metric Application Analysis 8 |
| --- |
| • The calculations produced by a software metric tool reflect all the user-defined rules and all the decisions built into the program code. The only way to ensure a repeatable, predictable measure for comparison to some previous measure, is to maintain configuration control of the software metric tool and of the code being analyzed. Then, comparisons can safely be made only between two measurements that claim to be produced by, for example, PC-Metric, Version 1.3, on code successfully assembled by the Z80 Assembler, Version 2.6.<br><br>To further ensure the accuracy of the measurements, it is essential to use the software metric tool to analyze a test case for which the values can be manually calculated for comparison. |

This will ensure that the proper attribute is measured, and is measured accurately.

Step 4. Define the segment of code that constitutes a module.

| McCabe Metric Application Analysis 9 |
|---|
| • A module can be defined trivially as a single statement, or as the next smallest independent function, or a subroutine, program, subsystem, or the entire system of programs. In general, the larger the module is defined, the larger the value of the Cyclomatic Complexity.<br><br>• The Cyclomatic Complexity measure requires that every statement can be reached from the beginning of the module and that flow continues from each statement to the end of the module (McCabe 1976). When this is not the case, the segment of code includes more than one independent module. |

A file was taken to be a module, since each file constituted a partition categorized, for certification purposes, as nonessential, essential, or critical. In general, this meant that a module was the same as an Assembly language subroutine. However, in some cases, files contained internal subroutines and/or additional external subroutines. In all cases, these were lumped together as a single module. The code consisted of 145 modules, thus defined.

Step 5. Define the rules for detecting the beginning and ending of each module.

| McCabe Metric Application Analysis 10 |
|---|
| • A module whose ending is not delineated, or is delineated in a way other than as specified in the rules, might be combined with the next module. This combination would inflate the count for the first module and would cause the following module to go undetected.<br><br>• A module whose beginning is not delineated, or is delineated in a way other than as specified in the rules, might be combined with the previous module or might be ignored. In either case, it goes undetected.<br><br>• If any of the code accidentally satisfies the rules for indicating the beginning or ending of a module, the count will likely be inaccurate.<br><br>• The rules are necessarily different for every computer language or language variation. They are very closely tied to the rules used in the interpreter, compiler, or assembler. It |

is assumed that the code has been translated, without error, by one of them. Code from a different language processor will produce unreliable results.

- Different people will probably define the rules differently. The various sets of rules may produce similar results most, but not all of the time.

The first line of each file was assumed to be the beginning of the module. The end was generated by the EOF. There was, therefore, little ambiguity for meeting the chosen module definition.

Step 6. Segregate the modules into categories of nonessential, essential, and critical so that metric results can be used to aid the different development processes required for each category of code.

Following the guidelines of RTCA/DO-178A, the developer divided the 145 modules into 84 nonessential, 59 essential, and 2 critical.

Step 7. For each module, count the number of program control flow conditions. Also, accumulate the measures for all modules in each category of criticality so that metric results can be used to aid the different development processes required for each category of code.

| McCabe Metric Application Analysis 11 |
|---|
| • The rules used to detect conditional branches must take into account the possibility that the same word of code is to be interpreted as a different operation, as determined by the context. For instance, the equal sign is used in conditions, but it is also used in assignment statements.<br><br>• It has been shown that counting control flow conditions does not produce a strictly monotonic relationship to subjective judgments of control flow complexity (Myers 1977). For instance compound conditions result in a simpler control flow than do multiple simple ones. Myers' Cyclomatic Complexity Interval measure quantifies complex conditions more accurately than McCabe's measure, but the Interval measure has not achieved much of a following. |

PC-Metric accounted for most of the problems associated with the first point of analysis mentioned above, by distinguishing between comments, declarations, and executable code. Within executable code, all unique keywords in a module are nearly always constrained to be unique in form by the interpreter, compiler, or assembler.

The Z80 code presented an additional problem: the conditional branch keywords can also be used in an unconditional form. This required that PC-Metric look

for the keyword and a following condition, before concluding that a conditional branch had been detected.

PC-Metric did not accurately count the case structure branches since each occurrence of an indirect jump was simply flagged by an additional increment to the extended complexity count. The number of cases were not counted.

Furthermore, PC-Metric had to distinguish between an asterisk in the first character position as designating a comment, but in subsequent positions as being the multiplication operator.

Step 8. Calculate the following derived measures.

a. Calculate the Cyclomatic Complexity, v(G), for a single module of code, as follows:

$$v(G) = \pi + 1$$

where $\pi$ is the number of program control flow conditions.

| McCabe Metric Application Analysis 12 |
|---|
| • This metric calculates the maximum number of linearly independent paths required to form a basis set for all the paths in a structured program. It is assumed that the program has an additional path from its last statement back to its first statement. Any possible path through the program can be expressed as a linear combination of some subset of this basis set of paths (McCabe 1976).<br><br>• A set of program path test cases cannot claim path coverage if it does not consist of at least the number of paths in the basis set. If it does, it still may not be a basis set. Although a basis set provides path coverage on only a partial path basis, at least the set of partial paths is finite. Full path coverage assumes that every possible path, from the beginning to the ending of the module, has been generated. A program with loops can have an infinite number of such paths (Prather 1983).<br><br>• This measure counts all conditions present, whether necessary or not. For example, code may contain a condition in a context where it will always be true. The Cyclomatic Complexity counts the number of conditions present in the algorithm, as implemented.<br><br>• This formula for structured programs requires that for every branching node there is exactly one collecting node, and that the program has unique entry and exit nodes. Exceptions to the rules always increase v(G). As long as there are relatively few exceptions to these rules, the Cyclomatic Complexity will be not |

be substantially affected. If there are many exceptions, the impact of multiple exits can be corrected by computing the following:

$$v(G) = \pi - s + 2$$

where "s" is the number of exits from the module (Harrison 1984). Of course, counting the number of exits requires additional analysis. If the code is primarily structured, the inaccuracy of the original equation is probably not significant enough to warrant the extra effort.

• The Cyclomatic Complexity of a collection of independent modules, as in a main program and its subroutines, is simply the sum of the Cyclomatic Complexities for each module. It is improper to calculate the total Cyclomatic Complexity by analyzing all the modules as if they were one continuous sequence of code. In this case, the value is deflated by the number of modules in the collection minus one.

The Cyclomatic Complexity for each module of the code is listed in appendix C. The Cyclomatic Complexity for each category of code is shown in table 3.5-1. PC-Metric calculated these totals in the manner just warned against in the last point of Analysis Box 12. This can be seen by comparing the critical code total to the individual counts. Thus, the values in table 3.5-1 are understated by the number of modules in the category minus one.

TABLE 3.5-1. CYCLOMATIC COMPLEXITY, BY CODE CATEGORY

| | Nonessential | Essential | Critical |
|---|---|---|---|
| v(G) | 821 | 730 | 13 |

b. The Essential Complexity, ev(G), can be calculated from the control graph of a prograr as follows:

$$ev(G) = v(G) - m$$

where "m" is the number of proper subgraphs, those structured subgraphs which contain no further subgraphs.

| McCabe Metric Application Analysis 13 |
|---|
| • This is a measure of the amount of unstructured code in a program. The smaller the value of ev, the greater the proportion of structured code. The Essential Complexity of a |

structured program is one. Greater values for the Essential Complexity of a module indicate the size of the non-structured portion of the code; that code that does not conform to the proper subgraph structure (McCabe 1976).

• Like the Cyclomatic Complexity, this measure also reflects the state of the algorithm, as implemented. It does not indicate whether the conditions, as implemented, are necessary to the solution of the problem that the algorithm claims to solve.

From the program control graphs in appendix C, it was determined that the first module contained one proper subgraph and the second module contained four. The values for the critical modules are shown in table 3.5-2.

TABLE 3.5-2. ESSENTIAL COMPLEXITY DATA

| | v(G) | m | ev(G) |
|---|---|---|---|
| CRITI001 | 6 | 1 | 5 |
| CRITI002 | 8 | 3 | 5 |

Step 9. Calculate the SQM defined in step 1 of this case.

a. Calculate the Maintainability quality factor, as follows:

$$\text{Maintainability} = \frac{\text{\# of modules that met the goal}}{\text{total number of modules}} \text{ \% compliance}$$

The value for the code can be calculated from the data in appendix C. (The noncompliant data are flagged with an asterisk.)

Maintainability = 125/145
= 86.2% compliance

Thus, the maintainability goal was met on the first check. The 20 noncompliant modules were reviewed, and some were reduced by separating some of the code into independent modules.

b. Based on the total Cyclomatic Complexity units for the code, the 21 programmers were assigned modules to debug, as follows:

Nonessential modules: 11 programmers

Complexity units = 821/11
= 74.6 units per programmer

Essential and critical modules: 10 programmers

Complexity units = 743/10
= 74.3 units per programmer

This assignment gave less than a one percent variation in programmer load. Similar calculations were made to determine how many modules to assign to each programmer.

c. The number of test cases required by McCabe's Cyclomatic Complexity for ensuring branch coverage was generally reasonable. The highest of the average values for each of the three categories was 14. However, 20 had values that exceeded 22. For these, alternate methods for determining the required number of test cases were used.

d. Calculate the Testability, as follows:

$$\text{Testability} = \frac{\text{\# of modules with } v(G) \leq 10}{\text{total \# of modules}} \text{ \% compliance}$$

The value for the critical modules of the code can be calculated from the data in appendix C.

Testability = 2/2
= 100% compliance

e. Calculate the Integrity, as follows:

$$\text{Integrity} = \frac{\text{\# of modules with } ev(G) \leq 5}{\text{total \# of modules}} \text{ \% compliance}$$

The value for the critical modules of the code can be calculated from the data in table 3.5-2.

Integrity = 2/2
= 100% compliance

### 3.5.2 Certification Conclusions

For certification purposes, it should be determined whether the metrics, as applied, were reliable indicators of the properties measured. It should also be determined whether such measures equivalently fulfill the requirements of the certification guidelines. This determination can be based on the experience of the developer, the experimentation performed by the developer, or the experimentation that the developer references, as documented in the certification package. It can also be based on the experience of the CE or on references by the CE to the handbook data sheets.

Based on these inputs, the SQM performed are evaluated to see if they help the CE answer any of the certification determinations posed in section 3.3. This

evaluation addresses each SQM, in the order in which they were specified in step 1.

a. The use of the Maintainability SQM indicates that the developer subjected the code to a disciplined, quantitatively controlled development process.

| *McCabe Metric Application Analysis 14* |
|---|
| • First, the CE should compare this software development project to those of the experiments on Maintainability. The CE would then likely conclude that this SQM offers some assurance that the software was developed according to a disciplined approach, as recommended by RTCA/DO-178A, paragraph 1.1. |

b. Since the debugging load was balanced using the Cyclomatic Complexity metric, the developer maintains that it is more likely that the modules are consistently debugged and that they operate properly. Otherwise, some programmers would have been overloaded and their search for bugs would have been truncated in order to keep on schedule.

| McCabe Metric Application Analysis 15 |
|---|
| • First, the CE should compare this software development project to those of the experiments on debugging times. The CE would then likely conclude that this SQM offers some assurance that the software will operate at the required level of performance. |

c. Since the testing of critical modules was regulated by the Cyclomatic Complexity, the developer maintains that branch coverage has been achieved. Otherwise, it would be possible to test an insufficient number of cases.

| McCabe Metric Application Analysis 16 |
|---|
| • The CE would likely conclude that this SQM offers some assurance that test coverage is obtained, as recommended by RTCA/DO-178A, paragraph 6.2.5.3.2. However, this metric gives the CE no assurance that the developer picked unique and comprehensive test cases. The developer may have performed the specified number of test cases, but this SQM does not ensure that the test cases used were a comprehensive set. |

d. Since the critical modules met the Cyclomatic Complexity threshold, the developer maintains that they are more testable than the other modules. Testing will not be inhibited by undetected coding errors.

| McCabe Metric Application Analysis 17 |
| --- |
| • First, the CE should compare this software development project to those of the experiments on testing time. The CE could then conclude that this SQM offers some assurance that the critical code has been subjected to a more demanding development process, appropriate to the software level, as recommended by RTCA/DO-178A, paragraph 6.2.1. However, this metric is not as assuring as others since it claims to be an absolute measure, rather than a relative measure. Particularly, the critical code satisfied the absolute level, as coded. It is possible that the nonessential and essential code would also have satisfied the absolute level, as coded. |

e. Since the critical modules met the Essential Complexity threshold, the developer maintains that they have more integrity than the other modules.

| McCabe Metric Application Analysis 18 |
| --- |
| • First, the CE should compare this software development project to those of the experiments on Essential Complexity. The CE could then conclude that this SQM offers some assurance that the critical code has been subjected to a more demanding development process, appropriate to the software level, as recommended by RTCA/DO-178A, paragraph 6.2.1. However, this metric is not as assuring as others since it was used as an absolute measure, rather than a relative measure. Particularly, the critical code satisfied the absolute level, as coded. It is possible that the nonessential and essential code would also have satisfied the absolute level, as coded. |

McCabe's metrics did not address whether any of the requirements stated for this software were fulfilled. Thus, the developer did not use SQM in the requirements analysis recommended by RTCA/DO-178A, paragraph 6.2.5.3.1.

In this example, the developer used both software metrics and SQM to analyze the flight control code being developed. In either case, the metrics provided a quantitative analysis of the software product. This analysis was used to assess and sometimes improve the software development product and process.

### 3.6 Case #3: Software Quality Metrics Based on RADC's Software Metrics

In this case, the developer used SQM of the type developed by the RADC to assess the code quantitatively. In particular, the methodology developed by Bowen, Wigle, and Tsai (1985) was used. The developer followed the steps described in the next section.

### 3.6.1 Metric Application and Analysis

Step 1. Define the groups of modules that have different quality factor requirements and specify the quality factors of concern for each, based on the functions that each group supports.

| RADC Metric Application Analysis 1 |
| --- |
| • The SQM developed by RADC are based on a predefined set of user-oriented quality factors. From this set, the developer selects the factors which apply to each group of modules.<br><br>These factors have been carefully defined to produce a comprehensive set of relatively independent factors. Furthermore, the SQM that use them have been shown to give meaningful, repeatable results in experiments. However, the CE can accept these SQM as reliable only if the code being developed belongs to the same population of code on which the experiments have been performed. Otherwise, the SQM relationships are based solely on intuition.<br><br>• The quality factors are not totally independent of one another. Some are complementary. A factor that is complemented by another but selected by itself will result in a false indication of the total quality present. All factors complementary to one that was specified should also be specified. |

For this avionic code project, the developer chose to lump all the code into one group, since it all contributes to the single function: flight control. The developer specified that Maintainability, Expandability, and Reusability were to be measured for all the code. Because the developer was using SQM for the first time, the more crucial factors, Reliability, Correctness, and Verifiability, were assured by conventional testing strategies, rather than by SQM. Since the strength of the factors complementary to Maintainability, namely Reliability and Correctness, was also assured, the SQM results for Maintainability would be meaningful.

Maintainability was considered important because flight control code is usually modified many times during the lifetime of the aircraft. Expandability was important because the same code was to be used in several models of the same aircraft, some of which required many additional features. Reusability was important because reusing code is profitable.

Step 2. Specify the level of quality considered acceptable for each factor selected for each group, based on the importance of that factor for the particular system function served, a survey of important factors, the factor interrelationships, and the impact on costs.

| RADC Metric Application Analysis 2 |
|---|
| • Some of the factors have negative interrelationships. When these are selected, high levels cannot be achieved for all of them; the levels must be balanced. This compromise results in lower quality factor scores than would be otherwise attainable.<br><br>• When factor scores are specified and calculated, numeric scores are used. However, when the quality goals are chosen, they should be based on a qualitative approximation of the importance of each factor. For instance, one must choose whether the code must have Excellent, Good, or Acceptable Reliability. These are the qualitative levels recommended for the Air Force. They are correlated to the numeric scores as follows (Lasky, Kaminsky, and Boaz 1990):<br><br>Excellent ≥ 0.9<br>Good ≥ 0.8<br>Acceptable ≥ 0.7 |

The level of quality considered acceptable varied for the software criticality categories. The developer made the specifications shown in table 3.6-1.

TABLE 3.6-1. QUALITY FACTOR GOALS, BY CODE CATEGORY

| Quality Factor | Nonessential | Essential | Critical |
|---|---|---|---|
| Maintainability | Excellent | Excellent | Good |
| Expandability | Excellent | Excellent | Good |
| Reusability | Acceptable | Good | Acceptable |

The Maintainability quality factor was especially important for the non-critical code, since it is subjected to the most revision. This quality factor was not as important for critical code. Critical code is usually subjected to the most thorough testing, and therefore requires less maintenance. It receives fewer enhancements because the certification process for critical code is so rigorous that a developer avoids making any changes which would require recertification.

The Expandability quality factor was important because it was planned that the code would be used in all the models of the particular aircraft. It was especially important for the non-critical code to be expandable because most of the enhancements were to be made to non-critical code.

The Reusability quality factor was important for all the code because reusing code saves money. This more demanding goal was set for essential code which must be reusable with a minimum number of modifications, since modifications to

essential code must all be recertified. In general, critical code must be written from scratch for each application. There is less demand that it be reusable.

| RADC Metric Application Analysis 3 |
|---|
| • These three factors are complementary to one another. Therefore, it was reasonable for the developer to expect to attain all the quality goals specified.<br><br>• The CE should note that the development process is being subjected to the disciplined approach of quantitative control for the purpose of ensuring a process commensurate with the code criticality, as required by RTCA/DO-178A. |

Step 3. Identify the criteria assigned to each selected quality factor and specify a weighting factor for each, based on the importance of the criterion for the particular function served, the interrelationships of the criterion with the selected factors, and the impact on costs.

| RADC Metric Application Analysis 4 |
|---|
| • The criteria are a predefined set of software characteristics assigned to the quality factors to which they contribute. These criteria have been carefully defined to produce a comprehensive set of relatively independent criteria for each factor. They have also been shown to give meaningful, repeatable results when used in SQM experiments. The developer assigns a weighting factor, from zero to one, to multiply by the score of each criterion. A weight of zero is assigned to those criteria that are not to be counted. The sum of the weighting values for each factor must be one.<br><br>• The criteria are not totally independent of other factors. Some may contribute to more than one of the selected factors. These may each be weighted lower so that one characteristic does not contribute too heavily to the total quality. Some may conflict with other factors. These may be weighted lower so that the quality goal for each factor can be attained, based on the strength of other criteria. |

The Maintainability criteria, with the specified weighting factors, are shown in table 3.6-2.

TABLE 3.6-2. MAINTAINABILITY CRITERIA

| Criterion | Weight |
|---|---|
| Consistency | 0.0 |
| Visibility | 0.0 |
| Modularity | 0.5 |
| Self-Descriptiveness | 0.5 |
| Simplicity | 0.0 |

The Expandability criteria, with the specified weighting factors, are shown in table 3.6-3.

TABLE 3.6-3. EXPANDABILITY CRITERIA

| Criterion | Weight |
|---|---|
| Augmentability | 0.0 |
| Generality | 0.6 |
| Virtuality | 0.0 |
| Modularity | 0.2 |
| Self-Descriptiveness | 0.2 |
| Simplicity | 0.0 |

The Reusability criteria, with the specified weighting factors, are shown in table 3.6-4.

TABLE 3.6-4. REUSABILITY CRITERIA

| Criterion | Weight |
|---|---|
| Application Independence | 0.0 |
| Document Accessibility | 0.0 |
| Functional Scope | 0.3 |
| Generality | 0.3 |
| Independence | 0.0 |
| System Clarity | 0.0 |
| Modularity | 0.2 |
| Self-Descriptiveness | 0.2 |
| Simplicity | 0.0 |

Each criterion was of equal relevance to the nonessential, essential, and critical code. Therefore, the function of the code was not a factor in determining the weights. Other concerns determined the specified weights.

In order not to complicate the learning process, the developer chose the minimum set of criteria that would exercise the different ways that criterion scores are combined. To reduce the cost of calculating the metrics, criteria that were based on easily automatable metrics were favored. Other criteria were assigned a zero weight.

Generality and Self-Descriptiveness were chosen because they represent criteria that are the most cost-effective to measure; they contribute to more than one factor. Modularity was chosen for that reason, and also because it represents a metric that is based on a combination of metric elements. Functional Scope was chosen as representative of criteria that do not contribute to any other factor.

Modularity and Self-Descriptiveness were weighted less heavily for the Expandability and Reusability factors so that those factors could be distinguished from Maintainability by the other criteria. This also keeps these criteria from having too large an impact on the overall quality. Since none of the chosen criteria exhibit noticeable interrelationships with the other factors measured, there was no reason to otherwise weight the criteria unevenly.

| RADC Metric Application Analysis 5 |
| --- |
| • This weight specification is a fairly reasonable one. It is especially important that the developer recognized the need to distinguish the factors from each other by emphasizing the unique criteria for each. It would have been even better if the number of unique criteria had been increased to match or exceed the number of criteria common to other factors. The developer also simplified the weighting process by using criteria that did not negatively impact any of the measured factors. |

Step 4. Identify the metrics assigned to each criterion and specify which metric elements are to be measured, based on the applicability of the metric element to the particular system and the cost of measuring the metric element.

| RADC Metric Application Analysis 6 |
| --- |
| • The metrics are a predefined set of software-oriented details assigned to the criterion to which they contribute. These metrics have been carefully defined to produce a set of independent metrics for each criterion. They have also been shown to given meaningful, repeatable results when used in SQM in experiments. The developer chooses to use those that apply and that will not cost more to measure than is gained from doing so. |

- Metric elements must all be defined in the same sense, such that an increase in any metric score produces an increase in the associated quality factor score.

- Because the factor scores are based on these metric questions, the SQM goals can be used to improve the software. If a goal is not met, simply fix the individual metric element failures to achieve a higher quality.

The metric elements to be calculated were selected from Metric Worksheets 4A and 4B (Bowen, Wigle, and Tsai 1985). Each criterion is usually based on many metrics, and each metric is usually based on several metric elements. In this case, only those metric elements that were easily automated were selected. Thus, the metric, MO.1, was based on two elements and SD.2, GE.1, and FS.1 were based on only one. These metric elements are listed in table 3.6-5. With these preliminary selections, development began. After the code was written, the following metric steps were taken.

TABLE 3.6-5. SELECTED METRIC ELEMENTS

| Criterion | Metric Element Number and Description |
|---|---|
| Modularity | MO.1(3) - Are the estimated lines of source code for this unit 100 lines or less, excluding comments? |
| | MO.1(7) - Is control always returned to the calling unit when execution is completed? |
| Self-Descriptiveness | SD.2(1) - Are there prologue comments which contain all information in accordance with the established standard? |
| Generality | GE.1(1) - How many units are called by more than one other unit? |
| Functional Scope | FS.1(2) - Is a description of the function(s) provided in the comments? |

Step 5. Define the rules for identifying and counting metric element objects and for calculating factor scores. The rules may be contained in a lookup table or may consist of rule statements.

| RADC Metric Application Analysis 7 |
|---|
| • The occurrence of a metric element object that was not anticipated by the rules will deflate the element count. For instance, if a comment count detects comment lines but not in-line comments, the measure of the amount of commenting would be deflated.<br><br>• A metric element object manifested in a way that was not anticipated by the rules will deflate the element count. For instance, if a nesting indentation check required that each level be indented with three spaces, but the code was consistently indented with a tab of three spaces, the code indentation would not be counted.<br><br>• Code that accidentally satisfies the rules for detecting a metric element object will inflate the element count. This would only occur if the rules were less stringent than the language processor.<br><br>• The rules are necessarily different for every computer language or language variation. They are very closely tied to the rules used in the interpreter, compiler, or assembler. It is assumed that the code has been translated, without error, by one of them. Code from a different language processor will produce unreliable results.<br><br>• Different people will probably define the rules differently. The various sets of rules may produce similar results most, but not all of the time. |

For this case, the developer wrote a program to calculate the previously selected RADC metric elements for the Z80 Assembly code. All the rules were imbedded in the program statements. None were in a table or otherwise user-defined.

| RADC Metric Application Analysis 8 |
|---|
| • The calculations produced by a software metric tool reflect all the decisions built into the program code. The only way to ensure a repeatable, predictable measure, for comparison to some previous measure, is to maintain configuration control of the software metric tool and of the code being analyzed. Then, comparisons can be safely made only between two measurements that claim to be produced by, for example, RADC-Metric, Version 1.3, on code successfully assembled by the Z80 Assembler, Version 2.6.<br><br>It is essential to use the software metric tool to analyze a test case for which the values can be manually calculated for |

comparison. This check will ensure that the proper attribute is measured accurately.

Step 6. Define the segment of code that constitutes a module.

| RADC Metric Application Analysis 9 |
| --- |
| • A module can be defined trivially as a single statement, or as the next smallest independent function, or a subroutine, program, subsystem, or the entire system of programs. These metrics can be applied at any level, but the detail of the calculations makes their application prohibitive at the lower levels. For code developed under DOD-STD-2167, it is not recommended that they be applied any lower than the level of the Computer Software Configuration Items (CSCI) (Lasky, Kaminsky, and Boaz 1990). |

A file was taken to be a module, since each file constituted a partition categorized, for certification purposes, as nonessential, essential, or critical. In general, this meant that a module was the same as an Assembly language subroutine. However, in some cases, files contained internal subroutines and/or additional external subroutines. In all cases, these were lumped together as a single module. The code consisted of 145 modules, thus defined.

Step 7. Define the rules for detecting the beginning and ending of each module.

| RADC Metric Application Analysis 10 |
| --- |
| • A module whose ending is not delineated, or is delineated in a way other than as specified in the rules, might be combined with the next module. This combination would inflate the metric for the first module and cause the following module to go undetected.<br><br>• A module whose beginning is not delineated, or is delineated in a way other than as specified in the rules, might be combined with the previous module or might be ignored. In either case, it goes undetected.<br><br>• If any of the code accidentally satisfies the rules for indicating the beginning or ending of a module, the metric will likely be inaccurate.<br><br>• The rules are necessarily different for every computer language or language variation. They are very closely tied to the rules used in the interpreter, compiler, or assembler. It is assumed that the code has been translated, without error, by |

one of them. Code from a different language processor will produce unreliable results.

- Different people will probably define the rules differently. The various sets of rules may produce similar results most, but not all of the time.

The first line of each file was assumed to be the beginning of the module. The end was generated by the EOF. There was, therefore, little ambiguity for meeting the chosen module definition.

Step 8. Segregate the modules into categories of nonessential, essential, and critical code so that metric results can be used to aid the different development processes required for each category of code.

Following the guidelines of RTCA/DO-178A, the developer divided the 145 modules into 84 nonessential, 59 essential, and 2 critical. These three categories were treated as CSCIs and each module was taken to be a unit, as defined by this methodology.

Step 9. Compute all the metrics for each module. Also, accumulate the measures for all modules in each category of criticality so that metric results can be used to aid the different development processes required for each category of code.

| RADC Metric Application Analysis 11 |
| --- |
| • The rules used to detect a particular metric element object must take into account the possibility that the same word of code is to be interpreted differently, as determined by the context. For instance, the equal sign is used in conditions, but it is also used in assignment statements. |

The custom program accounted for most of the problems associated with the above analysis by distinguishing between comments, declarations, and executable code. Within executable code, all unique keywords in a module are nearly always constrained to be unique in form by the interpreter, compiler, or assembler.

Furthermore, the program had to distinguish between an asterisk in the first character position as designating a comment, but in subsequent positions as being the multiplication operator.

The metric element values for each category of code were calculated as follows:

$$\text{Metric Element} = \frac{\text{the sum of the unit scores}}{\text{the number of applicable units in the CSCI}}$$

where each unit is scored with either a calculated value normalized to be between zero and one, or with a one for a yes answer and a zero for a no answer. The values are shown in table 3.6-6. They were calculated from the data in appendix C.

TABLE 3.6-6. METRIC ELEMENT SCORES, BY CODE CATEGORY

| Metric | Nonessential | Essential | Critical |
|---|---|---|---|
| MO.1(3) | 45/84 = 0.5 | 28/59 = 0.5 | 2/2 = 1.0 |
| MO.1(7) | 75/84 = 0.9 | 48/59 = 0.8 | 2/2 = 1.0 |
| SD.2(1) | 44/84 = 0.5 | 1/59 = 0.0 | 0/2 = 0.0 |
| GE.1(1) | 11/84 = 0.1 | 11/59 = 0.2 | 0/2 = 0.0 |
| FS.1(2) | 84/84 = 1.0 | 56/59 = 1.0 | 2/2 = 1.0 |

The metric scores can then be calculated form the element scores of table 3.6-6. The Modular Implementation metric, MO.1, is calculated as follows:

$$\text{MO.1} = \frac{\text{the sum of the numerators of MO.1(3) and MO.1(7)}}{\text{the sum of the denominators of MO.1(3) and MO.1(7)}}$$

The scores for the other metrics are simply equal to the value of the single metric elements composing each. The values for each category of code are shown in table 3.6-7.

TABLE 3.6-7. METRIC SCORES, BY CODE CATEGORY

| Criterion | Nonessential | Essential | Critical |
|---|---|---|---|
| MO.1 | 120/168 = 0.7 | 76/118 = 0.6 | 4/4 = 1.0 |
| SD.2 | 0.5 | 0.0 | 0.0 |
| GE.1 | 0.1 | 0.2 | 0.0 |
| FS.1 | 1.0 | 1.0 | 1.0 |

RADC Metric Application Analysis 12

- When a metric is based on several metric elements, several methods can be used to combine the metric element scores into a metric score, according to the following procedures and considerations:

1. Direct ratio scoring adds all the numerator scores and divides this sum by the sum of the denominator scores. This

has the effect of treating every individual score of every metric element equally. If one of the composing metric element objects occurs much more frequently than the others, (i.e., the denominator is much larger) then the metric score primarily reflects that metric element. The others contribute insignificantly to the metric score.

This attribute of direct ratio scoring is helpful if it draws attention to a single frequently occurring code weakness among many code features that are consistently and properly implemented. It is harmful if a single frequently occurring code strength masks many code features that are not consistently or properly implemented. The CE should examine the data to determine if any negative effects have occurred which would mask the true quality.

When each metric element has the same value for the denominator, the value calculated by this method is the same as that calculated by second order averaging.

2. Second order averaging adds all the metric element scores together and divides the sum by the number of metric elements. This has the advantage of giving each metric element an equal contribution to the metric score. If one of the composing metric elements is especially weak or strong, it will still have an effect on the average, but not in proportion to the number of occurrences of the code feature measured.

   This attribute of second order averaging is helpful if only the percentage of errors is important. It is harmful if the number of errors of a certain type is important. The CE should examine the data to determine if any of the negative effects have occurred, masking the true quality.

3. Weighting factors can be introduced to tailor the impact a particular metric element has on the metric score. Each metric element is multiplied by a weight, between zero and one, before the summation is performed. Normalization is achieved by assuring that the sum of the weights is one. This can be applied to either of the two types of averaging. When weighting factors are used, the CE should examine the weights applied to each metric element to determine if the quality is measured accurately.

Step 10. Calculate the criterion scores, as follows:

The scores for the criteria, Modularity, Self-Descriptiveness, Generality, and Functional Scope, are simply equal to the value of the single metric composing each. The values for each category of code are shown in table 3.6-8.

TABLE 3.6-8. CRITERIA SCORES, BY CODE CATEGORY

| Criterion | Nonessential | Essential | Critical |
|---|---|---|---|
| MO | 0.7 | 0.6 | 1.0 |
| SD | 0.5 | 0.0 | 0.0 |
| GE | 0.1 | 0.2 | 0.0 |
| FS | 1.0 | 1.0 | 1.0 |

| RADC Metric Application Analysis 13 |
|---|
| • The methods of combining metric element scores to get a metric score, as discussed above, also apply to combining metric scores to get a criterion score. |

Step 11. Calculate the SQM selected in step 1, and defined in step 3, as follows:

Maintainability = (0.5 x MO) + (0.5 x SD)

Expandability = (0.6 x GE) + (0.2 x MO) + (0.2 x SD)

Reusability = (0.3 x FS) + (0.3 x GE) + (0.2 x MO) + (0.2 x SD)

The values for each category of code are shown in table 3.6-9.

TABLE 3.6-9. SOFTWARE QUALITY METRICS SCORES, BY CODE CATEGORY

| Quality Factor | Nonessential | Essential | Critical |
|---|---|---|---|
| Maintainability | 0.6 | 0.3 | 0.5 |
| Expandability | 0.3 | 0.2 | 0.2 |
| Reusability | 0.6 | 0.5 | 0.5 |

| RADC Metric Application Analysis 14 |
|---|
| • The methods of combining metric element scores to get a metric score, as discussed above, also apply to combining criterion scores to get a factor score. The above calculations are an example of combining by weighting second order averages. |

- Criteria can contribute to more than one factor, as shown in the calculations above. (This is not the case for the RADC software metrics. Each software metric contributes to only one criterion.) When this occurs, a particular criterion can have an exaggerated contribution to the selected SQM. Criteria dampening is an additional method for combining criteria. This method corrects the problem. Criteria dampening can be applied to either the direct ratio or second order averaging scores, whether weighted or not.

Criteria dampening scales back the contribution a particular criterion makes to the factor scores. Each score for that criterion is multiplied by the proportion that score represents of the total score for that criterion for all factors. When the criterion scores are direct ratios, criteria dampening is achieved by multiplying each criterion score numerator by the ratio of its denominator to the sum of the denominators. When the criterion scores are second order averages, criteria dampening is achieved by dividing each criterion score by the number of factors to which it contributes. In the above calculations, each Modularity and Self-Descriptiveness score would be divided by three and each Generality score would be divided by two. The weighting that was done in step 3 exhibits a certain amount of criteria dampening that was done intuitively, since the multiply-occurring Modularity and Self-Descriptiveness criteria were weighted so low. Nevertheless, the criteria dampening method was not applied.

The Maintainability quality factor was not initially achieved for any category. The metric element scores were examined to determine how each module failed. In general, the Modularity criterion scores were good, but the Self-Descriptiveness scores were poor. The critical code was brought to a Maintainability measure of one by simply entering "Not Applicable" for header fields that did not apply to a failing module. This entry was also added, where necessary, to the remaining code. This fix was sufficient to bring the nonessential code into compliance, since it needed to have only good maintainability. The essential modules required more work.

The essential code contained many modules which greatly exceeded 100 lines of source code, and a few modules that transferred control to other modules, rather than returning to the calling module. Seven modules were fixed by reducing the code or restructuring the returns. This was sufficient to bring the essential code into compliance.

| RADC Metric Application Analysis 15 |
| --- |
| • Notice that this SQM not only quantified the quality of the software product, but also directly indicated how to improve the product. This indication makes an SQM particularly useful.<br><br>• If the long modules had been broken into subroutines rather than shortened, it would have produced more modules that were called by only one other module, thus lowering the score for the Generality criterion. |

The Expandability quality factor was not initially met for any category. The metric scores were examined to determine how each module failed. The weakness was with the Self-Descriptiveness and Generality criteria. The previous fix for the Self-Descriptiveness problem was applied but was insufficient to bring the factor score into compliance. Furthermore, the Generality criterion score was not able to be improved for any of the categories. The application was not general enough to allow many modules to be called by more than one other module. The developer chose to make no further improvement. The Expandability goal was not met.

| RADC Metric Application Analysis 16 |
| --- |
| • From the description of the Generality metric used, the CE can determine that the expandability required for this code will not be significantly impacted by a low score for this metric. The expansions required by the various models of the aircraft would occur within single modules or by the addition of modules. The existing module interaction would remain intact. |

The Reusability quality factor was not initially met for any category. The metric scores were examined to determine how each module failed. The Self-Descriptiveness and Generality criteria scores were particularly low.

Again, the Self-Descriptiveness fix was applied to bring its criterion score to one. This fix sufficed to bring the critical code into compliance and the nonessential code very close to compliance. Since the particular Generality metric used reflects the specificity of the calling environment, rather than the generality of the subroutines, the developer decided to deemphasize that criterion and emphasize the Functional Scope criterion. Furthermore, 100 percent compliance was achieved for the Functional Scope metric. By weighting Functional Scope by 0.5 and Generality by 0.1, the non-critical code was brought into compliance.

| RADC Metric Application Analysis 17 |
|---|
| • Rather than arbitrarily changing the weight applied to the Generality criterion, the developer could have added more generality metrics in order to exhibit reusability qualities that were present in the code, but not recognized. This probably would have been sufficient to overcome the low score on the GE.1(1) metric element. In general, a criterion should be based on several metrics. Otherwise, the criterion is being reduced to the presence of a single feature of code. |

The final values for all the SQM are shown in table 3.6-10.

TABLE 3.6-10. FINAL SOFTWARE QUALITY METRICS SCORES, BY CODE CATEGORY

| Quality Factor | Nonessential | Essential | Critical |
|---|---|---|---|
| Maintainability | 0.9 | 0.9 | 1.0 |
| Expandability | 0.4 | 0.4 | 0.4 |
| Reusability | 0.8 | 0.9 | 0.9 |

### 3.6.2 Certification Conclusions

For certification purposes, it should be determined whether the metrics, as applied, were indeed reliable indicators of the properties measured. It should also be determined whether such measures equivalently fulfill the requirements of the certification guidelines. This determination can be based on the experience of the developer, the experimentation performed by the developer, or the experimentation that the developer references, as documented in the certification package. It can also be based on the experience of the CE or on references by the CE to the handbook data sheets.

Based on these inputs, the SQM performed are evaluated to see if they help the CE answer any of the certification determinations posed in section 3.3. This evaluation addresses each of the questions, in the order in which they were posed.

a. The use of these SQM indicates that the developer subjected the code to a disciplined, quantitatively controlled development process.

| RADC Metric Application Analysis 18 |
| --- |
| • After comparing this software development project to those of the experiments on RADC's metrics, the CE would likely conclude that these SQM offer some assurance that the software was developed according to a disciplined approach, as recommended by RTCA/DO-178A, paragraph 1.1. |

b. The developer set more demanding goals for each level of criticality. These goals regulated the amount of improvement required to make the modules meet the goals set for them.

| RADC Metric Application Analysis 19 |
| --- |
| • After comparing this software development project to those of the experiments on RADC's metrics, the CE would likely conclude that these SQM offer some assurance that the software was developed by a process appropriate to the software level, as recommended by RTCA/DO-178A, paragraph 6.2.1. Regardless of the experiments, the SQM resulted in inconsistencies being detected and corrected. |

c. RADC's metrics did not address whether any of the requirements stated for this software were fulfilled. Thus, the developer did not use SQM in the requirements analysis recommended by RTCA/DO-178A, paragraph 6.2.5.3.1.

d. This developer could not use SQM to substantiate test coverage, as defined in RTCA/DO-178A, paragraph 6.2.5.3.2, because RADC's software metrics do not measure anything that substantiates structural test coverage.

e. Maintainability and Expandability were partly performance requirements for this software. The developer submitted these two SQM as indicators of the extent to which those qualities were present in the delivered software.

| RADC Metric Application Analysis 20 |
| --- |
| • After comparing this software development project to those of the experiments on RADC's metrics, the CE would likely conclude that these SQM offer some assurance that the software will operate at the required level of performance. |

In this example, the developer used SQM to analyze the flight control code being developed. This analysis was used to assess and sometimes improve the software development product and process.

## APPENDIX A - SOFTWARE METRIC DATA SHEETS

This appendix contains individual data sheets on each of the most prevalent software metrics encountered in this study. The data sheets are a summary of the concepts, constraints, criticisms, and other data needed to understand and critically evaluate the SQM that rely on these metrics.

Although the data sheets are not primarily intended as a guide for applying the metrics, some of the data sheets contain sufficient information that the metric could be properly and fully calculated by following the data sheet. On the other hand, some of the metrics are so extensive that the data sheets could not encompass all the necessary parts. In either case, the Technical Report, Software Quality Metrics (N. VanSuetendael and D. Elwell 1991), discusses each of them in detail. See the references listed on the data sheet for a complete coverage of a particular metric.

Each data sheet covers only one metric and they all contain the same fields of information. The fields are defined only as they apply to measuring source code. Many of the metrics can be applied more broadly, but the other possibilities are not addressed here.

The "SQM RELATIONSHIPS" field lists those quality factors that are known to be related to the particular software metric. This serves as a cross-reference to the applicable Software Quality Factor Data Sheet of appendix B.

The "TOOL" information is supplied on an as-available basis. It is not to be taken as a comprehensive list of tools available. Nor is the information to be taken as a recommendation.

Metrics that share a common heritage of theory are grouped together by family. The family name is given at the top of each sheet. The data sheets are arranged alphabetically by family name. Within each family they are arranged in the order in which they build upon each other, from the fundamental level to the highest level. When there is no hierarchical relationship within a family, the metric data sheets are arranged alphabetically by metric name.

## Function Points

This metric calculates a weighted sum of the number of elementary functions supported by a module of source code. The functions are categorized into five types. The number of Function Points (FP) provided to the user by a module of source code is given by the following rule:

RULE: FP = FC x PCA

FC is the function count produced as follows:

Function Count = Number of External Inputs x 4, plus
Number of External Outputs x 5, plus
Number of Logical Internal Files x 10, plus
Number of External Interface Files x 7, plus
Number of External Inquiries x 4

where the weights given were determined experimentally. They are the values that made the measure most accurately reflect the perceived amount of function delivered in real projects of average complexity. PCA is the Processing Complexity Adjustment Factor, calculated as follows:

PCA = 0.65 + (0.01 x PC),

where PC is the Processing Complexity. It is based on an estimate of the degree of influence that each of 14 specified application characteristics (Albrecht and Gaffney 1983) has on the complexity of the functions counted. Each characteristic is assigned a "0", "1", or "2" degree of influence. Then the PC is the sum of these 14 numbers. This produces a PCA which adjusts the function count by ±35 percent for application and environment complexity.

DOMAIN: This measure only applies to source code that implements an algorithm; the code must perform some function that relates to user inputs, user outputs, internal files, interface files, and/or user inquiries.

RANGE: This metric produces a real number greater than or equal to 1.3, the number of function points in the simplest algorithm which must contain an input and an output (assuming the lowest possible weight of one for each category and the lowest possible PCA of 0.65).

NECESSARY CONDITIONS: The criteria for determining the inputs, outputs, internal files, interface files, and inquiries from code must be defined for the context. The criteria for determining whether they are internal or external must be defined for the context. In order to use the weights given, the practitioner must also use Albrecht's definitions for the five types of function (Albrecht and Gaffney 1983).

QUALIFICATIONS: This metric has been validated primarily in a data processing system environment, rather than a real-time flight control environment. The given weights probably do not apply to code in avionic equipment.

The given weights are for programs that fall into a qualitatively chosen category of average complexity. Different sets of weights are given for simple or complex programs (Albrecht and Gaffney 1983).

The FP of an application is not necessarily equal to the sum of the FP of each of its sub-applications. Thus, FP is not an additive quantity.

CRITICAL ANALYSIS: This measure is not necessarily strictly monotonic or repeatable. A program reader may erroneously identify an input, output, internal file, interface file, or inquiry. In this case, an increase in the measure does not represent an increase in the amount of function. Furthermore, a different reader will likely judge the attributes differently. This effect has been observed in experimental analysis (Low and Jeffery 1990).

A problem with repeatability can also result from the arbitrariness of establishing application boundaries. The larger the application size, the more functions will be included. Furthermore, as the boundary is enlarged files will shift from being external interface files to being logical internal files. It is important that all measurements be based on an unambiguous specification for establishing application boundaries.

A software tool can also produce non-monotonic results. Although it has well-defined rules for identifying the attributes, the rules will likely misinterpret some uncommon realization. Whether the count produced by a tool is right or wrong, it will certainly produce a repeatable result.

Despite these problems, if the program reader or the software tool follows carefully defined rules, either can generate highly reliable counts.

The FP metric uses the source code to deduce the characteristics of the algorithm solved by the program. The rules used for this step are dependent on the programming language and technology used. But it is the algorithm that is measured, not the program. Thus, as Drummond (1985) also points out, the FP metric has the advantage of being language independent, as well as being independent of programmer style or experience. It is even independent of any changes in technology. Once the size of the algorithm is calculated, any program in any language that performs the same algorithm, whether poorly or well written, will provide the same number of function points.

SQM RELATIONSHIPS: Complexity, Simplicity

TOOLS: Checkpoint, Software Productivity Research, Inc., Burlington, MA
SIZE PLANNER, Quantitative Software Management, Inc., McClean, VA

REFERENCES: Albrecht 1979, 1985; Albrecht and Gaffney 1983; Behrens 1983; Drummond 1985; Jones 1988; Low and Jeffery 1990

## H - Height of a Tree

**This metric quantifies the depth of nesting in a set of modules of source code. It is based on the number of levels in a hierarchy tree of the control flow between modules.**

RULE: The Height of a hierarchy tree is the maximum height attained by any node in the hierarchy tree, except that the Height of a single node tree is defined to be one.

The Height of a node is the number of levels of nesting below the root node. Thus, the root node has a height of zero and any child node of the root node has a height of one, being nested one level below the root node. Because the tree is drawn upside-down, height increases with lower levels of nesting.

A module (or node) is defined to be a set of code that embodies an aggregate of thought representing a single function. A module must be a well-defined entity in order to ensure that the count is repeatable.

DOMAIN: This measure only applies to source code that implements an algorithm; the code must perform some function which may or may not be composed of subfunctions.

RANGE: This metric produces an integer greater than or equal to one, the Height of the tree for source code that has no nested modules.

NECESSARY CONDITIONS: The combinations of characters that constitute a node, the source code constructs that constitute nesting, and the boundaries of a module must be defined for the context.

QUALIFICATIONS: None.

CRITICAL ANALYSIS: This measure is not necessarily strictly monotonic or repeatable. A program reader may erroneously nest a node when drawing the hierarchy tree. In this case, an increase in the measure does not represent an increase in the depth of nesting. Furthermore, a different reader will likely judge nesting differently.

A software tool can also produce non-monotonic results. Although it has well-defined rules for identifying nodes and nesting, the rules will likely misinterpret some uncommon realization. Whether the count produced by a tool is right or wrong, it will certainly produce a repeatable result.

Despite these problems, if the program reader or the software tool follows carefully defined rules, either can generate highly reliable counts.

SQM RELATIONSHIPS: Complexity, Performance, Simplicity, Understandability

TOOLS: COMPLEXIMETER, Softmetrix, Inc., Chicago, IL

REFERENCES: Ejiogu 1984[1], 1984[2], 1987, 1988, 1990

## Twin Number

**This metric quantifies the breadth of explosion of a module (or node) of source code. It is based on the number of nodes nested directly below it (child nodes) in a hierarchy tree of the control flow between nodes.**

RULE: The Twin Number of a node is the number of child nodes of which it is the parent, except that the Twin Number of the node in a single node tree is defined to be one.

Every node that is immediately related to a particular node at the next lowest level (one level closer to the root) is considered a child of that node.

A module (or node) is defined to be a set of code that embodies an aggregate of thought representing a single function. A module must be a well-defined entity in order to ensure that the count is repeatable.

DOMAIN: This measure only applies to source code that implements an algorithm; the code must perform some function which may or may not be composed of subfunctions.

RANGE: This metric produces an integer greater than or equal to zero, the Twin Number of a module with no child nodes (monad).

NECESSARY CONDITIONS: The combinations of characters that constitute a node, the source code constructs that constitute nesting, and the boundaries of a module must be defined for the context.

QUALIFICATIONS: None.

CRITICAL ANALYSIS: This measure is not necessarily strictly monotonic or repeatable. A program reader may erroneously nest a node when drawing the hierarchy tree. In this case, an increase in the measure does not represent an increase in the number of child nodes. Furthermore, a different reader will likely judge nesting differently.

A software tool can also produce non-monotonic results. Although it has well-defined rules for identifying nodes and nesting, the rules will likely misinterpret some uncommon realization. Whether the count produced by a tool is right or wrong, it will certainly produce a repeatable result.

Despite these problems, if the program reader or the software tool follows carefully defined rules, either can generate highly reliable counts.

SQM RELATIONSHIPS: Complexity, Simplicity

TOOLS: COMPLEXIMETER, Softmetrix, Inc., Chicago, IL

REFERENCES: Ejiogu 1984[1], 1984[2], 1987, 1988, 1990

## M - Monadicity

This metric quantifies the number of irreducible modules (or nodes) in a set of modules of source code. It is based on the number of childless nodes in a hierarchy tree of the control flow between modules.

RULE: The Monadicity of a hierarchy tree is the number of nodes that have no children (a monad), except that a tree with a Height of one is constrained to have a Monadicity of one.

Every node that is immediately related to a particular node at the next lowest level (one level closer to the root) is considered a child of that node.

A module (or node) is defined to be a set of code that embodies an aggregate of thought representing a single function. A module must be a well-defined entity in order to ensure that the count is repeatable.

This is a measure of the number of leaves on the tree, where the node at the end of each branch is a leaf. It is a measure of the "bushiness" of the tree.

DOMAIN: This measure only applies to source code that implements an algorithm; the code must perform some function which may or may not be composed of subfunctions.

RANGE: This metric produces an integer greater than or equal to one, the Monadicity of tree with a Height of one.

NECESSARY CONDITIONS: The combinations of characters that constitute a node, the source code constructs that constitute nesting and the boundaries of a module must be defined for the context.

QUALIFICATIONS: None.

CRITICAL ANALYSIS: This measure is discontinuous between hierarchy trees of Height one and two. A tree with five child nodes at level one and one child node at level two has a monadicity of five. But if the single monad at level two is removed, the monadicity becomes one.

Even for hierarchy trees with a Height greater than one, this measure is not necessarily strictly monotonic or repeatable. A program reader may erroneously divide a monad into two monads when drawing the hierarchy tree. In this case, an increase in the measure does not represent an increase in the number of monads. Furthermore, a different reader will likely judge nodes differently.

A software tool can also produce non-monotonic results. Although it has well-defined rules for identifying nodes and nesting, the rules will likely misinterpret some uncommon realization. Whether the count

produced by a tool is right or wrong, it will certainly produce a repeatable result. Despite these problems, if the program reader or the software tool follows carefully defined rules, either can generate highly reliable counts.

SQM RELATIONSHIPS: Complexity, Simplicity

TOOLS: COMPLEXIMETER, Softmetrix, Inc., Chicago, IL

REFERENCES: Ejiogu 1984[1], 1984[2], 1987, 1988, 1990

## S - Software Size

This metric calculates the size of a set of modules (or nodes) of source code. It is based on the number of nodes in the hierarchy tree of the control flow between modules.

RULE: Count the number of nodes above the root node.

A module (or node) is defined to be a set of code that embodies an aggregate of thought representing a single function. A module must be a well-defined entity in order to ensure that the count is repeatable.

DOMAIN: This measure only applies to source code that implements an algorithm; the code must perform some function which may or may not be composed of subfunctions.

RANGE: This metric produces an integer greater than or equal to zero, the size of a single-node tree.

NECESSARY CONDITIONS: The combinations of characters that constitute a node, the source code constructs that constitute nesting, and the boundaries of a module must be defined for the context.

QUALIFICATIONS: None.

CRITICAL ANALYSIS: This measure is not monotonic. When two different modules call the same subroutine, it appears as two distinct nodes on the hierarchy tree. This increases the size even though there is no increase in function.

Even if a program contains no such problem, this measure is not necessarily strictly monotonic or repeatable. A program reader may erroneously divide a monad when drawing the hierarchy tree. In this case, an increase in the measure does not represent an increase in the number of monads. Furthermore, a different reader will likely judge nodes differently.

A software tool can also produce non-monotonic results. Although it has well-defined rules for identifying nodes and nesting, the rules will likely misinterpret some uncommon realization. Whether the count produced by a tool is right or wrong, it will certainly produce a repeatable result.

Despite these problems, if the program reader or the software tool follows carefully defined rules, either can generate highly reliable counts.

Very little experimental validation has been performed for this metric and none of it has been published. No independent experiments have been performed.

SQM RELATIONSHIPS: Complexity, Performance, Simplicity, Understandability

TOOLS: COMPLEXIMETER, Softmetrix, Inc., Chicago, IL

REFERENCES: Ejiogu 1984[1], 1984[2], 1987, 1988, 1990

## $S_c$ - Structural Complexity

This metric calculates the structural complexity of a set of modules of source code. It is based on the number and relationship of nodes in a hierarchy tree of the control flow between nodes,

RULE: $S_c = H \times R_t \times M$

where H is the Height of the tree, $R_t$ is the Twin Number of the root node, and M is the Monadicity of the tree. Each of these arguments is defined on a previous data sheet.

A module (or node) is defined to be a set of code that embodies an aggregate of thought representing a single function. A module must be a well-defined entity in order to ensure that the count is repeatable.

The Structural Complexity reflects the nature of "the relation of conglomerates of thoughts expressing the functional composition of the system" (Ejiogu 1984). It combines into one measure both the number of related modules of thought and the closeness of their relationship. This measure was designed to be applied to issues relating to programming productivity, since it quantifies the amount of complexity in a program.

DOMAIN: This measure only applies to source code that implements an algorithm; the code must perform some function which may or may not be composed of subfunctions.

RANGE: This metric produces an integer greater than or equal to one, the Structural Complexity of a single-node tree.

NECESSARY CONDITIONS: The combinations of characters that constitute a node, the source code constructs that constitute nesting, and the boundaries of a module must be defined for the context.

QUALIFICATIONS: While this metric is based on structured programming concepts, unstructured programs do not render it useless. Any effect that lack of structure has on the value of the measure is to be taken as a reflection of the increased complexity that results when a program deviates from the ideal. Excessive values alert the developer to the presence of unstructured code so that it can be determined whether its presence is an asset or a detriment. For instance, when two nodes call the same subroutine the count is increased, even though there is no increase in function. But since calling a subroutine twice is not a detrimental effect, this apparent increase in complexity is allowed to remain.

CRITICAL ANALYSIS: This measure is not necessarily strictly monotonic or repeatable. The problems are due to the counts that compose it. See

the respective software metric data sheets for details. Very little experimental validation has been performed for this metric and none of it has been published. No independent experiments have been performed.

SQM RELATIONSHIPS: Complexity, Performance, Simplicity, Understandability

TOOLS: COMPLEXIMETER, Softmetrix, Inc., Chicago, IL

REFERENCES: Ejiogu 1984[1], 1984[2], 1987, 1988, 1990

## $\eta_1$ - Number of Unique Operators

This metric quantifies the number of unique operators that occur in a module of source code.

RULE: Count the number of unique operators in the defined module.

An operator is any word of source code that represents an operation to be performed. (A word of source code is any sequence of characters set off as such by the delimiters defined for the particular language.) The operation is performed on the objects that are the arguments of the operator. In computer program source code, most operators are the keywords that define a program statement. Some keywords have multiple parts that must occur as a set. They constitute one compound operator. The classification of each word of code must be based on an analysis of what that word of code does, from a functional point of view.

Uniqueness is to be determined according to the uniqueness of function rather than syntactical form. A unique function is rarely represented by more than one form. When this does occur, the various forms are not to be counted as unique operators. For example, if an exponent is denoted by either an "E" or a "^", these two forms represent one function. Conversely, two unique functions may be represented by the same form. When this occurs, the form is to be counted as two distinct operators, as indicated by the context. For example, if an asterisk denotes either multiplication or a comment, this one form represents two functions.

A module is defined to be a set of code that is to have a measure assigned to it. It must be a well-defined entity in order to ensure that the count is repeatable.

DOMAIN: This measure only applies to that part of source code that implements an algorithm; the code must contain solely operators and operands. An algorithm with less than two operators and one operand is undefined.

RANGE: This metric produces an integer greater than or equal to two, the number of operators in the most succinct implementation of an algorithm.

NECESSARY CONDITIONS: Every possible combination of characters that constitutes an operator, the criteria for establishing uniqueness, and the boundaries of a module must be defined for the context.

QUALIFICATIONS: The number of unique operators in a module is not necessarily equal to the sum of the values for each of its sub-modules. Thus, $\eta_1$ is not an additive quantity.

CRITICAL ANALYSIS: This measure is not necessarily strictly monotonic or repeatable. A program reader may increment the count for a word that is not an operator. In this case, an increase in the measure does not represent an increase in the number of operators. Furthermore, a different reader will likely define a different set of operators.

A software tool can also produce non-monotonic results. Although it has well-defined rules for identifying operators, the rules will likely misinterpret some uncommon realization. Whether the count produced by a tool is right or wrong, it will certainly produce a repeatable result.

Despite these problems, if the program reader or the software tool follows carefully defined rules, either can generate highly reliable counts.

SQM RELATIONSHIPS: Clarity, Complexity, Modifiability, Performance, Reliability, Simplicity, Understandability

TOOLS: LOGISCOPE, Verilog USA Inc., Alexandria, VA
PC-Metric, SET Laboratories, Inc., Mulino, OR

REFERENCES: Albrecht and Gaffney 1983; Arthur 1983; Basili, Selby, and Phillips 1983; Bulut 1974; Carver 1986; Coulter 1983; Curtis 1980; Elshoff 1976; Fitzsimmons 1978; Funami 1976; Gordon 1976, 1979; Halstead 1972, 1977, 1979; Halstead and Zislis 1973; Henry, Kafura, and Harris 1981; Jones 1978; Kafura and Reddy 1987; Lassez 1981; Li and Cheung 1987; Prather 1984; Shen, Conte, and Dunsmore 1983; Shneiderman 1980; Siyan 1989; Weyuker 1988; Zislis 1973; Zweben 1977

## $\eta_2$ - Number of Unique Operands

This metric quantifies the number of unique operands that occur in a module of source code.

RULE: Count the number of unique operands in the defined module.

An operand is any word of source code that represents an argument upon which an operation is to be performed. (A word of source code is any sequence of characters set off as such by the delimiters defined for the particular language.) In computer program source code, most operands are the variables and constants of the program. The classification of each word of code must be based on an analysis of what that word of code does from a functional point of view.

Uniqueness is to be determined according to the uniqueness of function rather than syntactical form. A unique argument is rarely represented by more than one form. When this does occur, the various forms are not to be counted as unique operands. For example, if a variable name can be denoted by either VELOCITY or VEL, these two forms represent one argument. Conversely, two unique arguments may be represented by the same form. When this occurs, the form is to be counted as two distinct operands, as indicated by the context. For example, if the constant, 10, denotes 10 array elements in one part of a program, but it also denotes 10 volts in another part, this one operand represents two arguments.

A module is defined to be a set of code that is to have a measure assigned to it. It must be a well-defined entity in order to ensure that the count is repeatable.

DOMAIN: This measure only applies to that part of source code that implements an algorithm; the code must contain solely operators and operands. An algorithm with less than two operators and one operand is undefined.

RANGE: This metric produces an integer greater than or equal to one, the number of operands in the most succinct implementation of an algorithm.

NECESSARY CONDITIONS: Every possible combination of characters that constitutes an operand, the criteria for establishing uniqueness, and the boundaries of a module must be defined for the context.

QUALIFICATIONS: The number of unique operands in a module is not necessarily equal to the sum of the values for each of its sub-modules. Thus, $\eta_2$ is not an additive quantity.

CRITICAL ANALYSIS: This measure is not necessarily strictly monotonic or repeatable. A program reader may increment the count for a word that

is not an operand. In this case, an increase in the measure does not represent an increase in the number of operands. Furthermore, a different reader will likely define a different set of operands. A software tool can also produce non-monotonic results. Although it has well-defined rules for identifying operands, the rules will likely misinterpret some uncommon realization. Whether the count produced by a tool is right or wrong, it will certainly produce a repeatable result.

Despite these problems, if the program reader or the software tool follows carefully defined rules, either can generate highly reliable counts.

SQM RELATIONSHIPS: Clarity, Complexity, Modifiability, Performance, Reliability, Simplicity, Understandability

TOOLS: LOGISCOPE, Verilog USA Inc., Alexandria, VA
PC-Metric, SET Laboratories, Inc., Mulino, OR

REFERENCES: Albrecht and Gaffney 1983; Arthur 1983; Basili, Selby, and Phillips 1983; Bulut 1974; Carver 1986; Coulter 1983; Curtis 1980; Elshoff 1976; Fitzsimmons 1978; Funami 1976; Gordon 1976, 1979; Halstead 1972, 1977, 1979; Halstead and Zislis 1973; Henry, Kafura, and Harris 1981; Jones 1978; Kafura and Reddy 1987; Lassez 1981; Li and Cheung 1987; Prather 1984; Shen, Conte, and Dunsmore 1983; Shneiderman 1980; Siyan 1989; Weyuker 1988; Zislis 1973; Zweben 1977

## $N_1$ - Total Number of Operator Occurrences

This metric quantifies the number of occurrences of operators in a module of source code.

RULE: Count the total number of occurrences of operators in the defined module.

An operator is any word of source code that represents an operation to be performed. (A word of source code is any sequence of characters set off as such by the delimiters defined for the particular language.) The operation is performed on the objects that are the arguments of the operator. In computer program source code, most operators are the keywords that define a program statement. Some keywords have multiple parts that must occur as a set. They constitute one compound operator. The classification of each word of code must be based on an analysis of what that word of code does from a functional point of view.

A module is defined to be a set of code that is to have a measure assigned to it. It must be a well-defined entity in order to ensure that the count is repeatable.

DOMAIN: This measure only applies to that part of source code that implements an algorithm; the code must contain solely operators and operands. An algorithm with less than two operators and one operand is undefined.

RANGE: This metric produces an integer greater than or equal to two, the number of operator occurrences in the most succinct implementation of an algorithm.

NECESSARY CONDITIONS: Every possible combination of characters that constitutes an operator must be defined for the context. The boundaries of a module must be defined for the context.

QUALIFICATIONS: None.

CRITICAL ANALYSIS: This measure is not necessarily strictly monotonic or repeatable. A program reader may increment the count for a word that is not an operator. In this case, an increase in the measure does not represent an increase in the number of operators. Furthermore, a different reader will likely define a different set of operators.

A software tool can also produce non-monotonic results. Although it has well-defined rules for identifying operators, the rules will likely misinterpret some uncommon realization. Whether the count produced by a tool is right or wrong, it will certainly produce a repeatable result.

Despite these problems, if the program reader or the software tool follows carefully defined rules, either can generate highly reliable counts.

SQM RELATIONSHIPS: Clarity, Complexity, Modifiability, Performance, Reliability, Simplicity, Understandability

TOOLS: LOGISCOPE, Verilog USA Inc., Alexandria, VA
PC-Metric, SET Laboratories, Inc., Mulino, OR

REFERENCES: Albrecht and Gaffney 1983; Arthur 1983; Basili, Selby, and Phillips 1983; Bulut 1974; Carver 1986; Coulter 1983; Curtis 1980; Elshoff 1976; Fitzsimmons 1978; Funami 1976; Gordon 1976, 1979; Halstead 1972, 1977, 1979; Halstead and Zislis 1973; Henry, Kafura, and Harris 1981; Jones 1978; Kafura and Reddy 1987; Lassez 1981; Li and Cheung 1987; Prather 1984; Shen, Conte, and Dunsmore 1983; Shneiderman 1980; Siyan 1989; Weyuker 1988; Zislis 1973; Zweben 1977

## $N_2$ - Total Number of Operand Occurrences

This metric quantifies the number of occurrences of operands in a module of source code.

RULE: Count the total number of occurrences of operands in the defined module.

An operand is any word of source code that represents an argument upon which an operation is to be performed. (A word of source code is any sequence of characters set off as such by the delimiters defined for the particular language.) In computer program source code, operands are the variables and constants of the program. The classification of each word of code must be based on an analysis of what that word of code does, from a functional point of view.

A module is defined to be a set of code that is to have a measure assigned to it. It must be a well-defined entity in order to ensure that the count is repeatable.

DOMAIN: This measure only applies to that part of source code that implements an algorithm; the code must contain solely operators and operands. An algorithm with less than two operators and one operand is undefined.

RANGE: This metric produces an integer greater than or equal to one, the number of operand occurrences in the most succinct implementation of an algorithm.

NECESSARY CONDITIONS: Every possible combination of characters that constitutes an operand must be defined for the context. The boundaries of a module must be defined for the context.

QUALIFICATIONS: None.

CRITICAL ANALYSIS: This measure is not necessarily strictly monotonic or repeatable. A program reader may increment the count for a word that is not an operand. In this case, an increase in the measure does not represent an increase in the number of operands. Furthermore, a different reader will likely define a different set of operands.

A software tool can also produce non-monotonic results. Although it has well-defined rules for identifying operands, the rules will likely misinterpret some uncommon realization. Whether the count produced by a tool is right or wrong, it will certainly produce a repeatable result.

Despite these problems, if the program reader or the software tool follows carefully defined rules, either can generate highly reliable counts.

SQM RELATIONSHIPS: Clarity, Complexity, Modifiability, Performance, Reliability, Simplicity, Understandability

TOOLS: LOGISCOPE, Verilog USA Inc., Alexandria, VA
PC-Metric, SET Laboratories, Inc., Mulino, OR

REFERENCES: Albrecht and Gaffney 1983; Arthur 1983; Basili, Selby, and Phillips 1983; Bulut 1974; Carver 1986; Coulter 1983; Curtis 1980; Elshoff 1976; Fitzsimmons 1978; Funami 1976; Gordon 1976, 1979; Halstead 1972, 1977, 1979; Halstead and Zislis 1973; Henry, Kafura, and Harris 1981; Jones 1978; Kafura and Reddy 1987; Lassez 1981; Li and Cheung 1987; Prather 1984; Shen, Conte, and Dunsmore 1983; Shneiderman 1980; Siyan 1989; Weyuker 1988; Zislis 1973; Zweben 1977

## $\eta$ - Vocabulary

This metric calculates the total number of unique operators and operands that occur in a module of source code.

RULE: $\eta = \eta_1 + \eta_2$ words

where $\eta_1$ is the number of unique operators and $\eta_2$ is the number of unique operands. Each of these arguments is defined on a previous data sheet.

A module is defined to be a set of code that is to have a measure assigned to it. It must be a well-defined entity in order to ensure that the count is repeatable.

The Vocabulary represents the number of unique words in a module. Every word of executable code should be listed once in a vocabulary listing. However, when an operator and an operand are identical in form, they still constitute unique words since they serve unique functions. They are distinguished by their context.

DOMAIN: This measure only applies to that part of source code that implements an algorithm; the code must contain solely operators and operands. An algorithm with less than two operators and one operand is undefined.

RANGE: This metric produces an integer greater than or equal to three, the Vocabulary of the most succinct implementation of an algorithm.

NECESSARY CONDITIONS: Every possible combination of characters that constitutes an operator or operand, the criteria for establishing uniqueness, and the boundaries of a module must be defined for the context.

QUALIFICATIONS: The Vocabulary of a module is not necessarily equal to the sum of the Vocabularies of each of its sub-modules. Thus, Vocabulary is not an additive quantity.

CRITICAL ANALYSIS: This measure is not necessarily strictly monotonic or repeatable. The problems are due to the counts that compose it. See the respective software metric data sheets for details.

SQM RELATIONSHIPS: Clarity, Complexity, Modifiability, Performance, Reliability, Simplicity, Understandability

TOOLS: LOGISCOPE, Verilog USA Inc., Alexandria, VA
PC-Metric, SET Laboratories, Inc., Mulino, OR

REFERENCES: Albrecht and Gaffney 1983; Arthur 1983; Basili, Selby, and Phillips 1983; Bulut 1974; Carver 1986; Coulter 1983; Curtis 1980; Elshoff

1976; Fitzsimmons 1978; Funami 1976; Gordon 1976, 1979; Halstead 1972, 1977, 1979; Halstead and Zislis 1973; Henry, Kafura, and Harris 1981; Jones 1978; Kafura and Reddy 1987; Lassez 1981; Li and Cheung 1987; Prather 1984; Shen, Conte, and Dunsmore 1983; Shneiderman 1980; Siyan 1989; Weyuker 1988; Zislis 1973; Zweben 1977

## N - Implementation Length

This metric calculates the total number of occurrences of operators and operands in a module of source code.

RULE: $N = N_1 + N_2$ words

where $N_1$ is the total number of operator occurrences and $N_2$ is the total number of operand occurrences. Each of these arguments is defined on a previous data sheet.

A module is defined to be a set of code that is to have a measure assigned to it. It must be a well-defined entity in order to ensure that the count is repeatable.

The Implementation Length represents the count of every word in a module. No word of executable code should be left uncounted.

DOMAIN: This measure only applies to that part of source code that implements an algorithm; the code must contain solely operators and operands. An algorithm with less than two operators and one operand is undefined.

RANGE: This metric produces an integer greater than or equal to three, the Implementation Length of the most succinct implementation of an algorithm.

NECESSARY CONDITIONS: Every possible combination of characters that constitutes an operator or operand must be defined for the context. The boundaries of a module must be defined for the context.

QUALIFICATIONS: None.

CRITICAL ANALYSIS: This measure is not necessarily strictly monotonic or repeatable. The problems are due to the counts that compose it. See the respective software metric data sheets for details.

SQM RELATIONSHIPS: Clarity, Complexity, Modifiability, Performance, Reliability, Simplicity, Understandability

TOOLS: LOGISCOPE, Verilog USA Inc., Alexandria, VA
PC-Metric, SET Laboratories, Inc., Mulino, OR

REFERENCES: Albrecht and Gaffney 1983; Arthur 1983; Basili, Selby, and Phillips 1983; Bulut 1974; Carver 1986; Coulter 1983; Curtis 1980; Elshoff 1976; Fitzsimmons 1978; Funami 1976; Gordon 1976, 1979; Halstead 1972, 1977, 1979; Halstead and Zislis 1973; Henry, Kafura, and Harris 1981; Jones 1978; Kafura and Reddy 1987; Lassez 1981; Li and Cheung 1987;

Prather 1984; Shen, Conte, and Dunsmore 1983; Shneiderman 1980; Siyan 1989; Weyuker 1988; Zislis 1973; Zweben 1977

## $\hat{N}$ - Estimated Length

This metric estimates the total number of occurrences of operators and operands in a module of source code.

RULE: $\hat{N} = \eta_1 \log_2 \eta_1 + \eta_2 \log_2 \eta_2$ words

where $\eta_1$ is the number of unique operators and $\eta_2$ is the number of unique operands. Each of these arguments is defined on a previous data sheet.

A module is defined to be a set of code that is to have a measure assigned to it. It must be a well-defined entity in order to ensure that the count is repeatable.

DOMAIN: This measure only applies to that part of source code that implements an algorithm; the code must contain solely operators and operands. An algorithm with less than two operators and one operand is undefined.

RANGE: This metric produces a real number greater than or equal to two, the Estimated Length of the most succinct implementation of an algorithm.

NECESSARY CONDITIONS: Every possible combination of characters that constitutes an operator or operand, the criteria for establishing the uniqueness of operators and operands, and the boundaries of a module must be defined for the context.

QUALIFICATIONS: While the measure is given in terms of a real number, the number of words should be rounded up to the next highest integer.

This estimate assumes that every combination of operators and operands of length $\eta$ occurs only once in the module. This means that repeated segments of code, as long as $\eta$ words, are put into subroutines rather than in redundant code. If such redundancy is present in the code, the Implementation Length will be underestimated by the Estimated Length Equation.

The Estimated Length Equation also assumes that operators and operands alternate without variation. Any code that does not follow this convention will likely produce an underestimation of the Implementation Length.

Because the Estimated Length Equation is nonlinear, adding the Estimated Lengths of sub-modules would not be expected to give an accurate Estimated Length for the module that they constitute. Nevertheless, an experiment by Ingojo indicated that the relationship holds fairly well (Halstead 1977).

CRITICAL ANALYSIS: This measure is not necessarily strictly monotonic or repeatable. Some of the problems are due to problems with the counts that compose it. See the respective software metric data sheets for the details of the problems that they contribute to this metric.

Most of the problems with the Estimated Length, however, are due to the simplifying assumptions upon which the estimate is based. Whether or not these assumptions are reasonable, experiments show that $\hat{N}$ yields a close estimate of the Implementation Length, N (Halstead 1977).

SQM RELATIONSHIPS: Clarity, Complexity, Modifiability, Performance, Reliability, Simplicity, Understandability

TOOLS: LOGISCOPE, Verilog USA Inc., Alexandria, VA
PC-Metric, SET Laboratories, Inc., Mulino, OR

REFERENCES: Albrecht and Gaffney 1983; Arthur 1983; Basili, Selby, and Phillips 1983; Bulut 1974; Carver 1986; Coulter 1983; Curtis 1980; Elshoff 1976; Fitzsimmons 1978; Funami 1976; Gordon 1976, 1979; Halstead 1972, 1977, 1979; Halstead and Zislis 1973; Henry, Kafura, and Harris 1981; Jones 1978; Kafura and Reddy 1987; Lassez 1981; Li and Cheung 1987; Prather 1984; Shen, Conte, and Dunsmore 1983; Shneiderman 1980; Siyan 1989; Weyuker 1988; Zislis 1973; Zweben 1977

## V - Volume

This metric calculates the number of binary digits required to uniquely represent all of the operators and operands that occur in a module of length, N, and Vocabulary, $\eta$. It is not primarily a measure of the size or complexity of the function programmed, but a measure of the size of a particular implementation of a function. This metric should primarily be used to compare the size of the same program written in various languages or by various programmers.

RULE: $V = N\log_2\eta$ bits

where N is the Implementation Length and $\eta$ is the Vocabulary. Each of these arguments is defined on a previous data sheet.

A module is defined to be a set of code that is to have a measure assigned to it. It must be a well-defined entity in order to ensure that the count is repeatable.

DOMAIN: This measure only applies to that part of source code that implements an algorithm; the code must contain solely operators and operands. An algorithm with less than two operators and one operand is undefined.

RANGE: This metric produces a real number greater than or equal to about 4.755, the Volume of the most succinct implementation of an algorithm.

NECESSARY CONDITIONS: Every possible combination of characters that constitutes an operator or operand, the criteria for establishing the uniqueness of operators and operands, and the boundaries of a module must be defined for the context.

QUALIFICATIONS: While the measure is given in terms of a real number, the number of bits should be rounded up to the next highest integer.

The Volume of a module is not necessarily equal to the sum of the values of each of its sub-modules. Thus, Volume is not an additive quantity.

CRITICAL ANALYSIS: This measure is not necessarily strictly monotonic or repeatable. The problems are due to the counts that compose it. See the respective software metric data sheets for the details of the problems that they contribute to this metric.

This measure is related to the size of the function being implemented, but it is not a monotonic relationship. A larger algorithm could be implemented so tersely that its volume is less than a smaller algorithm. Conversely, a smaller algorithm could be implemented so verbosely that its volume is greater than a larger algorithm.

SQM RELATIONSHIPS: Clarity, Complexity, Modifiability, Performance, Reliability, Simplicity, Understandability

TOOLS: LOGISCOPE, Verilog USA Inc., Alexandria, VA
PC-Metric, SET Laboratories, Inc., Mulino, OR

REFERENCES: Albrecht and Gaffney 1983; Arthur 1983; Basili, Selby, and Phillips 1983; Bulut 1974; Carver 1986; Coulter 1983; Curtis 1980; Elshoff 1976; Fitzsimmons 1978; Funami 1976; Gordon 1976, 1979; Halstead 1972, 1977, 1979; Halstead and Zislis 1973; Henry, Kafura, and Harris 1981; Jones 1978; Kafura and Reddy 1987; Lassez 1981; Li and Cheung 1987; Prather 1984; Shen, Conte, and Dunsmore 1983; Shneiderman 1980; Siyan 1989; Weyuker 1988; Zislis 1973; Zweben 1977

## $V^*$ - Potential Volume

This metric calculates the number of binary digits required to implement the function of a module in its most efficient form, namely two operators (i.e., one that says "do the function" and another that groups the operands with the operator), and all the operands that it requires.

RULE: $V^* = (2 + \eta_2^*)\log_2(2 + \eta_2^*)$ bits

where $\eta_2^*$ is the sum of the number of unique inputs and outputs for the module.

A module is defined to be a set of code that is to have a measure assigned to it. It must be a well-defined entity in order to ensure that the count is repeatable.

DOMAIN: This measure only applies to that part of source code that implements an algorithm; the code must contain solely operators and operands. An algorithm with less than two operators and one operand is undefined. The operand ensures that there is at least one input or output.

RANGE: This metric produces a real number greater than or equal to about 4.755, the Potential Volume of the most succinct implementation of an algorithm.

NECESSARY CONDITIONS: Every possible combination of characters that constitutes an operator or operand, the criteria for establishing the uniqueness of operators and operands, and the boundaries of a module must be defined for the context.

QUALIFICATIONS: While the measure is given in terms of a real number, the number of bits should be rounded up to the next highest integer.

The Potential Volume of a module is not necessarily equal to the sum of the values of each of its sub-modules. Thus, Potential Volume is not an additive quantity.

CRITICAL ANALYSIS: This measure is not necessarily strictly monotonic or repeatable. A program reader may increment the count for some part of the code that is not an input or output. In this case, an increase in the measure does not represent an increase in the size of the function. Furthermore, a different reader will likely define a different set of inputs and outputs.

A software tool can also produce non-monotonic results. Although it has well-defined rules for identifying inputs and outputs, the rules will likely misinterpret some uncommon realization. Whether the count produced by a tool is right or wrong, it will certainly produce a

repeatable result. Despite these problems, if the program reader or the software tool follows carefully defined rules, either can generate highly reliable counts.

SQM RELATIONSHIPS: Conciseness, Efficiency

TOOLS: No listing.

REFERENCES: Albrecht and Gaffney 1983; Arthur 1983; Basili, Selby, and Phillips 1983; Bulut 1974; Carver 1986; Coulter 1983; Curtis 1980; Elshoff 1976; Fitzsimmons 1978; Funami 1976; Gordon 1976, 1979; Halstead 1972, 1977, 1979; Halstead and Zislis 1973; Henry, Kafura, and Harris 1981; Jones 1978; Kafura and Reddy 1987; Lassez 1981; Li and Cheung 1987; Prather 1984; Shen, Conte, and Dunsmore 1983; Shneiderman 1980; Siyan 1989; Weyuker 1988; Zislis 1973; Zweben 1977

## L - Program Level

This metric calculates how efficiently a module of source code is implemented. It is the ratio of the minimum number of bits it takes to represent an algorithm to the number of bits in a particular implementation. As the Vocabulary and/or Implementation Length decrease, the Program Level increases, i.e., the module is written at a higher level because each word carries more weight. Thus, Program Level indicates the amount of function per word rather than the number of words per function. In this sense, it measures what a program hides rather than what it shows.

RULE: $$L = \frac{V^*}{V} \frac{\text{bits of function}}{\text{bits of implementation}}$$

where $V^*$ is the Potential Volume and V is the Volume. Each of these arguments is defined on a previous data sheet.

A module is defined to be a set of code that is to have a measure assigned to it. It must be a well-defined entity in order to ensure that the count is repeatable.

DOMAIN: This measure only applies to that part of source code that implements an algorithm; the code must contain solely operators and operands. An algorithm with less than two operators and one operand is undefined. The operand ensures that there is at least one input or output.

RANGE: This metric produces a positive real number less than or equal to one, the Program Level of the most succinct implementation of an algorithm.

NECESSARY CONDITIONS: Every possible combination of characters that constitutes an operator or operand, the criteria for establishing the uniqueness of operators and operands, and the boundaries of a module must be defined for the context.

QUALIFICATIONS: The Program Level of a module is not necessarily equal to the sum of the values for each of its sub-modules. Thus, Program Level is not an additive quantity.

CRITICAL ANALYSIS: This measure is not necessarily strictly monotonic or repeatable. The problems are due to the counts that compose it. See the respective software metric data sheets for the details of the problems that they contribute to this metric.

SQM RELATIONSHIPS: Clarity, Complexity, Modifiability, Performance, Reliability, Simplicity, Understandability

TOOLS: PC-Metric, SET Laboratories, Inc., Mulino, OR

REFERENCES: Albrecht and Gaffney 1983; Arthur 1983; Basili, Selby, and Phillips 1983; Bulut 1974; Carver 1986; Coulter 1983; Curtis 1980; Elshoff 1976; Fitzsimmons 1978; Funami 1976; Gordon 1976, 1979; Halstead 1972, 1977, 1979; Halstead and Zislis 1973; Henry, Kafura, and Harris 1981; Jones 1978; Kafura and Reddy 1987; Lassez 1981; Li and Cheung 1987; Prather 1984; Shen, Conte, and Dunsmore 1983; Shneiderman 1980; Siyan 1989; Weyuker 1988; Zislis 1973; Zweben 1977

## $\hat{L}$ - Estimated Program Level

This metric calculates how efficiently a module of source code is implemented. It is an estimate of the Program Level metric that can be used when the number of unique inputs and outputs, $\eta_2^*$, is not known. The Estimated Program Level increases as the module is written at a higher level. This occurs as the number of operators decreases to its minimum value of two and as the number of operand occurrences decreases to only one occurrence of each operand.

RULE: $$\hat{L} = \frac{2}{\eta_1} \frac{\eta_2}{N_2}$$ (unitless)

where $\eta_2$ is the number of unique operands, $\eta_1$ is the number of unique operators, and $N_2$ is the total number of operand occurrences. Each of these arguments is defined on a previous data sheet.

A module is defined to be a set of code that is to have a measure assigned to it. It must be a well-defined entity in order to ensure that the count is repeatable.

DOMAIN: This measure only applies to that part of source code that implements an algorithm; the code must contain solely operators and operands. An algorithm with less than two operators and one operand is undefined.

RANGE: This metric produces a positive real number less than or equal to one, the Estimated Program Level of the most succinct implementation of an algorithm.

NECESSARY CONDITIONS: Every possible combination of characters that constitutes an operator or operand, the criteria for establishing the uniqueness of operators and operands, and the boundaries of a module must be defined for the context.

QUALIFICATIONS: The Estimated Program Level of a module is not necessarily equal to the sum of the estimated values for each of its sub-modules. Thus, the Estimated Program Level is not an additive quantity.

CRITICAL ANALYSIS: This measure is not necessarily strictly monotonic or repeatable. Some of the problems are due to the counts that compose it. See the respective software metric data sheets for the details of the problems that they contribute to this metric.

Most of the problems are due to the simplifying assumptions upon which the estimate is based. Whether or not these assumptions are reasonable, experiments show that $\hat{L}$ yields a close estimate of the Program Level, L (Bulut 1974, Elshoff 1976, Halstead 1977, and Ottenstein 1981). They can be used interchangeably.

SQM RELATIONSHIPS: Clarity, Complexity, Modifiability, Performance, Reliability, Simplicity, Understandability

TOOLS: LOGISCOPE, Verilog USA Inc., Alexandria, VA
PC-Metric, SET Laboratories, Inc., Mulino, OR

REFERENCES: Albrecht and Gaffney 1983; Arthur 1983; Basili, Selby, and Phillips 1983; Bulut 1974; Carver 1986; Coulter 1983; Curtis 1980; Elshoff 1976; Fitzsimmons 1978; Funami 1976; Gordon 1976, 1979; Halstead 1972, 1977, 1979; Halstead and Zislis 1973; Henry, Kafura, and Harris 1981; Jones 1978; Kafura and Reddy 1987; Lassez 1981; Li and Cheung 1987; Prather 1984; Shen, Conte, and Dunsmore 1983; Shneiderman 1980; Siyan 1989; Weyuker 1988; Zislis 1973; Zweben 1977

## I - Intelligence Content

This metric simply calculates the product of the Estimated Program Level (how efficiently the module is implemented) and the Volume (the size of the implementation) of a module of source code. It represents that constant quantity of information that is present in any implementation of a particular function, in any language, at any level. This quantity is basically an estimated Potential Volume of the program; I and $V^*$ can often be used interchangeably.

RULE: $$I = \frac{2\ \eta_2}{\eta_1\ N_2} \times (N_1 + N_2)\log_2(\eta_1 + \eta_2) \quad \text{bits}$$

where $\eta_2$ is the number of unique operands, $N_1$ is the total number of operator occurrences, $N_2$ is the total number of operand occurrences, and $\eta_1$ is the number of unique operators. Each of these arguments is defined on a previous data sheet.

A module is defined to be a set of code that is to have a measure assigned to it. It must be a well-defined entity in order to ensure that the count is repeatable.

DOMAIN: This measure only applies to that part of source code that implements an algorithm; the code must contain solely operators and operands. An algorithm with less than two operators and one operand is undefined.

RANGE: This metric produces a real number greater than or equal to about 4.755, the Intelligence Content of the most succinct implementation of an algorithm.

NECESSARY CONDITIONS: Every possible combination of characters that constitutes an operator or operand, the criteria for establishing the uniqueness of operators and operands, and the boundaries of a module must be defined for the context.

QUALIFICATIONS: While the measure is given in terms of a real number, the number of bits should be rounded up to the next highest integer.

The Intelligence Content of a module is not necessarily equal to the sum of the values for each of its sub-modules. Thus, Intelligence Content is not an additive quantity.

CRITICAL ANALYSIS: This measure is not necessarily strictly monotonic or repeatable. The problems are due to the counts that compose it. See the respective software metric data sheets for the details of the problems that they contribute to this metric.

The Intelligence Content measures the size of an algorithm based on its implementation. The Potential Volume measures the size of an algorithm based on its functional aspects: inputs and outputs. Therefore, when the Intelligence Content is used as an estimate for the Potential Volume, the accuracy of the estimate depends upon the quality of the programming. Redundant and extraneous uses of operators and operands alter the estimate. Halstead has characterized improper usages and has categorized them into six groups that he calls impurity classes (Halstead 1977).

Although it is generally undesirable that the constancy of the Intelligence Content be affected by the presence of program impurities, this effect results in an additional use of this metric. It provides a measure of the amount of impurity in an algorithm implementation. The closer the estimate is to the Potential Volume, the purer the algorithm implementation.

SQM RELATIONSHIPS: Conciseness, Efficiency

TOOLS: LOGISCOPE, Verilog USA Inc., Alexandria, VA
PC-Metric, SET Laboratories, Inc., Mulino, OR

REFERENCES: Albrecht and Gaffney 1983; Arthur 1983; Basili, Selby, and Phillips 1983; Bulut 1974; Carver 1986; Coulter 1983; Curtis 1980; Elshoff 1976; Fitzsimmons 1978; Funami 1976; Gordon 1976, 1979; Halstead 1972, 1977, 1979; Halstead and Zislis 1973; Henry, Kafura, and Harris 1981; Jones 1978; Kafura and Reddy 1987; Lassez 1981; Li and Cheung 1987; Prather 1984; Shen, Conte, and Dunsmore 1983; Shneiderman 1980; Siyan 1989; Weyuker 1988; Zislis 1973; Zweben 1977

## E - Programming Effort

This metric calculates the number of elementary mental discriminations done by a programmer to reduce a preconceived algorithm to a module of source code in a language in which the programmer is fluent. It is based on the assumption that, when writing a program, a programmer selects each word of the program by mentally searching a list of words from which to choose. Specifically, the programmer performs a mental binary search of the vocabulary of $\eta$ words in order to select the N words used in the implementation. Furthermore, each comparison (or mental discrimination) in the selection process requires an effort related to the difficulty of understanding the program. This program difficulty is supplied by the reciprocal of the Program Level.

RULE: $E = V/L$ discriminations

where V is the Volume and L is the Program Level. Each of these arguments is defined on a previous data sheet. A module is defined to be a set of code that is to have a measure assigned to it. It must be a well-defined entity in order to ensure that the count is repeatable.

This relationship indicates that the greater the Volume, or the lower the Program Level, the greater the effort required to write a program.

DOMAIN: This measure only applies to that part of source code that implements an algorithm; the code must contain solely operators and operands. An algorithm with less than two operators and one operand is undefined.

RANGE: This metric produces a real number greater than or equal to about 4.755, the Programming Effort for the most succinct implementation of an algorithm.

NECESSARY CONDITIONS: Every possible combination of characters that constitutes an operator or operand, the criteria for establishing the uniqueness of operators and operands, and the boundaries of a module must be defined for the context.

QUALIFICATIONS: While the measure is given in terms of a real number, the number of bits should be rounded up to the next highest integer.

The Program Effort of a module is not necessarily equal to the sum of the values for each of its sub-modules. Thus, Program Effort is not an additive quantity.

CRITICAL ANALYSIS: This measure is not necessarily strictly monotonic or repeatable. The problems are due to the counts that compose it. See the respective software metric data sheets for the details of the problems that they contribute to this metric. Often, the Estimated

Program Level is used in place of the Program Level. In this case, any lack of monotonicity in the Programming Effort metric would be produced primarily by the assumptions underlying the Estimated Program Level.

SQM RELATIONSHIPS: Clarity, Complexity, Maintainability, Reliability, Simplicity

TOOLS: LOGISCOPE, Verilog USA Inc., Alexandria, VA
PC-Metric, SET Laboratories, Inc., Mulino, OR

REFERENCES: Albrecht and Gaffney 1983; Arthur 1983; Basili, Selby, and Phillips 1983; Bulut 1974; Carver 1986; Coulter 1983; Curtis 1980; Elshoff 1976; Fitzsimmons 1978; Funami 1976; Gordon 1976, 1979; Halstead 1972, 1977, 1979; Halstead and Zislis 1973; Henry, Kafura, and Harris 1981; Jones 1978; Kafura and Reddy 1987; Lassez 1981; Li and Cheung 1987; Prather 1984; Shen, Conte, and Dunsmore 1983; Shneiderman 1980; Siyan 1989; Weyuker 1988; Zislis 1973; Zweben 1977

## $\hat{T}$ - Estimated Programming Time

This metric calculates the time it took a programmer to make the number of elementary mental discriminations performed when a module of source code was programmed. The number of elementary mental discriminations is given by the Programming Effort.

RULE: $$\hat{T} = \frac{E}{3600S} = \frac{\eta_1 N_2(\eta_1 \log_2 \eta_1 + \eta_2 \log_2 \eta_2)\log_2 \eta}{7{,}200\eta_2 S} \text{ hours}$$

where E is the Programming Effort and the Stroud Number, S, is the total number of elementary mental discriminations that a programmer makes per second. This calculation assumes that the programmer's attention is entirely undivided and that the range of values found from psychological experimentation, $5<S<20$, applies to programming activity.

Each of the remaining arguments of this rule is defined on a previous data sheet. A module is defined to be a set of code that is to have a measure assigned to it. It must be a well-defined entity in order to ensure that the count is repeatable.

DOMAIN: This measure only applies to that part of source code that implements an algorithm; the code must contain solely operators and operands. An algorithm with less than two operators and one operand is undefined.

RANGE: This metric produces a real number greater than or equal to about one quarter of a second, the Estimated Programming Time for the most succinct implementation of an algorithm (with $S=20$).

NECESSARY CONDITIONS: Every possible combination of characters that constitutes an operator or operand, the criteria for establishing the uniqueness of operators and operands, and the boundaries of a module must be defined for the context.

QUALIFICATIONS: The Estimated Programming Time of a module is not necessarily equal to the sum of the estimated values for each of its sub-modules. Thus, Estimated Programming Time is not an additive quantity.

CRITICAL ANALYSIS: This measure is not necessarily strictly monotonic or repeatable. Some of the problems are due to the counts that compose it. See the respective software metric data sheets for the details of the problems that they contribute to this metric.

Most of the problems are due to the variability associated with the Stroud Number. It is not necessarily monotonic or repeatable for all programming contexts. It is not necessarily monotonic because a

programmer's knowledge in one discipline may result in a shorter programming time for a program with a greater value of Program Effort than for a program with a lesser value of Program Effort. This could occur when the former program is in a discipline with which the programmer is familiar, but the latter program is not. It is not necessarily repeatable because on some days a programmer will concentrate better than on others.

SQM RELATIONSHIPS: Performance

TOOLS: LOGISCOPE, Verilog USA Inc., Alexandria, VA
PC-Metric, *SET Laboratories*, Inc., Mulino, OR

REFERENCES: Albrecht and Gaffney 1983; Arthur 1983; Basili, Selby, and Phillips 1983; Bulut 1974; Carver 1986; Coulter 1983; Curtis 1980; Elshoff 1976; Fitzsimmons 1978; Funami 1976; Gordon 1976, 1979; Halstead 1972, 1977, 1979; Halstead and Zislis 1973; Henry, Kafura, and Harris 1981; Jones 1978; Kafura and Reddy 1987; Lassez 1981; Li and Cheung 1987; Prather 1984; Shen, Conte, and Dunsmore 1983; Shneiderman 1980; Siyan 1989; Weyuker 1988; Zislis 1973; Zweben 1977

## $\lambda$ - Language Level

This metric calculates the efficiency with which algorithms can be implemented in a particular language. For any module of source code written in the particular language, the Language Level should be a constant, regardless of the Volume of the module or the Program Level at which it is written.

RULE: $\lambda = L^2V$

where L is the Program Level and V is the Volume. Each of these arguments is defined on a previous data sheet.

A module is defined to be a set of code that is to have a measure assigned to it. It must be a well-defined entity in order to ensure that the count is repeatable.

DOMAIN: This measure only applies to that part of source code that implements an algorithm; the code must contain solely operators and operands. An algorithm with less than two operators and one operand is undefined.

RANGE: This metric produces a positive real number less than or equal to the Volume of the module.

NECESSARY CONDITIONS: Every possible combination of characters that constitutes an operator or operand, the criteria for establishing the uniqueness of operators and operands, and the boundaries of a module must be defined for the context.

QUALIFICATIONS: The Language Level of a module is not necessarily equal to the sum of the values for each of its sub-modules. Thus, Language Level is not an additive quantity.

CRITICAL ANALYSIS: This measure is not necessarily strictly monotonic or repeatable. The problems are due to the counts that compose it. See the respective software metric data sheets for the details of the problems that they contribute to this metric.

Often, the Estimated Program Level is used in place of the Program Level. In this case, any lack of monotonicity in the Language Level would be produced primarily by the assumptions underlying the Estimated Program Level.

SQM RELATIONSHIPS: None.

TOOLS: LOGISCOPE, Verilog USA Inc., Alexandria, VA
PC-Metric, SET Laboratories, Inc., Mulino, OR

REFERENCES: Albrecht and Gaffney 1983; Arthur 1983; Basili, Selby, and Phillips 1983; Bulut 1974; Carver 1986; Coulter 1983; Curtis 1980; Elshoff 1976; Fitzsimmons 1978; Funami 1976; Gordon 1976, 1979; Halstead 1972, 1977, 1979; Halstead and Zislis 1973; Henry, Kafura, and Harris 1981; Jones 1978; Kafura and Reddy 1987; Lassez 1981; Li and Cheung 1987; Prather 1984; Shen, Conte, and Dunsmore 1983; Shneiderman 1980; Siyan 1989; Weyuker 1988; Zislis 1973; Zweben 1977

## $\hat{B}$ - Number of Bugs

This metric calculates an estimate of the number of bugs that a module of source code contains at the time of measurement.

RULE: $\hat{B} = V/E_0$ bugs

where V is the Volume and $E_0$ is the number of discriminations per bug. The Volume is as defined on a previous data sheet. $E_0$ is determined by an evaluation of a programmer's previous work. Experimentation has found that 3,200 discriminations per bug is a typical value.

A module is defined to be a set of code that is to have a measure assigned to it. It must be a well-defined entity in order to ensure that the count is repeatable.

DOMAIN: This measure only applies to that part of source code that implements an algorithm; the code must contain solely operators and operands. An algorithm with less than two operators and one operand is undefined.

RANGE: This metric produces a positive real number.

NECESSARY CONDITIONS: Every possible combination of characters that constitutes an operator or operand, the criteria for establishing the uniqueness of operators and operands, and the boundaries of a module must be defined for the context.

QUALIFICATIONS: While the measure is given in terms of a real number, the number of bugs should be rounded up to the next highest integer.

The Number of Bugs in a module is not necessarily equal to the sum of the values for each of its sub-modules. Thus, Number of Bugs is not an additive quantity.

CRITICAL ANALYSIS: This measure is not necessarily strictly monotonic or repeatable. Some of the problems are due to the counts that compose it. See the respective software metric data sheets for the details of the problems that they contribute to this metric.

Most of the problems are due to the variability associated with $E_0$. It is not necessarily monotonic or repeatable for all programming contexts. It is not necessarily monotonic because a programmer's knowledge in one discipline may result in fewer bugs for a program with a greater Volume than for a program with a lesser Volume. This could occur when the former program is in a discipline with which the programmer is familiar, but the latter program is not. It is not

necessarily repeatable because on some days a programmer will concentrate better than on others.

SQM RELATIONSHIPS: Maintainability

TOOLS: LOGISCOPE, Verilog USA Inc., Alexandria, VA
PC-Metric, SET Laboratories, Inc., Mulino, OR

REFERENCES: Albrecht and Gaffney 1983; Arthur 1983; Basili, Selby, and Phillips 1983; Bulut 1974; Carver 1986; Coulter 1983; Curtis 1980; Elshoff 1976; Fitzsimmons 1978; Funami 1976; Gordon 1976, 1979; Halstead 1972, 1977, 1979; Halstead and Zislis 1973; Henry, Kafura, and Harris 1981; Jones 1978; Kafura and Reddy 1987; Lassez 1981; Li and Cheung 1987; Prather 1984; Shen, Conte, and Dunsmore 1983; Shneiderman 1980; Siyan 1989; Weyuker 1988; Zislis 1973; Zweben 1977

## Information Flow

This metric calculates the complexity of a module of source code based on the size of the module and the flow of information within it.

RULE: The Information Flow Complexity (IFC) of a module is the sum of the IFCs for each procedure that composes it. The IFC for each procedure is calculated as follows:

$$IFC = Length \times (Fan\text{-}in \times Fan\text{-}out)^2$$

where Length is given by the number of lines of source code in the procedure, Fan-in is the number of local data flows that terminate at the procedure, and Fan-out is the number of local data flows that emanate from the procedure.

A module is defined as the set of procedures that either directly update or directly retrieve information from the particular data structure with which the module is associated.

Local data flow is data that is passed from procedure to procedure without going through the data structure.

DOMAIN: This measure only applies to source code that implements an algorithm; the code must perform some function that has inputs and outputs.

RANGE: This metric produces an integer greater than or equal to one, the IFC of a one statement procedure that has one input and one output.

NECESSARY CONDITIONS: The criteria for determining the local inputs and outputs must be defined for the context. The criteria for delimiting procedures and modules must be defined for the context.

QUALIFICATIONS: The IFC of a module is not necessarily equal to the IFC of each of its submodules. Thus, IFC is not an additive quantity.

CRITICAL ANALYSIS: This measure is not necessarily strictly monotonic or repeatable. A program reader may erroneously count local inputs, local outputs, or lines of source code. In this case, an increase in the measure does not represent an increase in the amount of information flow. Furthermore, a different reader will likely judge the attributes differently.

A problem with repeatability can also result from the arbitrariness of establishing module boundaries. The larger the module, the greater will be the IFC. Furthermore, as the module boundaries shift, the fan-in and fan-out counts will change. It is important that all measurements be based on an unambiguous specification for establishing module boundaries.

A software tool can also produce non-monotonic results. Although it has well-defined rules for identifying the inputs and outputs and for counting the lines of code, the rules will likely misinterpret some uncommon realization. Whether the count produced by a tool is right or wrong, it will certainly produce a repeatable result.

Despite these problems, if the program reader or the software tool follows carefully defined rules, either can generate highly reliable counts.

The IFC metric uses the source code to deduce the characteristics of the data flow of the algorithm solved by the program. The rules used for this step are dependent on the programming language and technology used. But it is primarily the data flow that is measured, not the program. Although the process for deducing the data flow is implementation dependent, the resulting value of the IFC metric has the advantage of being highly language independent, as well as highly independent of programmer style or experience. It is even highly independent of any changes in technology. Once the data flow complexity of an algorithm is calculated, most any program in any language that performs the same algorithm, whether poorly or well written, will have the same IFC value.

This complexity measure has been shown to represent the decrease in complexity that sometimes can be made by adding lines of code (Henry and Kafura 1984). It appears to measure a dimension of complexity somewhat different from that measured by Halstead's and McCabe's metrics (Kafura and Reddy 1987).

SQM RELATIONSHIPS: Complexity, Maintainability, Modifiability, Performance, Reliability, Simplicity, Understandability

TOOLS: No listing.

REFERENCES: Henry and Kafura 1984; Henry, Kafura, and Harris 1981; Kafura and Reddy 1987

## v(G) - Cyclomatic Complexity

This metric calculates the number of linearly independent program flow paths in the basis set of paths for a module of source code. Every possible path of program flow through a module can be represented as a linear combination of some subset of the basis set of paths.

RULE: $v(G) = e - n + 2p$

where e is the number of edges and n is the number of nodes in the program flow control graphs of all the modules. The variable p is the number of modules.

A module is defined to be a set of connected code that is to have a measure assigned to it. It must be a well-defined entity in order to ensure that the count is repeatable. To be connected, there must exist an unbroken chain of nodes and edges between any two nodes. A subroutine is a module, distinct from its calling module, unless the graph is drawn with the subroutine code substituted for the calling statement.

To draw the control graph, first, block the code into predicate nodes, collecting nodes, and function nodes. A predicate node is a block of code ending with a statement that branches to multiple locations. A collecting node begins with a labeled statement to which flow branches. When a predicate node is labeled, it is both a predicate node and a collecting node. Each strictly sequential block of statements between any two predicate or collecting nodes compose a function node.

Once the code is blocked, graphically represent each block with a circle and the flow of control between them with lines. In this graph, the circles are nodes and the lines are edges. The function nodes can be omitted from the graph without affecting the value of the Cyclomatic Complexity.

DOMAIN: This metric can be applied to any source code that can be blocked into nodes.

RANGE: This metric produces an integer greater than or equal to one, the Cyclomatic Complexity of code with purely sequential flow.

NECESSARY CONDITIONS: Every possible combination of characters that constitutes each type of node must be defined for the context. The boundaries of a module must be defined for the context.

QUALIFICATIONS: None.

CRITICAL ANALYSIS: This measure is not necessarily strictly monotonic or repeatable. A program reader may miss a node when drawing the graph. In this case, a net increase in the edge-node difference is not

represented by an increase in the measure. Furthermore, a different reader will likely judge the nodes differently for the same code.

A software tool can also produce non-monotonic results. Although it has well-defined rules for identifying nodes, the rules will likely misinterpret some uncommon realization. Whether the count produced by a tool is right or wrong, it will certainly produce a repeatable result.

Despite these problems, if the program reader or the software tool follows carefully defined rules, either can generate highly reliable counts.

SQM RELATIONSHIPS: Complexity, Maintainability, Modifiability, Modularity, Performance, Reliability, Simplicity, Testability, Understandability

TOOLS: Analysis of Complexity Tool (ACT), McCabe & Assocs., Columbia, MD
Battlemap Analysis Tool (BAT), McCabe & Assocs., Columbia, MD
LOGISCOPE, Verilog USA Inc., Alexandria, VA
PC-Metric, SET Laboratories, Inc., Mulino, OR

REFERENCES: Arthur 1983; Basili 1983; Basili and Hutchens 1983; Basili, Selby, and Phillips 1983; Carver 1986; Cote et al 1988; Crawford, McIntosh, and Pregibon 1985; Curtis 1980; Elshoff 1982, 1983; Evangelist 1983; Gaffney 1981; Harrison 1984; Henry, Kafura, and Harris 1981; Kafura and Reddy 1987; Li and Cheung 1987; Lind and Vairavan 1989; Mannino, Stoddard, and Sudduth 1990; McCabe 1976, 1982, 1989; McCabe and Butler 1989; McClure 1978; Myers 1977; Prather 1983, 1984; Ramamurthy and Melton 1988; Schneidewind and Hoffmann 1979; Shneiderman 1980; Siyan 1989; Walsh 1979; Ward 1989; Weyuker 1988

## v(G) - Cyclomatic Complexity
### Alternate Rule 1

This metric quantifies the number of linearly independent program flow paths in the basis set of paths for a module of source code. Every possible path of program flow through a module can be represented as a linear combination of some subset of the basis set of paths.

ALTERNATIVE RULE 1: The Cyclomatic Complexity of a module is equal to the number of regions delineated by its control graph, with an additional edge from the last node to the first. The control graph must be drawn with no crossing edges: it must be a planar graph.

A module is defined to be a set of connected code that is to have a measure assigned to it. It must be a well-defined entity in order to ensure that the count is repeatable. To be connected, there must exist an unbroken chain of nodes and edges between any two nodes. A subroutine is a module, distinct from its calling module, unless the graph is drawn with the subroutine code substituted for the calling statement.

To draw the control graph, first, block the code into predicate nodes, collecting nodes, and function nodes. A predicate node is a block of code ending with a statement that branches to multiple locations. A collecting node begins with a labeled statement to which flow branches. When a predicate node is labeled, it is both a predicate node and a collecting node. Each strictly sequential block of statements between any two predicate or collecting nodes compose a function node.

Once the code is blocked, graphically represent each block with a circle and the flow of control between them with lines. In this graph, the circles are nodes and the lines are edges. Also draw an edge from the exit node to the entrance node. The function nodes can be omitted from the graph without affecting the value of the Cyclomatic Complexity. The region surrounding the graph must also be counted as a region.

DOMAIN: This metric can be applied to any source code that can be blocked into nodes and can be drawn in a planar graph.

RANGE: This metric produces an integer greater than or equal to one, the Cyclomatic Complexity of code with purely sequential flow.

NECESSARY CONDITIONS: Every possible combination of characters that constitutes each type of node must be defined for the context. The boundaries of a module must be defined for the context.

QUALIFICATIONS: The control graphs of all structured and many unstructured programs can be drawn as planar graphs. Sometimes the control graph of unstructured code cannot be drawn as a planar graph.

CRITICAL ANALYSIS: This measure is not necessarily strictly monotonic or repeatable. A program reader may miss a node when drawing the graph. In this case, a net increase in the edge-node difference is not represented by an increase in the measure. Furthermore, a different reader will likely judge the nodes differently for the same code.

A software tool can also produce non-monotonic results. Although it has well-defined rules for identifying nodes, the rules will likely misinterpret some uncommon realization. Whether the count produced by a tool is right or wrong, it will certainly produce a repeatable result.

Despite these problems, if the program reader or the software tool follows carefully defined rules, either can generate highly reliable counts.

SQM RELATIONSHIPS: Complexity, Maintainability, Modifiability, Modularity, Performance, Reliability, Simplicity, Testability, Understandability

TOOLS: Analysis of Complexity Tool (ACT), McCabe & Assocs., Columbia, MD
Battlemap Analysis Tool (BAT), McCabe & Assocs., Columbia, MD
LOGISCOPE, Verilog USA Inc., Alexandria, VA
PC-Metric, SET Laboratories, Inc., Mulino, OR

REFERENCES: Arthur 1983; Basili 1983; Basili and Hutchens 1983; Basili, Selby, and Phillips 1983; Carver 1986; Cote et al 1988; Crawford, McIntosh, and Pregibon 1985; Curtis 1980; Elshoff 1982, 1983; Evangelist 1983; Gaffney 1981; Harrison 1984; Henry, Kafura, and Harris 1981; Kafura and Reddy 1987; Li and Cheung 1987; Lind and Vairavan 1989; Mannino, Stoddard, and Sudduth 1990; McCabe 1976, 1982, 1989; McCabe and Butler 1989; McClure 1978; Myers 1977; Prather 1983, 1984; Ramamurthy and Melton 1988; Schneidewind and Hoffmann 1979; Shneiderman 1980; Siyan 1989; Walsh 1979; Ward 1989; Weyuker 1988

## v(G) - Cyclomatic Complexity
### Alternate Rule 2

This metric quantifies the number of linearly independent program flow paths in the basis set of paths for a module of source code. Every possible path of program flow through a module can be represented as a linear combination of some subset of the basis set of paths.

ALTERNATIVE RULE 2: $v(G) = \pi + 1$

where $\pi$ is the number of control flow branch conditions in the module.

A module is defined to be a set of connected code that is to have a measure assigned to it. It must be a well-defined entity in order to ensure that the count is repeatable. To be connected, there must exist an unbroken chain of nodes and edges between any two nodes. A subroutine is a module, distinct from its calling module, unless the graph is drawn with the subroutine code substituted for the calling statement.

This formula for structured programs requires that for every predicate node there is exactly one collecting node, and that the program has unique entry and exit nodes. Exceptions to the rules always alter v(G). As long as there are relatively few exceptions to these rules, the Cyclomatic Complexity will be not be substantially affected. If there are a significant number of exceptions, the impact of multiple exits can be corrected by computing the following:

$$v(G) = \pi - s + 2$$

where s is the number of exits from the module (Harrison 1984). Of course, counting the number of exits requires additional analysis. If the code is primarily structured, the inaccuracy of the original equation is probably not significant enough to warrant the extra effort.

DOMAIN: This metric can be applied to any executable source code.

RANGE: This metric produces an integer greater than or equal to one, the Cyclomatic Complexity of code with purely sequential flow.

NECESSARY CONDITIONS: Every possible combination of characters that constitutes a branch condition must be defined for the context. The boundaries of a module must be defined for the context.

QUALIFICATIONS: This measure counts all conditions present, whether necessary or not. For example, code may contain a condition in a context where it will always be true. The Cyclomatic Complexity counts the number of conditions present in the algorithm, as implemented.

The Cyclomatic Complexity of a collection of independent modules, as in a main program and its subroutines, is simply the sum of the Cyclomatic Complexities for each module. For this rule, it is improper to calculate the total Cyclomatic Complexity by analyzing all the modules as if they were one continuous sequence of code. In this case, the value is deflated by the number of modules in the collection minus one.

CRITICAL ANALYSIS: This measure is not necessarily strictly monotonic or repeatable. A program reader may miss a condition when reading the code. In this case, a net increase in the number of branch conditions is not represented by an increase in the measure. Furthermore, a different reader will likely judge the conditions differently for the same code.

A software tool can also produce non-monotonic results. Although it has well-defined rules for identifying branch conditions, the rules will likely misinterpret some uncommon realization. Whether the count produced by a tool is right or wrong, it will certainly produce a repeatable result.

Despite these problems, if the program reader or the software tool follows carefully defined rules, either can generate highly reliable counts.

SQM RELATIONSHIPS: Complexity, Maintainability, Modifiability, Modularity, Performance, Reliability, Simplicity, Testability, Understandability

TOOLS: Analysis of Complexity Tool (ACT), McCabe & Assocs., Columbia, MD
Battlemap Analysis Tool (BAT), McCabe & Assocs., Columbia, MD
LOGISCOPE, Verilog USA Inc., Alexandria, VA
PC-Metric, SET Laboratories, Inc., Mulino, OR

REFERENCES: Arthur 1983; Basili 1983; Basili and Hutchens 1983; Basili, Selby, and Phillips 1983; Carver 1986; Cote et al 1988; Crawford, McIntosh, and Pregibon 1985; Curtis 1980; Elshoff 1982, 1983; Evangelist 1983; Gaffney 1981; Harrison 1984; Henry, Kafura, and Harris 1981; Kafura and Reddy 1987; Li and Cheung 1987; Lind and Vairavan 1989; Mannino, Stoddard, and Sudduth 1990; McCabe 1976, 1982, 1989; McCabe and Butler 1989; McClure 1978; Myers 1977; Prather 1983, 1984; Ramamurthy and Melton 1988; Schneidewind and Hoffmann 1979; Shneiderman 1980; Siyan 1989; Walsh 1979; Ward 1989; Weyuker 1988

## ev(G) - Essential Complexity

This metric calculates the amount of unstructured code in a module.

RULE: $ev(G) = v(G) - m$

where m is the number of proper subgraphs with unique entry and exit nodes.

A proper subgraph is any part of the graph that consists of only a "SEQUENCE" construct, an "IF-THEN-ELSE" construct, a "DO-UNTIL" construct, a "DO-WHILE" construct, or a "CASE" construct.

A module is defined to be a set of connected code that is to have a measure assigned to it. It must be a well-defined entity in order to ensure that the count is repeatable. To be connected, there must exist an unbroken chain of nodes and edges between any two nodes. A subroutine is a module, distinct from its calling module, unless the graph is drawn with the subroutine code substituted for the calling statement.

The smaller the value of ev(G), the greater the proportion of structured code. The Essential Complexity of a structured module is one. Greater values for the Essential Complexity of a module indicate the size of the nonstructured portion of the code; that code that does not conform to the proper subgraph structure (McCabe 1976).

Like the Cyclomatic Complexity, this measure also reflects the state of the algorithm, as implemented. It does not indicate whether the conditions, as implemented, are necessary for the algorithm.

To draw the control graph, first block the code into predicate nodes, processing nodes, and collecting nodes. A predicate node is a block of code ending with a statement that branches to multiple locations. A collecting node begins with a labeled statement to which flow branches. When a predicate node is labeled, it is both a predicate node and a collecting node. Each strictly sequential block of statements between any two predicate or collecting nodes compose a function node.

Once the code is blocked, graphically represent each block with a circle and the flow of control between them with lines. In this graph, the circles are nodes and the lines are edges. The function nodes can be omitted from the graph without affecting the value of the Cyclomatic Complexity.

DOMAIN: This metric can be applied to any source code that can be blocked into nodes.

RANGE: This metric produces an integer greater than or equal to one, the Essential Complexity of a totally structured module.

NECESSARY CONDITIONS: Every possible combination of characters that constitutes each type of node must be defined for the context. The boundaries of a module must be defined for the context.

QUALIFICATIONS: This measure only applies to executable source code.

CRITICAL ANALYSIS: This measure is not necessarily strictly monotonic or repeatable. A program reader may miss a node when drawing the graph. In this case, a net increase in the edge-node difference is not represented by an increase in the measure. Furthermore, a different reader will likely judge the nodes differently for the same code.

A software tool can also produce non-monotonic results. Although it has well-defined rules for identifying nodes, the rules will likely misinterpret some uncommon realization. Whether the count produced by a tool is right or wrong, it will certainly produce a repeatable result.

Despite these problems, if the program reader or the software tool follows carefully defined rules, either can generate highly reliable counts.

SQM RELATIONSHIPS: Complexity, Conciseness, Efficiency, Modularity, Performance, Reliability, Simplicity, Understandability

TOOLS: Analysis of Complexity Tool (ACT), McCabe & Assocs., Columbia, MD
Battlemap Analysis Tool (BAT), McCabe & Assocs., Columbia, MD
LOGISCOPE, Verilog USA Inc., Alexandria, VA

REFERENCES: Arthur 1983; Basili 1983; Basili and Hutchens 1983; Basili, Selby, and Phillips 1983; Carver 1986; Cote et al 1988; Crawford, McIntosh, and Pregibon 1985; Curtis 1980; Elshoff 1982, 1983; Evangelist 1983; Gaffney 1981; Harrison 1984; Henry, Kafura, and Harris 1981; Kafura and Reddy 1987; Li and Cheung 1987; Lind and Vairavan 1989; Mannino, Stoddard, and Sudduth 1990; McCabe 1976, 1982, 1989; McCabe and Butler 1989; McClure 1978; Myers 1977; Prather 1983, 1984; Ramamurthy and Melton 1988; Schneidewind and Hoffmann 1979; Shneiderman 1980; Siyan 1989; Walsh 1979; Ward 1989; Weyuker 1988

## FS.1(2) - Function Specificity

This metric calculates, for a single Computer Software Configuration Item (CSCI), the ratio of modules in which the functions of the module are described in the comments to total applicable modules.

RULE: For each module of the CSCI, answer the question:

"Is a description of the functions provided in the comments?" (Bowen, Wigle, and Tsai 1985)

Divide the total number of affirmative answers by the number of modules to which the question applied.

A CSCI is a set of code as defined in MIL-STD-483 and as used in DOD-STD-2167.

A module (or unit) is defined as a set of code to which a measure is assigned. It must be a well-defined entity in order to ensure that the count is repeatable.

DOMAIN: This metric can be applied to any module of source code that performs a function.

RANGE: This metric produces a real number between zero and one.

NECESSARY CONDITIONS: The combinations of characters that constitute a description of the function must be defined for the context. The boundaries of a module must be defined for the context.

QUALIFICATIONS: This single question requires both that the comments exist and that they contain a description of the functions.

CRITICAL ANALYSIS: This measure is not necessarily strictly monotonic or repeatable. A program reader may erroneously judge that a description in a comment satisfies the requirement. In this case, an increase in the measure does not represent an increase in how specifically the function is described. Furthermore, a different reader will likely judge the sufficiency of the description differently for the same code. At best, it only establishes that the function was described, not how well it was described.

A software tool can also produce non-monotonic results. Although it has well-defined rules for evaluating the descriptions, the rules will likely misinterpret some uncommon implementation. Whether the count produced by a tool is right or wrong, the tool will certainly produce a repeatable result.

Despite these problems, if the program reader or the software tool follows carefully defined rules, either can generate highly reliable counts.

SQM RELATIONSHIPS: Reusability

TOOLS: ***, METRIQS, San Juan Capistrano, CA
AdaMAT, Dynamics Research Corporation, Andover, MA
AMS, Rome Air Development Center, Griffiss AFB, NY
(*** This tool is used internally to support their consulting service.)

REFERENCES: Boehm et al 1978; Bowen, Wigle, and Tsai 1985; Lasky, Kaminsky, and Boaz 1990; McCall, Richards, and Walters 1977; Millman and Curtis 1980; Murine 1983, 1985[1], 1985[2], 1986, 1988; Pierce, Hartley, and Worrells 1987; Shneiderman 1980; Sunazuka, Azuma, and Yamagishi 1985; Warthman 1987

## GE.1(1) - Unit Referencing

This metric calculates, for a single Computer Software Configuration Item (CSCI), the ratio of modules that are sufficiently general that they are called by more than one other module to total number of modules.

RULE: For each CSCI, answer the question:

"How many units are called by more than one other unit?" (Bowen, Wigle, and Tsai 1985)

Divide the answer by the number of modules in the CSCI.

A CSCI is a set of code as defined in MIL-STD-483 and as used in DOD-STD-2167.

A module (or unit) is defined as a set of code to which a measure is assigned. It must be a well-defined entity in order to ensure that the count is repeatable.

DOMAIN: This metric can be applied to any module of source code.

RANGE: This metric produces a real number between zero and one.

NECESSARY CONDITIONS: The combinations of characters that constitute a call must be defined for the context. The boundaries of a module must be defined for the context.

QUALIFICATIONS: None.

CRITICAL ANALYSIS: This measure is not necessarily monotonic. A module may be sufficiently general to be called by many other modules in various applications, but only be required by one other module in a particular application. Thus, it is not a measure of the generality of the module called.

This metric is highly repeatable, even when determined by a program reader, since a module is either called or not called.

SQM RELATIONSHIPS: Expandability, Flexibility, Reusability

TOOLS: ***, METRIQS, San Juan Capistrano, CA
AdaMAT, Dynamics Research Corporation, Andover, MA
AMS, Rome Air Development Center, Griffiss AFB, NY
(*** This tool is used internally to support their consulting service.)

REFERENCES: Boehm et al 1978; Bowen, Wigle, and Tsai 1985; Lasky, Kaminsky, and Boaz 1990; McCall, Richards, and Walters 1977; Millman and Curtis 1980; Murine 1983, 1985a, 1985b, 1986, 1988; Pierce, Hartley, and Worrells 1987; Shneiderman 1980; Sunazuka, Azuma, and Yamagishi 1985; Warthman 1987

## MO.1(3) - Modular Implementation

This metric calculates, for a single Computer Software Configuration Item (CSCI), the ratio of modules that are sufficiently modularized (so that they are not too large) to total applicable modules.

RULE: For each module of the CSCI, answer the question:

"Are the estimated lines of source code for this unit 100 lines or less, excluding comments?" (Bowen, Wigle, and Tsai 1985)

Divide the total number of affirmative answers by the number of modules in the set to which the question applied.

A CSCI is a set of code as defined in MIL-STD-483 and as used in DOD-STD-2167.

A module (or unit) is defined as a set of code to which a measure is assigned. It must be a well-defined entity in order to ensure that the count is repeatable.

DOMAIN: This metric can be applied to any module of source code that contains executable statements.

RANGE: This metric produces a real number between zero and one.

NECESSARY CONDITIONS: The combinations of characters that constitute a comment, the combinations of characters that constitute lines of source code, and the boundaries of a module must be defined for the context.

QUALIFICATIONS: The count of lines of source code is called an estimate because it is recognized that there is no one standard for defining code that constitutes source code.

CRITICAL ANALYSIS: This measure is not necessarily strictly monotonic or repeatable. A program reader may erroneously judge that a line of code is not source code. In this case, an increase in the measure does not represent an increase in the modularity of the implementation. Furthermore, a different reader will likely judge lines of code differently for the same code.

A software tool can also produce non-monotonic results. Although it has well-defined rules for identifying the comments and lines of source code, the rules will likely misinterpret some uncommon implementation. Whether the count produced by a tool is right or wrong, the tool will certainly produce a repeatable result.

Despite these problems, if the program reader or the software tool follows carefully defined rules, either can generate highly reliable counts.

SQM RELATIONSHIPS: Expandability, Flexibility, Interoperability, Maintainability, Portability, Reusability, Survivability, Verifiability

TOOLS: ***, METRIQS, San Juan Capistrano, CA
AdaMAT. Dynamics Research Corporation, Andover, MA
AMS, Rome Air Development Center, Griffiss AFB, NY
(*** This tool is used internally to support their consulting service.)

REFERENCES: Boehm et al 1978; Bowen, Wigle, and Tsai 1985; Lasky, Kaminsky, and Boaz 1990; McCall, Richards, and Walters 1977; Millman and Curtis 1980; Murine 1983, 1985a, 1985b, 1986, 1988; Pierce, Hartley, and Worrells 1987; Shneiderman 1980; Sunazuka, Azuma, and Yamagishi 1985; Warthman 1987

## MO.1(7) - Modular Implementation

This metric calculates, for a single Computer Software Configuration Item (CSCI), the ratio of modules that always transfer control back to the calling unit to total applicable modules.

RULE: For each module of the CSCI, answer the question:

"Is control always returned to the calling unit when execution is completed?" (Bowen, Wigle, and Tsai 1985)

Divide the total number of affirmative answers by the number of modules in the set to which the question applied.

A CSCI is a set of code as defined in MIL-STD-483 and as used in DOD-STD-2167.

A module (or unit) is defined as a set of code to which a measure is assigned. It must be a well-defined entity in order to ensure that the count is repeatable.

DOMAIN: This metric can be applied to any module of source code that is called by another module.

RANGE: This metric produces a real number between zero and one.

NECESSARY CONDITIONS: The combinations of characters that determine when execution of a module is completed, what constitutes a calling module, and the boundaries of a module must be defined for the context.

QUALIFICATIONS: None.

CRITICAL ANALYSIS: This measure is not necessarily monotonic. A module may transfer control back to the calling module, but not to the next statement in the sequence. This would break up the modularity of the calling module. In this case, an increase in the measure does not represent an increase in the modularity of the implementation.

This metric is highly repeatable, even when determined by a program reader, since control is either returned to the calling module or is not returned to the calling module.

SQM RELATIONSHIPS: Expandability, Flexibility, Interoperability, Maintainability, Portability, Reusability, Survivability, Verifiability

TOOLS: ***, METRIQS, San Juan Capistrano, CA
AdaMAT, Dynamics Research Corporation, Andover, MA
AMS, Rome Air Development Center, Griffiss AFB, NY
(*** This tool is used internally to support their consulting service.)

REFERENCES: Boehm et al 1978; Bowen, Wigle, and Tsai 1985; Lasky, Kaminsky, and Boaz 1990; McCall, Richards, and Walters 1977; Millman and Curtis 1980; Murine 1983, 1985a, 1985b, 1986, 1988; Pierce, Hartley, and Worrells 1987; Shneiderman 1980; Sunazuka, Azuma, and Yamagishi 1985; Warthman 1987

## SD.2(1) - Effectiveness of Comments

This metric calculates, for a single Computer Software Configuration Item (CSCI), the ratio of modules in which header comments are complete to total applicable modules.

RULE: For each module of the CSCI, answer the question:

"Are there prologue comments which contain all information in accordance with the established standard?" (Bowen, Wigle, and Tsai 1985)

Divide the total number of affirmative answers by the number of modules in the set to which the question applied.

A CSCI is a set of code as defined in MIL-STD-483 and as used in DOD-STD-2167.

A module (or unit) is defined as a set of code to which a measure is assigned. It must be a well-defined entity in order to ensure that the count is repeatable.

DOMAIN: This metric can be applied to any source code for which a standard has been set for header comments.

RANGE: This metric produces a real number between zero and one.

NECESSARY CONDITIONS: The combinations of characters that identify a prologue comment, the combinations of characters that must constitute each required piece of information, and the boundaries of a module must be defined for the context.

QUALIFICATIONS: This question requires that the comment exist and that it contain the specified information. The only case in which the question does not apply is for a module for which no prologue comments are required.

CRITICAL ANALYSIS: This measure is not necessarily strictly monotonic or repeatable. A program reader may erroneously judge that a piece of information in a comment satisfies the requirement. In this case, an increase in the measure does not represent an increase in the comment effectiveness. Furthermore, a different reader will likely judge the sufficiency of the comments differently for the same code.

A software tool can also produce non-monotonic results. Although it has well-defined rules for evaluating the information, the rules will likely misinterpret some uncommon implementation. Whether the count produced by a tool is right or wrong, the tool will certainly produce a repeatable result.

Despite these problems, if the program reader or the software tool follows carefully defined rules, either can generate highly reliable counts.

SQM RELATIONSHIPS: Expandability, Flexibility, Maintainability, Portability, Reusability, Verifiability

TOOLS: ***, METRIQS, San Juan Capistrano, CA
AdaMAT, Dynamics Research Corporation, Andover, MA
AMS, Rome Air Development Center, Griffiss AFB, NY
(*** This tool is used internally to support their consulting service.)

REFERENCES: Boehm et al 1978; Bowen, Wigle, and Tsai 1985; Lasky, Kaminsky, and Boaz 1990; McCall, Richards, and Walters 1977; Millman and Curtis 1980; Murine 1983, 1985a, 1985b, 1986, 1988; Pierce, Hartley, and Worrells 1987; Shneiderman 1980; Sunazuka, Azuma, and Yamagishi 1985; Warthman 1987

## APPENDIX B - SOFTWARE QUALITY FACTOR DATA SHEETS

This appendix contains individual data sheets on each of the most prevalent software quality factors encountered in this study. The data sheets are a summary of the concepts, constraints, criticisms, and other data needed to understand and critically evaluate the application of Software Quality Metrics (SQM) that measure these quality factors.

Although the data sheets are not primarily intended as a guide for applying SQM, some of the combinations of these data sheets with the Software Metric Data Sheets that they reference contain sufficient information that an SQM could be properly and fully calculated from them. On the other hand, some of the SQM are so extensive that the data sheets could not encompass all the necessary parts. In either case, the Technical Report, Software Quality Metrics (N. VanSuetendael and D. Elwell 1991), discusses the most prominent SQM in detail. See the references listed on these data sheets for complete coverage of a particular quality factor. See the references listed on the Software Metric Data Sheets for complete coverage of a particular software metric that an SQM uses to measure a quality factor.

Each data sheet covers only one factor and they all contain the same fields of information. The fields are defined only as they apply to measuring source code. Many of the metrics can be applied more broadly, but the other possibilities are not addressed here. The data sheets are arranged alphabetically by factor name.

The "CLAIM" field is filled by the particular quality factor description or definition that is the most comprehensive, without being too general, and the most independent. The goal of identifying quality factors is to come up with a set of factors that are independent of one another and that comprehensively cover the full range of quality factors of concern for software.

The "SQM RELATIONSHIPS" field lists those software metrics that are known to be related to the particular quality factor. This serves as a cross-reference to the applicable Software Metric Data Sheet of appendix A. However, not all of the referenced software metrics are addressed in appendix A. See the articles listed in the bibliography for further information.

The "TOOLS" information is supplied on an as-available basis. It is not to be taken as a comprehensive list of tools available. Nor is the information to be taken as a recommendation.

SOFTWARE QUALITY FACTOR DATA SHEET

## Accuracy

This quality factor addresses the concern that programs provide the precision required for each output. Accuracy is important because most computer manipulations are not exact, but are limited approximations.

CLAIM: Some researchers claim that

"A software product possesses accuracy to the extent that its outputs are sufficiently precise to satisfy their intended use." (Boehm et al 1978).

DOMAIN: This factor only applies to source code for which the required precision has been explicitly stated for each output.

NECESSARY CONDITIONS: The required precision must be defined for the context. The outputs must be defined for the context.

QUALIFICATIONS: None.

CRITICAL ANALYSIS: This factor does not address whether a calculation or output should or should not be present. The former is the concern of completeness, the latter of conciseness. Nor does it address whether the proper quantity is being calculated or output. That is the concern of correctness.

SQM RELATIONSHIPS: RADC's AC.1(7)

TOOLS: ***, METRIQS, San Juan Capistrano, CA
AdaMAT, Dynamics Research Corporation, Andover, MA
AMS, Rome Air Development Center, Griffiss AFB, NY
(*** This tool is used internally to support their consulting service.)

REFERENCES: Boehm et al 1978; Bowen, Wigle, and Tsai 1985; Lasky, Kaminsky, and Boaz 1989; McCall, Richards, and Walters 1977; Millman and Curtis 1980; Murine 1983, 1985a, 1985b, 1986, 1988; Pierce, Hartley, and Worrells 1987; Shneiderman 1980; Sunazuka, Azuma, and Yamagishi 1985; Warthman 1987

SOFTWARE QUALITY FACTOR DATA SHEET

## Clarity

This quality factor addresses the concern that programs be easily understood by people. Programmers are concerned particularly that source listings be easily understood. Users are concerned that the human interface of the program is easily understood.

CLAIM: One researcher claims that software clarity is a

"Measure of how clear a program is, i.e., how easy it is to read, understand, and use" (Kosarajo and Ledgard, as quoted in McCall, Richards, and Walters 1977).

Although users are concerned about the clarity of the program dialogue, it is best to let the users' concern for clarity be handled by the quality factor, usability. Then clarity only addresses the ease of reading and understanding the program.

DOMAIN: This factor applies to any source code. Programming is such an iterative process that the code should be programmed with clarity at least to aid the programmer who writes some code, reviews it, and then writes some more.

NECESSARY CONDITIONS: The audience to which the code must be clear must be defined for the context. The criteria for determining that the program is clear must be determined for the context.

QUALIFICATIONS: This is a programmer-oriented quality factor, not a user-oriented one.

CRITICAL ANALYSIS: Program clarity can suffer either because the necessary information is not provided, or because the information provided is not well presented. Either problem impacts program clarity, since this quality factor is concerned about the net impact on the ease of understanding the program, regardless of the cause. In this case, clarity can be equated with self-descriptiveness.

SQM RELATIONSHIPS: Halstead's Effort, RADC's SD.2(1), SD.2(3)

TOOLS: ***, METRIQS, San Juan Capistrano, CA
AdaMAT, Dynamics Research Corporation, Andover, MA
AMS, Rome Air Development Center, Griffiss AFB, NY
LOGISCOPE, Verilog USA Inc., Alexandria, VA
PC-Metric, SET Laboratories, Inc., Mulino, OR
(*** This tool is used internally to support their consulting service.)

REFERENCES: Albrecht and Gaffney 1983; Arthur 1983; Basili, Selby, and Phillips 1983; Boehm et al 1978; Bowen, Wigle, and Tsai 1985; Bulut 1974; Carver 1986; Coulter 1983; Curtis 1980; Elshoff 1976; Fitzsimmons 1978; Funami 1976; Gordon 1976, 1979; Halstead 1972, 1973, 1977, 1979; Halstead and Zislis 1973; Henry, Kafura, and Harris 1981; Jones 1978; Kafura and Reddy 1987; Lasky, Kaminsky, and Boaz 1989; Lassez 1981; Li and Cheung 1987; McCall, Richards, and Walters 1977; Millman and Curtis 1980; Murine 1983, 1985a, 1985b, 1986, 1988; Pierce, Hartley, and Worrells 1987; Prather 1984; Shen, Conte, and Dunsmore 1983; Shneiderman 1980; Siyan 1989; Sunazuka, Azuma, and Yamagishi 1985; Warthman 1987; Weyuker 1988; Zislis 1973; Zweben 1977

SOFTWARE QUALITY FACTOR DATA SHEET

## Completeness

This quality factor addresses the concern that program functions be implemented completely.

CLAIM: Some researchers claim that software completeness is determined by

"Those characteristics of software which provide full implementation of the functions required" (Bowen, Wigle, and Tsai 1985).

DOMAIN: This factor only applies to source code for which the required functions have been explicitly stated.

NECESSARY CONDITIONS: The required functions and their extent must be defined for the context.

QUALIFICATIONS: None.

CRITICAL ANALYSIS: This definition of completeness distinguishes functional completeness from the completeness of the source code. The latter encompasses additional attributes, such as program comments, that are desired by programmers rather than the user. Furthermore, these attributes are already addressed by the understandability and clarity quality factors.

SQM RELATIONSHIPS: RADC's CP.1(2)

TOOLS: ***, IETRIQS, San Juan Capistrano, CA
AdaMAT, Dynamics Research Corporation, Andover, MA
AMS, Rome Air Development Center, Griffiss AFB, NY
(*** This tool is used internally to support their consulting service.)

REFERENCES: Boehm et al 1978; Bowen, Wigle, and Tsai 1985; Lasky, Kaminsky, and Boaz 1989; McCall, Richards, and Walters 1977; Millman and Curtis 1980; Murine 1983, 1985a, 1985b, 1986, 1988; Pierce, Hartley, and Worrells 1987; Shneiderman 1980; Sunazuka, Azuma, and Yamagishi 1985; Wartham 1987

SOFTWARE QUALITY FACTOR DATA SHEET

## Complexity

This quality factor addresses the concern that programs not be complex. There are many types of program complexity, but the complexity of concern is the complexity that is constructed in the mind of a programmer, the psychological complexity.

CLAIM: The authors claim that

> The complexity of a program is the extent to which it is involved or intricate, composed of many interwoven parts.

DOMAIN: This factor applies to any source code.

NECESSARY CONDITIONS: The criteria for determining complexity must be defined for the context.

QUALIFICATIONS: Although the psychological complexity is the important one, a given SQM may claim that psychological complexity is totally encompassed by the complexity of a particular structure of a program. In this case, the two complexities can be used interchangeably. It is important that the type of complexity measure be qualified. Different types must not be directly compared.

This is a programmer-oriented quality factor, not a user-oriented one.

CRITICAL ANALYSIS: This quality factor addresses the complexity of program code, not the complexity of the user interface. Programmers are concerned that programs not be complex so that they can easily and assuredly satisfy programmer and user needs. Any complexity that may exist for the user is addressed by the quality factor, usability.

SQM RELATIONSHIPS: Albrecht's Function Points; Ejiogu's Structural Complexity; Halstead's Length, Volume, and Effort; Henry's Information Flow; McCabe's Cyclomatic Complexity and Essential Complexity; RADC's SI.3(1)

TOOLS: ***, METRIQS, San Juan Capistrano, CA
AdaMAT, Dynamics Research Corporation, Andover, MA
AMS, Rome Air Development Center, Griffiss AFB, NY
Analysis of Complexity Tool (ACT), McCabe & Assocs., Columbia, MD
Battlemap Analysis Tool (BAT), McCabe & Assocs., Columbia, MD
Checkpoint, Software Productivity Research, Inc., Burlington, MA
COMPLEXIMETER, Softmetrix, Inc., Chicago, IL
LOGISCOPE, Verilog USA Inc., Alexandria, VA
PC-Metric, SET Laboratories, Inc., Mulino, OR
SIZE PLANNER, Quantitative Software Management, Inc., McClean, VA

(*** This tool is used internally to support their consulting service.)

REFERENCES: Albrecht 1979, 1985; Albrecht and Gaffney 1983; Arthur 1983; Basili 1983; Basili and Hutchens 1983; Basili, Selby, and Phillips 1983; Behrens 1983; Boehm et al 1978; Bowen, Wigle, and Tsai 1985; Bulut 1974; Carver 1986; Cote et al 1988; Coulter 1983; Crawford, McIntosh, and Pregibon 1985; Curtis 1980; Drummond 1985; Ejiogu 1984a, 1984b, 1987, 1988, 1990; Elshoff 1976, 1982, 1983; Evangelist 1983; Fitzsimmons 1978; Funami 1976; Gaffney 1981; Gordon 1976, 1979; Halstead 1972, 1973, 1977, 1979; Halstead and Zislis 1973; Harrison 1984; Henry 1979; Henry and Kafura 1984; Henry, Kafura, and Harris 1981; Jones 1978, 1988; Kafura and Henry 1982; Kafura and Reddy 1987; Lasky, Kaminsky, and Boaz 1989; Lassez 1981; Li and Cheung 1987; Lind and Vairavan 1989; Low and Jeffrey 1990; Mannino, Stoddard, and Sudduth 1990; McCabe 1976, 1982, 1989; McCabe and Butler 1989; McCall, Richards, and Walters 1977; McClure 1978; Millman and Curtis 1980; Murine 1983, 1985a, 1985b, 1986, 1988; Myers 1977; Pierce, Hartley, and Worrells 1987; Prather 1983, 1984; Ramamurthy and Melton 1988; Schneidewind and Hoffmann 1979; Shen, Conte, and Dunsmore 1983; Shneiderman 1980; Siyan 1989; Sunazuka, Azuma, and Yamagishi 1985; Walsh 1979; Ward 1989; Warthman 1987; Weyuker 1988; Zislis 1973; Zweben 1977

SOFTWARE QUALITY FACTOR DATA SHEET

## Conciseness

This quality factor addresses the concern that programs not contain any extraneous information.

CLAIM: Some researchers claim that software conciseness is determined by

"The ability to satisfy functional requirements with minimum amount of software" (McCall, Richards, and Walters 1977).

DOMAIN: This factor only applies to source code for which the functional requirements have been explicitly stated.

NECESSARY CONDITIONS: The functional requirements must be defined for the context.

QUALIFICATIONS: This is a programmer-oriented quality factor, not a user-oriented one.

CRITICAL ANALYSIS: This quality factor basically addresses how efficiently the source code was used. Any efficiency quality factors may already encompass this quality factor. Another word used for efficiency is *effectiveness*.

SQM RELATIONSHIPS: Halstead's Potential Volume, McCabe's Essential Complexity,

TOOLS: Analysis of Complexity Tool (ACT), McCabe & Assocs., Columbia, MD
Battlemap Analysis Tool (BAT), McCabe & Assocs., Columbia, MD
LOGISCOPE, Verilog USA Inc., Alexandria, VA

REFERENCES: Halstead 1977; McCabe 1976, 1982

SOFTWARE QUALITY FACTOR DATA SHEET

## Consistency

This quality factor addresses the concern that the source code syntax and constructs in programs be implemented uniformly.

CLAIM: Some researchers claim that software consistency is determined by

> "Those characteristics of software which provide for uniform design and implementation techniques and notation" (Bowen, Wigle, and Tsai 1985).

DOMAIN: This factor applies to any source code.

NECESSARY CONDITIONS: The standard notation must be defined for the context.

QUALIFICATIONS: This is a programmer-oriented quality factor, not a user-oriented one.

CRITICAL ANALYSIS: This quality factor addresses self-consistency. External consistency is more properly addressed by the quality factors, completeness, correctness, and performance, since they all compare the implementation with an external standard.

SQM RELATIONSHIPS: RADC's software metrics

TOOLS: ***, METRIQS, San Juan Capistrano, CA
AdaMAT, Dynamics Research Corporation, Andover, MA
AMS, Rome Air Development Center, Griffiss AFB, NY
(*** This tool is used internally to support their consulting service.)

REFERENCES: Boehm et al 1978; Bowen, Wigle, and Tsai 1985; Lasky, Kaminsky, and Boaz 1989; McCall, Richards, and Walters 1977; Millman and Curtis 1980; Murine 1983, 1985a, 1985b, 1986, 1988; Pierce, Hartley, and Worrells 1987; Shneiderman 1980; Sunazuka, Azuma, and Yamagishi 1985; Wartham 1987

SOFTWARE QUALITY FACTOR DATA SHEET

## Correctness

This quality factor addresses the concern that software design and documentation formats conform to the specifications and standards set for them. It is not concerned with any content affecting software operation or performance.

CLAIM: Some researchers claim that software correctness is determined by the

"Extent to which the software conforms to its specifications and standards" (Bowen, Wigle, and Tsai 1985).

DOMAIN: This factor only applies to source code for which specifications and standards have been explicitly stated.

NECESSARY CONDITIONS: The specifications and standards must be defined for the context.

QUALIFICATIONS: This is a programmer-oriented quality factor, not a user-oriented one.

CRITICAL ANALYSIS: This quality factor must not be confused with completeness. Although each specification may have been completely addressed, it may have been addressed incorrectly. Completeness of a function must be satisfied before its correctness can be established.

SQM RELATIONSHIPS: RADC's CP.1(2)

TOOLS: ***, METRIQS, San Juan Capistrano, CA
AdaMAT, Dynamics Research Corporation, Andover, MA
AMS, Rome Air Development Center, Griffiss AFB, NY
(*** This tool is used internally to support their consulting service.)

REFERENCES: Boehm et al 1978; Bowen, Wigle, and Tsai 1985; Lasky, Kaminsky, and Boaz 1989; McCall, Richards, and Walters 1977; Millman and Curtis 1980; Murine 1983, 1985a, 1985b, 1986, 1988; Pierce, Hartley, and Worrells 1987; Shneiderman 1980; Sunazuka, Azuma, and Yamagishi 1985; Wartham 1987

SOFTWARE QUALITY FACTOR DATA SHEET

## Efficiency

This quality factor addresses the concern that programs optimally use any computer resources.

CLAIM: Some researchers claim that

> "A software product possesses the characteristic Efficiency to the extent that it fulfills its purpose without waste of resources." (Boehm et al 1978)
>
> In practice, one is not usually concerned with general efficiency, but with the efficiency of utilization of a particular resource, for example, efficient use of processing time.

DOMAIN: This factor only applies to source code for which the purpose has been explicitly stated.

NECESSARY CONDITIONS: The criteria for establishing that a resource is wasted must be defined for the context.

QUALIFICATIONS: None.

CRITICAL ANALYSIS: This quality factor must not be confused with how the software rates for a particular performance measure. Performance is concerned with how close the measure comes to a standard. Efficiency requires that the standard be met, and then addresses how effectively it was met.

SQM RELATIONSHIPS: Halstead's Potential Volume, McCabe's Essential Complexity, RADC's EP.2(4)

TOOLS: ***, METRIQS, San Juan Capistrano, CA
AdaMAT, Dynamics Research Corporation, Andover, MA
AMS, Rome Air Development Center, Griffiss AFB, NY
Analysis of Complexity Tool (ACT), McCabe & Assocs., Columbia, MD
Battlemap Analysis Tool (BAT), McCabe & Assocs., Columbia, MD
(*** This tool is used internally to support their consulting service.)

REFERENCES: Boehm et al 1978; Bowen, Wigle, and Tsai 1985; Halstead 1977; Lasky, Kaminsky, and Boaz 1989; McCabe 1976, 1982; McCall, Richards, and Walters 1977; Millman and Curtis 1980; Murine 1983, 1985a, 1985b, 1986, 1988; Pierce, Hartley, and Worrells 1987; Shneiderman 1980; Sunazuka, Azuma, and Yamagishi 1985; Wartham 1987

SOFTWARE QUALITY FACTOR DATA SHEET

## Expandability

This quality factor addresses the concern that program limitations be easy to extend.

CLAIM: Some researchers claim that software expandability is determined by the

> "Relative effort [required] to increase the software capability or performance by enhancing current functions or by adding new functions or data" (Bowen, Wigle, and Tsai 1985).

DOMAIN: This factor applies to any source code.

NECESSARY CONDITIONS: The criteria for distinguishing a change from an expansion must be defined for the context.

QUALIFICATIONS: None.

CRITICAL ANALYSIS: This quality factor must not be confused with flexibility, maintainability, or reusability. Expandability assumes that all existing functions are still performed, but possibly added to or enhanced. Expandability also assumes that there is no change in the context in which the program is used. Another word used to describe this quality factor is augmentability.

SQM RELATIONSHIPS: RADC's GE.1(1), GE.2(2), MO.1(3), MO.1(5), MO.1(7), SD.2(1), SD.2(3), SI.3(1)

TOOLS: ***, METRIQS, San Juan Capistrano, CA
AdaMAT, Dynamics Research Corporation, Andover, MA
AMS, Rome Air Development Center, Griffiss AFB, NY
(*** This tool is used internally to support their consulting service.)

REFERENCES: Boehm et al 1978; Bowen, Wigle, and Tsai 1985; Lasky, Kaminsky, and Boaz 1989; McCall, Richards, and Walters 1977; Millman and Curtis 1980; Murine 1983, 1985a, 1985b, 1986, 1988; Pierce, Hartley, and Worrells 1987; Shneiderman 1980; Sunazuka, Azuma, and Yamagishi 1985; Wartham 1987

SOFTWARE QUALITY FACTOR DATA SHEET

## Flexibility

This quality factor addresses the concern that programs be easy to change to meet different requirements, with no change in the context.

CLAIM: Some researchers claim that software flexibility is determined by the

> "Ease of effort for changing the software missions, functions, or data to satisfy other requirements" (Bowen, Wigle, and Tsai 1985).

DOMAIN: This factor applies to any source code.

NECESSARY CONDITIONS: The criteria for distinguishing different requirements from a change in context or an expansion must be defined for the context.

QUALIFICATIONS: None.

CRITICAL ANALYSIS: This quality factor must not be confused with expandability, maintainability, or reusability. Expandability addresses the simple case of satisfying extended requirements, rather than different requirements. In order to keep flexibility independent of expandability, flexibility should assess the ability to accommodate different requirements. Maintainability assumes no change in the functions of a program. Reusability requires a change of context.

SQM RELATIONSHIPS: RADC's GE.1(1), GE.2(2), MO.1(3), MO.1(5), MO.1(7), SD.2(1), SD.2(3), SI.3(1)

TOOLS: ***, METRIQS, San Juan Capistrano, CA
AdaMAT, Dynamics Research Corporation, Andover, MA
AMS, Rome Air Development Center, Griffiss AFB, NY
(*** This tool is used internally to support their consulting service.)

REFERENCES: Boehm et al 1978; Bowen, Wigle, and Tsai 1985; Lasky, Kaminsky, and Boaz 1989; McCall, Richards, and Walters 1977; Millman and Curtis 1980; Murine 1983, 1985a, 1985b, 1986, 1988; Pierce, Hartley, and Worrells 1987; Shneiderman 1980; Sunazuka, Azuma, and Yamagishi 1985; Wartham 1987

SOFTWARE QUALITY FACTOR DATA SHEET

## Integrity

This quality factor addresses the concern that programs must continue to perform their function even under adverse conditions: inputs that are unexpected, improper, or harmful.

CLAIM: Some researchers claim that software integrity is determined by the

> "Ability of software to prevent purposeful or accidental damage to the data or software" (United States Army Computer Systems Command as quoted in McCall, Richards, and Walters 1977).
>
> This definition assumes that damage to the data or software will result in the loss of function.

DOMAIN: This factor applies to any source code.

NECESSARY CONDITIONS: The adverse conditions must be defined for the context.

QUALIFICATIONS: This definition assumes that program integrity is an issue only when the program is operating. When it is not operating, the operating system must guard the program and its data from damage; the program cannot.

CRITICAL ANALYSIS: This quality factor must not be confused with security or reliability. Security contributes directly to integrity, but security is only concerned with unauthorized access to information. When a program is not secure, the confidentiality of the data is compromised, whether or not the program is subjected to adverse conditions. Similarly, reliability is determined by program failures that occur during execution under the specified conditions, not during adverse conditions.

SQM RELATIONSHIPS: RADC's software metrics

TOOLS: ***, METRIQS, San Juan Capistrano, CA
AdaMAT, Dynamics Research Corporation, Andover, MA
AMS, Rome Air Development Center, Griffiss AFB, NY
(*** This tool is used internally to support their consulting service.)

REFERENCES: Boehm et al 1978; Bowen, Wigle, and Tsai 1985; Lasky, Kaminsky, and Boaz 1989; McCall, Richards, and Walters 1977; Millman and Curtis 1980; Murine 1983, 1985a, 1985b, 1986, 1988; Pierce, Hartley, and Worrells 1987; Shneiderman 1980; Sunazuka, Azuma, and Yamagishi 1985; Wartham 1987

SOFTWARE QUALITY FACTOR DATA SHEET

## Maintainability

This quality factor addresses the concern that programs be easy to fix, once a failure is identified.

CLAIM: Some researchers claim that software maintainability is determined by the

"Ease of effort for locating and fixing a software failure within a specified time period" (Bowen, Wigle, and Tsai 1985).

DOMAIN: This factor only applies to source code for which operational failure and restoration is defined.

NECESSARY CONDITIONS: The criteria for program failures must be defined for the context.

QUALIFICATIONS: In general, maintainability encompasses the ease of identifying and making any changes to programs. But when used as one of a set of quality factors, maintainability is usually restricted to changes that are made to fix software failures as in the definition above. Often, maintainability is further restricted to the ease of making the fix, once the cause for the failure is found. This restriction keeps it highly independent. In particular, it makes maintainability independent of complexity and understandability.

CRITICAL ANALYSIS: This quality factor must not be confused with expandability, flexibility, modifiability, portability, or reusability. None of these quality factors apply to changes made in response to software failures. They address the concerns related to making changes for other reasons.

SQM RELATIONSHIPS: Halstead's Effort; Henry's Information Flow; McCabe's Cyclomatic Complexity; RADC's MO.1(3), MO.1(5), MO.1(7), SD.2(1), SD.2(3), SI.3(1)

TOOLS: ***, METRIQS, San Juan Capistrano, CA
AdaMAT, Dynamics Research Corporation, Andover, MA
AMS, Rome Air Development Center, Griffiss AFB, NY
Analysis of Complexity Tool (ACT), McCabe & Assocs., Columbia, MD
Battlemap Analysis Tool (BAT), McCabe & Assocs., Columbia, MD
PC-Metric, SET Laboratories, Inc., Mulino, OR
(*** This tool is used internally to support their consulting service.)

REFERENCES: Albrecht and Gaffney 1983; Arthur 1983; Basili 1983; Basili and Hutchens 1983; Basili, Selby, and Phillips 1983; Boehm et al 1978; Bowen, Wigle, and Tsai 1985; Bulut 1974; Carver 1986; Cote et al 1988;

Coulter 1983; Crawford, McIntosh, and Pregibon 1985; Curtis 1980; Elshoff 1976, 1982, 1983; Evangelist 1983; Fitzsimmons 1978; Funami 1976; Gaffney 1981; Gordon 1976, 1979; Halstead 1972, 1973, 1977, 1979; Halstead and Zislis 1973; Harrison 1984; Henry 1979; Henry and Kafura 1984; Henry, Kafura, and Harris 1981; Jones 1978; Kafura and Henry 1982; Kafura and Reddy 1987; Lasky, Kaminsky, and Boaz 1989; Lassez 1981; Li and Cheung 1987; Lind and Vairavan 1989; Mannino, Stoddard, and Sudduth 1990; McCabe 1976, 1982, 1989; McCabe and Butler 1989; McCall, Richards, and Walters 1977; McClure 1978; Millman and Curtis 1980; Murine 1983, 1985a, 1985b, 1986, 1988; Myers 1977; Pierce, Hartley, and Worrells 1987; Prather 1983, 1984; Ramamurthy and Melton 1988; Schneidewind and Hoffmann 1979; Shen, Conte, and Dunsmore 1983; Shneiderman 1980; Siyan 1989; Sunazuka, Azuma, and Yamagishi 1985; Walsh 1979; Ward 1989; Warthman 1987; Weyuker 1988; Zislis 1973; Zweben 1977

## Modifiability

This quality factor addresses the concern that programs be easy to change, regardless of the reason for the change.

CLAIM: Some researchers claim that

> "A software product possesses modifiability to the extent that it facilitates the incorporation of changes, once the nature of the desired change has been determined." (Boehm et al 1978)

DOMAIN: This factor applies to any source code.

NECESSARY CONDITIONS: The criteria for a program change must be defined for the context.

QUALIFICATIONS: The effort expended to understand how to change a program is not addressed by this quality factor. Only the next step of implementing the change is addressed.

CRITICAL ANALYSIS: This quality factor encompasses all the concerns addressed by expandability, flexibility, maintainability, portability, and reusability. This does not mean that they are identical, but rather that modifiability is fundamental to them all. The ease of making all types of changes must be addressed for this quality factor. The other quality factors emphasize a particular type of modifiability.

SQM RELATIONSHIPS: Halstead's Effort; Henry's Information Flow; McCabe's Cyclomatic Complexity

TOOLS: Analysis of Complexity Tool (ACT), McCabe & Assocs., Columbia, MD
Battlemap Analysis Tool (BAT), McCabe & Assocs., Columbia, MD
LOGISCOPE, Verilog USA Inc., Alexandria, VA
PC-Metric, SET Laboratories, Inc., Mulino, OR

REFERENCES: Albrecht and Gaffney 1983; Arthur 1983; Basili 1983; Basili and Hutchens 1983; Basili, Selby, and Phillips 1983; Bulut 1974; Carver 1986; Cote et al 1988; Coulter 1983; Crawford, McIntosh, and Pregibon 1985; Curtis 1980; Elshoff 1976, 1982, 1983; Evangelist 1983; Fitzsimmons 1978; Funami 1976; Gaffney 1981; Gordon 1976, 1979; Halstead 1972, 1973, 1977, 1979; Halstead and Zislis 1973; Harrison 1984; Henry 1979; Henry and Kafura 1984; Henry, Kafura, and Harris 1981; Jones 1978; Kafura and Henry 1982; Kafura and Reddy 1987; Lassez 1981; Li and Cheung 1987; Lind and Vairavan 1989; Mannino, Stoddard, and Sudduth 1990; McCabe 1976, 1982, 1989; McCabe and Butler 1989; McClure 1978; Myers 1977; Prather 1983, 1984; Ramamurthy and Melton

1988; Schneidewind and Hoffmann 1979; Shen, Conte, and Dunsmore 1983; Shneiderman 1980; Siyan 1989; Walsh 1979; Ward 1989; Weyuker 1988; Zislis 1973; Zweben 1977

SOFTWARE QUALITY FACTOR DATA SHEET

## Modularity

This quality factor addresses the concern that programs be composed of many small, simple, independent steps that are clearly delineated by the code.

CLAIM: One researcher claims that software modularity is the

> "Formal way of dividing a program into a number of sub-units each having a well defined function and relationship to the rest of the program" (Mealy as quoted in McCall, Richards, and Walters 1977).
>
> These sub-units are called modules. Thus, the quality factor is modularity. The program itself may be a sub-unit of some larger program.

DOMAIN: This factor applies to any source code.

NECESSARY CONDITIONS: The criteria for distinguishing modules and intermodule relationships must be defined for the context.

QUALIFICATIONS: This is a programmer-oriented quality factor, not a user-oriented one.

CRITICAL ANALYSIS: This quality factor is usually used to indicate the presence of one of the following quality factors: expandability, flexibility, maintainability, modifiability, portability, reusability, and understandability. Depending on which of these quality factors is to be indicated by modularity, different modules and intermodule relationships are identified. For example, a program that is broken into one set of modules to enhance expandability may be broken into a different set of modules to enhance understandability.

SQM RELATIONSHIPS: McCabe's Cyclomatic Complexity and Essential Complexity; RADC's MO.1(3), MO.1(5), MO.1(7)

TOOLS: ***, METRIQS, San Juan Capistrano, CA
AdaMAT, Dynamics Research Corporation, Andover, MA
AMS, Rome Air Development Center, Griffiss AFB, NY
Analysis of Complexity Tool (ACT), McCabe & Assocs., Columbia, MD
Battlemap Analysis Tool (BAT), McCabe & Assocs., Columbia, MD
LOGISCOPE, Verilog USA Inc., Alexandria, VA
PC-Metric, SET Laboratories, Inc., Mulino, OR
(*** This tool is used internally to support their consulting service.)

REFERENCES: Arthur 1983; Basili 1983; Basili and Hutchens 1983; Basili, Selby, and Phillips 1983; Boehm et al 1978; Bowen, Wigle, and Tsai 1985;

Carver 1986; Cote et al 1988; Crawford, McIntosh, and Pregibon 1985; Curtis 1980; Elshoff 1982, 1983; Evangelist 1983; Gaffney 1981; Harrison 1984; Henry, Kafura, and Harris 1981; Kafura and Reddy 1987; Lasky, Kaminsky, and Boaz 1989; Li and Cheung 1987; Lind and Vairavan 1989; Mannino, Stoddard, and Sudduth 1990; McCabe 1976, 1982, 1989; McCabe and Butler 1989; McCall, Richards, and Walters 1977; McClure 1978; Millman and Curtis 1980; Murine 1983, 1985a, 1985b, 1986, 1988; Myers 1977; Pierce, Hartley, and Worrells 1987; Prather 1983, 1984; Ramamurthy and Melton 1988; Schneidewind and Hoffmann 1979; Shneiderman 1980; Siyan 1989; Sunazuka, Azuma, and Yamagishi 1985; Walsh 1979; Ward 1989; Warthman 1987; Weyuker 1988

SOFTWARE QUALITY FACTOR DATA SHEET

## Performance

*This quality factor addresses* the concern of how well a program attribute or function is implemented with respect to some standard. Often, this is related to the utilization of resources.

CLAIM: One researcher claims that software performance is determined by

> "The effectiveness with which resources of the host system are utilized toward meeting the objective of the software system" (Dennis as quoted in McCall, Richards, and Walters 1977).

DOMAIN: This factor only applies to source code for which performance standards have been set.

NECESSARY CONDITIONS: The standard against which the implementation is to be compared must be defined for the context.

QUALIFICATIONS: None.

CRITICAL ANALYSIS: This quality factor must not be confused with completeness or correctness. Performance is concerned with how well the job is done, given that a program completely and correctly meets its specifications. The performance quality factor is used to determine which satisfactory program performs better.

SQM RELATIONSHIPS: Albrecht's Function Points; Ejiogu's Structural Complexity; Halstead's Length, Volume, and Effort; Henry's Information Flow; McCabe's Cyclomatic Complexity and Essential Complexity

TOOLS: Analysis of Complexity Tool (ACT), McCabe & Assocs., Columbia, MD
Battlemap Analysis Tool (BAT), McCabe & Assocs., Columbia, MD
Checkpoint, Software Productivity Research, Inc., Burlington, MA
*COMPLEXIMETER, Softmetrix, Inc., Chicago*, IL
LOGISCOPE, Verilog USA Inc., Alexandria, VA
PC-Metric, SET Laboratories, Inc., Mulino, OR
SIZE PLANNER, Quantitative Software Management, Inc., McClean, VA

REFERENCES: Albrecht 1979, 1985; Albrecht and Gaffney 1983; Arthur 1983; Basili 1983; Basili and Hutchens 1983; Basili, Selby, and Phillips 1983; Behrens 1983; Bulut 1974; Carver 1986; Cote et al 1988; Coulter 1983; Crawford, McIntosh, and Pregibon 1985; Curtis 1980; Drummond 1985; Ejiogu 1984a, 1984b, 1987, 1988, 1990; Elshoff 1976, 1982, 1983; Evangelist 1983; Fitzsimmons 1978; Funami 1976; Gaffney 1981; Gordon 1976, 1979; Halstead 1972, 1973, 1977, 1979; Halstead and Zislis 1973; Harrison 1984; Henry 1979; Henry and Kafura 1984; Henry, Kafura, and Harris 1981; Jones 1978, 1988; Kafura and Henry 1982; Kafura and Reddy 1987; Lassez 1981; Li and Cheung 1987; Lind and Vairavan 1989; Low and

Jeffrey 1990; Mannino, Stoddard, and Sudduth 1990; McCabe 1976, 1982, 1989; McCabe and Butler 1989; McClure 1978; Myers 1977; Prather 1983, 1984; Ramamurthy and Melton 1988; Schneidewind and Hoffmann 1979; Shen, Conte, and Dunsmore 1983; Shneiderman 1980; Siyan 1989; Walsh 1979; Ward 1989; Weyuker 1988; Zislis 1973; Zweben 1977

SOFTWARE QUALITY FACTOR DATA SHEET

## Portability

This quality factor addresses the concern that programs be changed easily to operate on a different set of equipment.

CLAIM: One researcher claims that software portability is determined by

> "How quickly and cheaply the software system can be converted to perform the same functions using different equipment" (Kosy as quoted in McCall, Richards, and Walters 1977).

DOMAIN: This factor only applies to source code that performs functions that are supported on other equipment.

NECESSARY CONDITIONS: The criteria for distinguishing lack of modifiability from lack of portability must be defined for the context.

QUALIFICATIONS: The portability of the code, not the function that it performs, is addressed by this quality factor. If the hardware on which it operates is unique, there is no machine to which the code can be transported to perform the same function. This is not the fault of the code.

Every change required to operate a program on different equipment is weighted by the modifiability of the code. If the modifiability is poor, a very machine-independent program may still appear to be not very portable.

CRITICAL ANALYSIS: This quality factor must not be confused with reusability. Portability addresses only those changes required to make the program work on a different machine; everything else remains the same.

SQM RELATIONSHIPS: RADC's MO.1(3), MO.1(5), MO.1(7), SD.2(1), SD.2(3)

TOOLS: ***, METRIQS, San Juan Capistrano, CA
AdaMAT, Dynamics Research Corporation, Andover, MA
AMS, Rome Air Development Center, Griffiss AFB, NY
(*** This tool is used internally to support their consulting service.)

REFERENCES: Boehm et al 1978; Bowen, Wigle, and Tsai 1985; Lasky, Kaminsky, and Boaz 1989; McCall, Richards, and Walters 1977; Millman and Curtis 1980; Murine 1983, 1985a, 1985b, 1986, 1988; Pierce, Hartley, and Worrells 1987; Shneiderman 1980; Suñazuka, Azuma, and Yamagishi 1985; Wartham 1987

SOFTWARE QUALITY FACTOR DATA SHEET

## Reliability

This quality factor addresses the concern that programs continue to perform properly over time.

CLAIM: One researcher claims that software reliability is determined by

> "The probability that a software system will operate without failure for at least a given period of time when used under stated conditions" (Kosy as quoted in McCall, Richards, and Walters 1977).

DOMAIN: This factor only applies to source code for which the proper operating conditions are specified.

NECESSARY CONDITIONS: The proper operating conditions and the criteria for establishing a failure must be defined for the context.

QUALIFICATIONS: For code-based SQM, this quality factor must be given by probabilities based solely on program code. Most reliability estimates are based on experimental data.

CRITICAL ANALYSIS: This quality factor must not be confused with accuracy or correctness. Reliability addresses the continuance of functions already presumed to be both accurate and correct.

SQM RELATIONSHIPS: Halstead's Effort; Henry's Information Flow; McCabe's Cyclomatic Complexity and Essential Complexity, RADC's SI.3(1)

TOOLS: ***, METRIQS, San Juan Capistrano, CA
AdaMAT, Dynamics Research Corporation, Andover, MA
AMS, Rome Air Development Center, Griffiss AFB, NY
Analysis of Complexity Tool (ACT), McCabe & Assocs., Columbia, MD
Battlemap Analysis Tool (BAT), McCabe & Assocs., Columbia, MD
LOGISCOPE, Verilog USA Inc., Alexandria, VA
PC-Metric, SET Laboratories, Inc., Mulino, OR
(*** This tool is used internally to support their consulting service.)

REFERENCES: Albrecht and Gaffney 1983; Arthur 1983; Basili 1983; Basili and Hutchens 1983; Basili, Selby, and Phillips 1983; Boehm et al 1978; Bowen, Wigle, and Tsai 1985; Bulut 1974; Carver 1986; Cote et al 1988; Coulter 1983; Crawford, McIntosh, and Pregibon 1985; Curtis 1980; Elshoff 1976, 1982, 1983; Evangelist 1983; Fitzsimmons 1978; Funami 1976; Gaffney 1981; Gordon 1976, 1979; Halstead 1972, 1973, 1977, 1979; Halstead and Zislis 1973; Harrison 1984; Henry 1979; Henry and Kafura 1984; Henry, Kafura, and Harris 1981; Jones 1978; Kafura and Henry 1982; Kafura and Reddy 1987; Lasky, Kaminsky, and Boaz 1989;

Lassez 1981; Li and Cheung 1987; Lind and Vairavan 1989; Mannino, Stoddard, and Sudduth 1990; McCabe 1976, 1982, 1989; McCabe and Butler 1989; McCall, Richards, and Walters 1977; McClure 1978; Millman and Curtis 1980; Murine 1983, 1985a, 1985b, 1986, 1988; Myers 1977; Pierce, Hartley, and Worrells 1987; Prather 1983, 1984; Ramamurthy and Melton 1988; Schneidewind and Hoffmann 1979; Shen, Conte, and Dunsmore 1983; Shneiderman 1980; Siyan 1989; Sunazuka, Azuma, and Yamagishi 1985; Walsh 1979; Ward 1989; Warthman 1987; Weyuker 1988; Zislis 1973; Zweben 1977

SOFTWARE QUALITY FACTOR DATA SHEET

## Reusability

This quality factor addresses the concern that programs be easy to reuse in a different application.

CLAIM: Some researchers claim that software reusability is determined by the

"Relative effort to convert a software component for use in a different application" (Bowen, Wigle, and Tsai 1985).

DOMAIN: This factor applies to any source code.

NECESSARY CONDITIONS: The proper operating conditions and the criteria for establishing a failure must be defined for the context.

QUALIFICATIONS: Every change required to use a program in a different application is weighted by the modifiability of the code. If the modifiability is poor, a very application-independent program may still appear to be not very reusable.

This is a programmer-oriented quality factor, not a user-oriented one.

CRITICAL ANALYSIS: This quality factor must not be confused with expandability, flexibility, or portability. All three address ease of changing a program, but for reasons other than using the program to perform the same function in a different application.

SQM RELATIONSHIPS: RADC's AP.3(2), FS.1(2), GE.1(1), GE.2(2), MO.1(3), MO.1(5), MO.1(7), SD.2(1), SD.2(3), SI.3(1), ST.5(2)

TOOLS: ***, METRIQS, San Juan Capistrano, CA
AdaMAT, Dynamics Research Corporation, Andover, MA
AMS, Rome Air Development Center, Griffiss AFB, NY
(*** This tool is used internally to support their consulting service.)

REFERENCES: Boehm et al 1978; Bowen, Wigle, and Tsai 1985; Lasky, Kaminsky, and Boaz 1989; McCall, Richards, and Walters 1977; Millman and Curtis 1980; Murine 1983, 1985a, 1985b, 1986, 1988; Pierce, Hartley, and Worrells 1987; Shneiderman 1980; Sunazuka, Azuma, and Yamagishi 1985; Wartham 1987

## Simplicity

This quality factor addresses the concern that, as much as possible, programs be implemented in strictly sequential steps that depend only on the step before it. Comments should exhibit a similar straight forwardness.

CLAIM: Some researchers claim that software simplicity is determined by

> "Those characteristics of software which provide for definition and implementation of functions in the most noncomplex and understandable manner" (Bowen, Wigle, and Tsai 1985).
>
> Simplicity is the opposite of the quality factor, complexity.

DOMAIN: This factor applies to any source code.

NECESSARY CONDITIONS: The criteria for determining simplicity must be defined for the context.

QUALIFICATIONS: Simplicity of implementation must be evaluated independently of the simplicity or complexity of the algorithm. Simplicity addresses how simply an algorithm of a certain complexity is implemented.

> This is a programmer-oriented quality factor, not a user-oriented one.

CRITICAL ANALYSIS: This quality factor is usually represented by other quality factors. Since simplicity is desired because it enhances understanding, clarity and understandability are more commonly used.

SQM RELATIONSHIPS: Albrecht's Function Points; Ejiogu's Structural Complexity; Halstead's Length, Volume, and Effort; Henry's Information Flow; McCabe's Cyclomatic Complexity and Essential Complexity; RADC's SI.3(1)

TOOLS: ***, METRIQS, San Juan Capistrano, CA
AdaMAT, Dynamics Research Corporation, Andover, MA
AMS, Rome Air Development Center, Griffiss AFB, NY
Analysis of Complexity Tool (ACT), McCabe & Assocs., Columbia, MD
Battlemap Analysis Tool (BAT), McCabe & Assocs., Columbia, MD
Checkpoint, Software Productivity Research, Inc., Burlington, MA
COMPLEXIMETER, Softmetrix, Inc., Chicago, IL
LOGISCOPE, Verilog USA Inc., Alexandria, VA
PC-Metric, SET Laboratories, Inc., Mulino, OR
SIZE PLANNER, Quantitative Software Management, Inc., McClean, VA
(*** This tool is used internally to support their consulting service.)

REFERENCES: Albrecht 1979, 1985; Albrecht and Gaffney 1983; Arthur 1983; Basili 1983; Basili and Hutchens 1983; Basili, Selby, and Phillips 1983; Behrens 1983; Boehm et al 1978; Bowen, Wigle, and Tsai 1985; Bulut 1974; Carver 1986; Cote et al 1988; Coulter 1983; Crawford, McIntosh, and Pregibon 1985; Curtis 1980; Drummond 1985; Ejiogu 1984a, 1984b, 1987, 1988, 1990; Elshoff 1976, 1982, 1983; Evangelist 1983; Fitzsimmons 1978; Funami 1976; Gaffney 1981; Gordon 1976, 1979; Halstead 1972, 1973, 1977, 1979; Halstead and Zislis 1973; Harrison 1984; Henry 1979; Henry and Kafura 1984; Henry, Kafura, and Harris 1981; Jones 1978, 1988; Kafura and Henry 1982; Kafura and Reddy 1987; Lasky, Kaminsky, and Boaz 1989; Lassez 1981; Li and Cheung 1987; Lind and Vairavan 1989; Low and Jeffrey 1990; Mannino, Stoddard, and Sudduth 1990; McCabe 1976, 1982, 1989; McCabe and Butler 1989; McCall, Richards, and Walters 1977; McClure 1978; Millman and Curtis 1980; Murine 1983, 1985a, 1985b, 1986, 1988; Myers 1977; Pierce, Hartley, and Worrells 1987; Prather 1983, 1984; Ramamurthy and Melton 1988; Schneidewind and Hoffmann 1979; Shen, Conte, and Dunsmore 1983; Shneiderman 1980; Siyan 1989; Sunazuka, Azuma, and Yamagishi 1985; Walsh 1979; Ward 1989; Warthman 1987; Weyuker 1988; Zislis 1973; Zweben 1977

SOFTWARE QUALITY FACTOR DATA SHEET

## Testability

This quality factor addresses the concern that programs be easy to test.

CLAIM: Some researchers claim that

> "A software product possesses the characteristic Testability to the extent that it facilitates the establishment of acceptance criteria and supports evaluation of its performance." (Boehm et al 1978).

DOMAIN: This factor only applies to source code for which acceptance criteria can been established.

NECESSARY CONDITIONS: The test plan must be defined for the context.

QUALIFICATIONS: This is a programmer-oriented quality factor, not a user-oriented one.

CRITICAL ANALYSIS: This quality factor must not be confused with maintainability. Testability is only concerned with finding problems, not fixing them.

SQM RELATIONSHIPS: McCabe's Cyclomatic Complexity

TOOLS: Analysis of Complexity Tool (ACT), McCabe & Assocs., Columbia, MD.
Battlemap Analysis Tool (BAT), McCabe & Assocs., Columbia, MD
LOGISCOPE, Verilog USA Inc., Alexandria, VA
PC-Metric, SET Laboratories, Inc., Mulino, OR

REFERENCES: Arthur 1983; Basili 1983; Basili and Hutchens 1983; Basili, Selby and Phillips 1983; Carver 1986; Cote et al 1988; Crawford, McIntosh, and Pregibon 1985; Curtis 1980; Elshoff 1982, 1983; Evangelist 1983; Gaffney 1981; Harrison 1984; Henry, Kafura, and Harris 1981; Kafura and Reddy 1987; Li and Cheung 1987; Lind and Vairavan 1989; Mannino, Stoddard, and Sudduth 1990; McCabe 1976, 1982, 1989; McCabe and Butler 1989; McClure 1978; Myers 1977; Prather 1983, 1984; Ramamurthy and Melton 1988; Schneidewind and Hoffmann 1979; Shneiderman 1980; Siyan 1989; Walsh 1979; Ward 1989; Weyuker 1988

SOFTWARE QUALITY FACTOR DATA SHEET

## Understandability

This quality factor addresses the concern that programs be easy to understand.

CLAIM: One researcher claims that software understandability is determined by the

"Ease with which the implementation can be understood" (Richards as quoted in McCall, Richards, and Walters 1977).

DOMAIN: This factor applies to any source code.

NECESSARY CONDITIONS: The criteria for establishing whether a program is understood by a program reader must be defined for the context.

QUALIFICATIONS: This is a programmer-oriented quality factor, not a user-oriented one.

CRITICAL ANALYSIS: This quality factor must not be confused with modifiability. Before any change can be made to a program, the code must be understood as implemented. The next step may be to modify it.

SQM RELATIONSHIPS: Ejiogu's Structural Complexity; Halstead's Effort; Henry's Information Flow; McCabe's Cyclomatic Complexity and Essential Complexity

TOOLS: Analysis of Complexity Tool (ACT), McCabe & Assocs., Columbia, MD
Battlemap Analysis Tool (BAT), McCabe & Assocs., Columbia, MD
COMPLEXIMETER, Softmetrix, Inc., Chicago, IL
PC-Metric, SET Laboratories, Inc., Mulino, OR

REFERENCES: Albrecht and Gaffney 1983; Arthur 1983; Basili 1983; Basili and Hutchens 1983; Basili, Selby, and Phillips 1983; Bulut 1974; Carver 1986; Cote et al 1988; Coulter 1983; Crawford, McIntosh, and Pregibon 1985; Curtis 1980; Ejiogu 1984a, 1984b, 1987, 1988, 1990; Elshoff 1976, 1982, 1983; Evangelist 1983; Fitzsimmons 1978; Funami 1976; Gaffney 1981; Gordon 1976, 1979; Halstead 1972, 1973, 1977, 1979; Halstead and Zislis 1973; Harrison 1984; Henry 1979; Henry and Kafura 1984; Henry, Kafura, and Harris 1981; Jones 1978; Kafura and Henry 1982; Kafura and Reddy 1987; Lassez 1981; Li and Cheung 1987; Lind and Vairavan 1989; Mannino, Stoddard, and Sudduth 1990; McCabe 1976, 1982, 1989; McCabe and Butler 1989; McClure 1978; Myers 1977; Prather 1983, 1984; Ramamurthy and Melton 1988; Schneidewind and Hoffmann 1979; Shen, Conte, and Dunsmore 1983; Shneiderman 1980; Siyan 1989; Walsh 1979; Ward 1989; Weyuker 1988; Zislis 1973; Zweben 1977

SOFTWARE QUALITY FACTOR DATA SHEET

## Usability

This quality factor addresses the concern that programs be easy to use.

CLAIM: Some researchers claim that

> "A software product possesses the characteristic Usability to the extent that it is convenient and practicable to use." (Boehm et al 1978).

DOMAIN: This factor applies to any source code.

NECESSARY CONDITIONS: The criteria for establishing that a program is easy to use must be defined for the context.

QUALIFICATIONS: To keep this quality factor independent of maintainability, portability, and reusability, this quality factor must address the usability of an error-free program that is run on the intended machine, in the intended context or application.

CRITICAL ANALYSIS: This quality factor must not be confused with programmer concerns, like understandability. Usability is a strictly user-oriented quality factor.

SQM RELATIONSHIPS: RADC's software metrics

TOOLS: ***, METRIQS, San Juan Capistrano, CA
AdaMAT, Dynamics Research Corporation, Andover, MA
AMS, Rome Air Development Center, Griffiss AFB, NY
(*** This tool is used internally to support their consulting service.)

REFERENCES: Boehm et al 1978; Bowen, Wigle, and Tsai 1985; Lasky, Kaminsky, and Boaz 1989; McCall, Richards, and Walters 1977; Millman and Curtis 1980; Murine 1983, 1985a, 1985b, 1986, 1988; Pierce, Hartley, and Worrells 1987; Shneiderman 1980; Sunazuka, Azuma, and Yamagishi 1985; Wartham 1987

## APPENDIX C - SOFTWARE METRIC DATA

This appendix contains the software metric data from which the software metrics and Software Quality Metrics (SQM) of the worked examples were computed. Halstead's and McCabe's software metrics are as computed by PC-Metric, Release 2.0, of the Assembler version. Rome Air Development Center's (RADC) software metrics and SQM were computed by the authors' own text analyzer. Values that tripped the thresholds for particular SQM in the worked examples are marked with an asterisk (*).

The code analyzed was supplied by a major avionics manufacturer. It is a modular, real-time, flight control application written in Assembly language. It is a standalone, memory-resident application. It does not rely on an operating system. The Assembly language uses simple constructs and does not rely on any library functions. Diagnostics and self-checks are included.

The code consists of 143 files of subroutines, averaging about 164 lines of executable code per file. Many of the files contain additional internal and external subroutines. The files were categorized by the developer into 84 nonessential files and 59 essential files. Each file is documented with a header of information of consistent format. The code is well commented throughout. The majority of the files are fairly simple, but many are very complex. The files of the essential code are about 10 percent larger and more complex than the nonessential code.

Two of the essential modules were copied, renamed, and arbitrarily categorized as critical modules, for the sake of providing a more complete example.

The 84 modules of nonessential code had the Halstead and McCabe measures shown in the following table. Two modules are not listed because they contained no executable statements. The total scores are the measures of the code when the modules are treated as one large module.

| | $\eta_1$ | $\eta_2$ | $N_1$ | $N_2$ | N | $\hat{N}$ | V | E | v(G) | Li | St | $\hat{L}$ | I | $\hat{B}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| NONESS01 | 35 | 98 | 573 | 447 | 1020 | 828 | 7196* | 574426 | 30* | 355* | 275 | 0.013 | 90.2 | 2.2 |
| NONESS02 | 21 | 39 | 135 | 116 | 251 | 298 | 1483 | 46304 | 13 | 87 | 65 | 0.032 | 47.5 | 0.5 |
| NONESS03 | 23 | 29 | 187 | 154 | 341 | 245 | 1944 | 118709 | 7 | 111 | 100 | 0.016 | 31.8 | 0.6 |
| NONESS04 | 27 | 30 | 129 | 92 | 221 | 276 | 1289 | 53367 | 8 | 72 | 61 | 0.024 | 31.1 | 0.4 |
| NONESS05 | 29 | 36 | 179 | 138 | 317 | 327 | 1909 | 106114 | 12 | 111 | 86 | 0.018 | 34.3 | 0.6 |
| NONESS06 | 12 | 17 | 61 | 48 | 109 | 113 | 530 | 8971 | 1 | 37 | 28 | 0.059 | 31.3 | 0.2 |
| NONESS07 | 15 | 45 | 211 | 157 | 368 | 306 | 2174 | 56879 | 18 | 121 | 90 | 0.038 | 83.1 | 0.7 |
| NONESS08 | 11 | 10 | 20 | 13 | 33 | 71 | 145 | 1036 | 1 | 13 | 8 | 0.140 | 20.3 | 0.0 |
| NONESS09 | 13 | 18 | 49 | 50 | 99 | 123 | 490 | 8856 | 4 | 41 | 29 | 0.055 | 27.2 | 0.2 |
| NONESS10 | 13 | 16 | 47 | 48 | 95 | 112 | 462 | 8999 | 5 | 41 | 29 | 0.051 | 23.7 | 0.1 |

| | $\eta_1$ | $\eta_2$ | $N_1$ | $N_2$ | N | $\hat{N}$ | V | E | v(G) | Li | St | $\hat{L}$ | I | $\hat{B}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| NONESS11 | 23 | 72 | 439 | 318 | 757 | 548 | 4973* | 252606 | 19 | 248* | 195 | 0.020 | 97.9 | 1.6 |
| NONESS12 | 27 | 87 | 528 | 368 | 896 | 689 | 6122* | 349603 | 29* | 318* | 246 | 0.018 | 107.2 | 1.9 |
| NONESS13 | 21 | 94 | 366 | 270 | 636 | 708 | 4354* | 131307 | 11 | 189 | 156 | 0.033 | 144.4 | 1.4 |
| NONESS14 | 15 | 20 | 77 | 57 | 134 | 145 | 687 | 14692 | 1 | 52 | 38 | 0.047 | 32.2 | 0.2 |
| NONESS15 | 25 | 29 | 120 | 90 | 210 | 257 | 1209 | 46882 | 9 | 70 | 55 | 0.026 | 31.2 | 0.4 |
| NONESS16 | 11 | 13 | 45 | 33 | 78 | 86 | 358 | 4993 | 1 | 57 | 27 | 0.072 | 25.6 | 0.1 |
| NONESS17 | 15 | 11 | 33 | 32 | 65 | 97 | 306 | 6666 | 5 | 33 | 23 | 0.046 | 14.0 | 0.1 |
| NONESS18 | 26 | 53 | 204 | 167 | 371 | 426 | 2339 | 95799 | 14 | 124 | 99 | 0.024 | 57.1 | 0.7 |
| NONESS19 | 23 | 62 | 294 | 242 | 536 | 473 | 3435 | 154207 | 15 | 191 | 139 | 0.022 | 76.5 | 1.1 |
| NONESS20 | 27 | 86 | 531 | 403 | 934 | 681 | 6370* | 402980 | 50* | 323* | 235 | 0.016 | 100.7 | 2.0 |
| NONESS21 | 27 | 36 | 158 | 124 | 282 | 314 | 1686 | 78380 | 8 | 91 | 76 | 0.022 | 36.2 | 0.5 |
| NONESS22 | 15 | 16 | 53 | 40 | 93 | 123 | 461 | 8639 | 3 | 35 | 26 | 0.053 | 24.6 | 0.1 |
| NONESS23 | 21 | 43 | 238 | 196 | 434 | 326 | 2604 | 124629 | 16 | 128 | 110 | 0.021 | 54.4 | 0.8 |
| NONESS24 | 22 | 32 | 118 | 95 | 213 | 258 | 1226 | 40030 | 7 | 78 | 55 | 0.031 | 37.5 | 0.4 |
| NONESS25 | 40 | 55 | 276 | 221 | 497 | 531 | 3265 | 262405 | 21 | 173 | 141 | 0.012 | 40.6 | 1.0 |
| NONESS26 | 9 | 10 | 30 | 20 | 50 | 62 | 212 | 1912 | 1 | 19 | 13 | 0.111 | 23.6 | 0.1 |
| NONESS27 | 43 | 61 | 391 | 304 | 695 | 595 | 4657* | 498965 | 24* | 224* | 191 | 0.009 | 43.5 | 1.5 |
| NONESS28 | 25 | 77 | 385 | 333 | 718 | 599 | 4791* | 258983 | 36* | 230* | 183 | 0.018 | 88.6 | 1.5 |
| NONESS29 | 15 | 17 | 65 | 48 | 113 | 128 | 565 | 11965 | 1 | 41 | 32 | 0.047 | 26.7 | 0.2 |
| NONESS30 | 16 | 1 | 29 | 1 | 30 | 64 | 123 | 981 | 1 | 19 | 15 | 0.125 | 15.3 | 0.0 |
| NONESS31 | 12 | 5 | 18 | 7 | 25 | 55 | 102 | 858 | 2 | 16 | 9 | 0.119 | 12.2 | 0.0 |
| NONESS32 | 11 | 28 | 120 | 106 | 226 | 173 | 1195 | 24871 | 14 | 95 | 59 | 0.048 | 57.4 | 0.4 |
| NONESS33 | 18 | 26 | 78 | 57 | 135 | 197 | 737 | 14542 | 3 | 53 | 35 | 0.051 | 37.4 | 0.2 |
| NONESS34 | 12 | 5 | 18 | 7 | 25 | 55 | 102 | 858 | 2 | 14 | 9 | 0.119 | 12.2 | 0.0 |
| NONESS35 | 18 | 33 | 170 | 136 | 306 | 242 | 1736 | 64381 | 3 | 103 | 83 | 0.027 | 46.8 | 0.5 |
| NONESS36 | 15 | 18 | 68 | 56 | 124 | 134 | 626 | 14595 | 5 | 55 | 34 | 0.043 | 26.8 | 0.2 |
| NONESS37 | 18 | 26 | 87 | 67 | 154 | 197 | 841 | 19499 | 9 | 58 | 39 | 0.043 | 36.3 | 0.3 |
| NONESS38 | 7 | 6 | 17 | 13 | 30 | 35 | 111 | 842 | 1 | 13 | 8 | 0.132 | 14.6 | 0.0 |
| NONESS39 | 14 | 22 | 64 | 49 | 113 | 151 | 584 | 9108 | 5 | 35 | 28 | 0.064 | 37.5 | 0.2 |
| NONESS40 | 15 | 6 | 29 | 9 | 38 | 74 | 167 | 1878 | 3 | 19 | 13 | 0.089 | 14.8 | 0.1 |
| NONESS41 | 26 | 69 | 299 | 234 | 533 | 544 | 3502 | 154381 | 18 | 195 | 144 | 0.023 | 79.4 | 1.1 |
| NONESS42 | 28 | 35 | 204 | 123 | 327 | 314 | 1955 | 96165 | 11 | 108 | 92 | 0.020 | 39.7 | 0.6 |
| NONESS43 | 31 | 18 | 98 | 42 | 140 | 229 | 786 | 28429 | 6 | 65 | 57 | 0.028 | 21.7 | 0.2 |
| NONESS44 | 14 | 39 | 113 | 86 | 199 | 259 | 1140 | 17595 | 3 | 52 | 45 | 0.065 | 73.8 | 0.4 |
| NONESS45 | 22 | 30 | 110 | 81 | 191 | 245 | 1089 | 32337 | 12 | 66 | 49 | 0.034 | 36.7 | 0.3 |
| NONESS46 | 27 | 38 | 189 | 151 | 340 | 328 | 2048 | 109843 | 19 | 115 | 89 | 0.019 | 38.2 | 0.6 |
| NONESS47 | 20 | 15 | 72 | 41 | 113 | 145 | 580 | 15843 | 10 | 46 | 33 | 0.037 | 21.2 | 0.2 |
| NONESS48 | 5 | 1 | 7 | 1 | 8 | 12 | 21 | 52 | 1 | 8 | 4 | 0.400 | 8.3 | 0.0 |
| NONESS49 | 18 | 39 | 132 | 104 | 236 | 281 | 1377 | 33037 | 7 | 90 | 60 | 0.042 | 57.4 | 0.4 |
| NONESS50 | 29 | 86 | 498 | 388 | 886 | 694 | 6065* | 396771 | 24* | 313* | 234 | 0.015 | 92.7 | 1.9 |

| | $\eta_1$ | $\eta_2$ | $N_1$ | $N_2$ | N | $\hat{N}$ | V | E | v(G) | Li | St | $\hat{L}$ | I | $\hat{B}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| NONESS51 | 30 | 102 | 528 | 419 | 947 | 828 | 6671* | 411054 | 53* | 318* | 232 | 0.016 | 108.3 | 2.1 |
| NONESS52 | 11 | 26 | 106 | 82 | 188 | 160 | 979 | 16988 | 4 | 54 | 44 | 0.058 | 56.5 | 0.3 |
| NONESS53 | 5 | 7 | 14 | 11 | 25 | 31 | 90 | 352 | 1 | 11 | 6 | 0.255 | 22.8 | 0.0 |
| NONESS54 | 34 | 55 | 274 | 217 | 491 | 491 | 3180 | 213263 | 14 | 184 | 134 | 0.015 | 47.4 | 1.0 |
| NONESS55 | 28 | 60 | 328 | 251 | 579 | 489 | 3740 | 219040 | 15 | 205 | 163 | 0.017 | 63.9 | 1.2 |
| NONESS56 | 34 | 96 | 564 | 416 | 980 | 805 | 6882* | 506968 | 34* | 330* | 258 | 0.014 | 93.4 | 2.2 |
| NONESS57 | 27 | 54 | 382 | 270 | 652 | 439 | 4134* | 279017 | 4 | 239* | 192 | 0.015 | 61.2 | 1.3 |
| NONESS58 | 12 | 31 | 153 | 109 | 262 | 197 | 1422 | 29993 | 6 | 76 | 64 | 0.047 | 67.4 | 0.4 |
| NONESS59 | 26 | 34 | 200 | 169 | 369 | 295 | 2180 | 140843 | 6 | 249* | 101 | 0.015 | 33.7 | 0.7 |
| NONESS60 | 26 | 36 | 156 | 116 | 272 | 308 | 1620 | 67841 | 13 | 99 | 75 | 0.024 | 38.7 | 0.5 |
| NONESS61 | 26 | 56 | 186 | 168 | 354 | 447 | 2251 | 87772 | 12 | 121 | 95 | 0.026 | 57.7 | 0.7 |
| NONESS62 | 18 | 29 | 89 | 75 | 164 | 216 | 911 | 21203 | 4 | 59 | 45 | 0.043 | 39.1 | 0.3 |
| NONESS63 | 35 | 68 | 375 | 306 | 681 | 593 | 4554* | 358589 | 24* | 281* | 182 | 0.013 | 57.8 | 1.4 |
| NONESS64 | 30 | 60 | 289 | 232 | 521 | 502 | 3382 | 196171 | 21 | 167 | 133 | 0.017 | 58.3 | 1.1 |
| NONESS65 | 8 | 15 | 47 | 36 | 83 | 83 | 375 | 3604 | 2 | 28 | 20 | 0.104 | 39.1 | 0.1 |
| NONESS66 | 8 | 16 | 49 | 37 | 86 | 88 | 394 | 3647 | 2 | 31 | 23 | 0.108 | 42.6 | 0.1 |
| NONESS67 | 10 | 19 | 79 | 60 | 139 | 114 | 675 | 10662 | 4 | 48 | 34 | 0.063 | 42.8 | 0.2 |
| NONESS68 | 34 | 54 | 275 | 217 | 492 | 484 | 3178 | 217107 | 14 | 183 | 135 | 0.015 | 46.5 | 1.0 |
| NONESS69 | 27 | 46 | 214 | 174 | 388 | 382 | 2402 | 122641 | 9 | 140 | 102 | 0.020 | 47.0 | 0.8 |
| NONESS70 | 37 | 94 | 585 | 439 | 1024 | 809 | 7202* | 622265 | 33* | 329* | 260 | 0.012 | 83.4 | 2.3 |
| NONESS71 | 22 | 42 | 253 | 171 | 424 | 325 | 2544 | 113935 | 2 | 161 | 123 | 0.022 | 56.8 | 0.8 |
| NONESS72 | 33 | 45 | 156 | 104 | 260 | 414 | 1634 | 62318 | 7 | 99 | 74 | 0.026 | 42.9 | 0.5 |
| NONESS73 | 26 | 38 | 154 | 116 | 270 | 322 | 1620 | 64288 | 13 | 97 | 73 | 0.025 | 40.8 | 0.5 |
| NONESS74 | 23 | 27 | 97 | 75 | 172 | 232 | 971 | 31010 | 4 | 66 | 49 | 0.031 | 30.4 | 0.3 |
| NONESS75 | 14 | 12 | 38 | 31 | 69 | 96 | 324 | 5865 | 4 | 33 | 19 | 0.055 | 17.9 | 0.1 |
| NONESS76 | 7 | 12 | 25 | 22 | 47 | 63 | 200 | 1281 | 1 | 20 | 12 | 0.156 | 31.1 | 0.1 |
| NONESS77 | 16 | 41 | 186 | 148 | 334 | 284 | 1948 | 56260 | 15 | 113 | 90 | 0.035 | 67.5 | 0.6 |
| NONESS78 | 12 | 14 | 45 | 30 | 75 | 96 | 353 | 4533 | 2 | 26 | 20 | 0.078 | 27.4 | 0.1 |
| NONESS79 | 34 | 79 | 525 | 392 | 917 | 671 | 6254* | 527561 | 36* | 365* | 254 | 0.012 | 74.1 | 2.0 |
| NONESS80 | 32 | 65 | 414 | 307 | 721 | 551 | 4759* | 359599 | 12 | 274* | 194 | 0.013 | 63.0 | 1.5 |
| NONESS81 | 1 | 2 | 1 | 2 | 3 | 2 | 5 | 2 | 1 | 188 | 1 | 2.000 | 9.5 | 0.0 |
| NONESS82 | 5 | 1 | 7 | 2 | 9 | 12 | 23 | 116 | 1 | 8 | 4 | 0.200 | 4.7 | 0.0 |
| Average | 21 | 38 | 185 | 141 | 326 | 305 | 2061 | 116121 | 11 | 119 | 87 | 0.075 | 46.8 | 0.6 |
| Std Dev | 9 | 26 | 161 | 123 | 284 | 222 | 1994 | 154596 | 11 | 98 | 74 | 0.222 | 27.1 | 0.6 |

Total Scores

| $\eta_1$ | $\eta_2$ | $N_1$ | $N_2$ | v(G) | St | LOC |
|---|---|---|---|---|---|---|
| 129 | 1418 | 15141 | 11602 | 821 | 7161 | 15561 |

The 59 modules of essential code had the Halstead and McCabe measures shown in the following table. Two modules are not listed because they contained no executable statements. The total scores are the measures of the code when the modules are treated as one large module.

| | $\eta_1$ | $\eta_2$ | $N_1$ | $N_2$ | N | $\hat{N}$ | V | E | v(G) | Li | St | $\hat{L}$ | I | $\hat{B}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ESSENT01 | 19 | 22 | 79 | 62 | 141 | 179 | 755 | 20225 | 6 | 47 | 37 | 0.037 | 28.2 | 0.2 |
| ESSENT02 | 33 | 54 | 264 | 209 | 473 | 477 | 3048 | 194618 | 21 | 171 | 122 | 0.016 | 47.7 | 1.0 |
| ESSENT03 | 27 | 21 | 103 | 69 | 172 | 221 | 961 | 42610 | 6 | 64 | 55 | 0.023 | 21.7 | 0.3 |
| ESSENT04 | 23 | 19 | 78 | 62 | 140 | 185 | 755 | 28330 | 4 | 98 | 39 | 0.027 | 20.1 | 0.2 |
| ESSENT05 | 9 | 9 | 23 | 18 | 41 | 57 | 171 | 1539 | 4 | 21 | 17 | 0.111 | 19.0 | 0.1 |
| ESSENT06 | 8 | 8 | 17 | 15 | 32 | 48 | 128 | 960 | 4 | 18 | 12 | 0.133* | 17.1 | 0.0 |
| ESSENT07 | 6 | 2 | 9 | 3 | 12 | 18 | 36 | 162 | 2 | 12 | 8 | 0.222* | 8.0 | 0.0 |
| ESSENT08 | 22 | 31 | 148 | 116 | 264 | 252 | 1512 | 62243 | 12 | 97 | 70 | 0.024 | 36.7 | 0.5 |
| ESSENT09 | 17 | 29 | 109 | 89 | 198 | 210 | 1094 | 28530 | 5 | 83 | 49 | 0.038 | 41.9 | 0.3 |
| ESSENT10 | 19 | 85 | 318 | 224 | 542 | 626 | 3632 | 90919 | 3 | 200 | 143 | 0.040 | 145.1 | 1.1 |
| ESSENT11 | 25 | 65 | 257 | 165 | 422 | 508 | 2740 | 86928 | 4 | 155 | 115 | 0.032 | 86.3 | 0.9 |
| ESSENT12 | 15 | 40 | 155 | 128 | 283 | 271 | 1636 | 39267 | 17 | 97 | 69 | 0.042 | 68.2 | 0.5 |
| ESSENT13 | 8 | 17 | 38 | 30 | 68 | 93 | 316 | 2229 | 1 | 24 | 17 | 0.142* | 44.7 | 0.1 |
| ESSENT14 | 27 | 51 | 239 | 184 | 423 | 418 | 2659 | 129496 | 17 | 141 | 106 | 0.021 | 54.6 | 0.8 |
| ESSENT15 | 22 | 40 | 198 | 164 | 362 | 311 | 2155 | 97209 | 16 | 133 | 92 | 0.022 | 47.8 | 0.7 |
| ESSENT16 | 13 | 51 | 329 | 261 | 590 | 337 | 3540 | 117757 | 38* | 227 | 146 | 0.030 | 106.4 | 1.1 |
| ESSENT17 | 63 | 108 | 686 | 476 | 1162 | 1106 | 8620* | 1196680 | 30* | 403* | 317 | 0.007 | 62.1 | 2.7 |
| ESSENT18 | 13 | 26 | 93 | 83 | 176 | 170 | 930 | 19302 | 9 | 197 | 49 | 0.048 | 44.8 | 0.3 |
| ESSENT19 | 19 | 14 | 59 | 49 | 108 | 134 | 545 | 18114 | 5 | 45 | 37 | 0.030 | 16.4 | 0.2 |
| ESSENT20 | 6 | 5 | 8 | 8 | 16 | 27 | 55 | 266 | 1 | 9 | 5 | 0.208* | 11.5 | 0.0 |
| ESSENT21 | 18 | 32 | 117 | 99 | 216 | 235 | 1219 | 33944 | 6 | 101 | 59 | 0.036 | 43.8 | 0.4 |
| ESSENT22 | 50 | 56 | 289 | 226 | 515 | 607 | 3465 | 349582 | 61* | 192 | 170 | 0.010 | 34.3 | 1.1 |
| ESSENT23 | 14 | 13 | 35 | 31 | 66 | 101 | 314 | 5238 | 6 | 31 | 22 | 0.060 | 18.8 | 0.1 |
| ESSENT24 | 11 | 9 | 23 | 21 | 44 | 67 | 190 | 2440 | 4 | 19 | 14 | 0.078 | 14.8 | 0.1 |
| ESSENT25 | 36 | 86 | 549 | 375 | 924 | 739 | 6404* | 502640 | 33* | 376* | 284 | 0.013 | 81.6 | 2.0 |
| ESSENT26 | 14 | 15 | 33 | 33 | 66 | 112 | 321 | 4938 | 5 | 27 | 20 | 0.065 | 20.8 | 0.1 |
| ESSENT27 | 11 | 9 | 15 | 15 | 30 | 67 | 130 | 1189 | 3 | 15 | 10 | 0.109 | 14.1 | 0.0 |
| ESSENT28 | 6 | 5 | 11 | 11 | 22 | 27 | 76 | 502 | 1 | 11 | 7 | 0.152* | 11.5 | 0.0 |
| ESSENT29 | 5 | 5 | 12 | 10 | 22 | 23 | 73 | 365 | 1 | 12 | 8 | 0.200* | 14.6 | 0.0 |
| ESSENT30 | 6 | 5 | 7 | 6 | 13 | 27 | 45 | 162 | 2 | 9 | 5 | 0.278* | 12.5 | 0.0 |
| ESSENT31 | 29 | 69 | 394 | 306 | 700 | 562 | 4630* | 297748 | 24 | 221 | 181 | 0.016 | 72.0 | 1.4 |
| ESSENT32 | 27 | 46 | 209 | 162 | 371 | 382 | 2296 | 109180 | 14 | 131 | 100 | 0.021 | 48.3 | 0.7 |
| ESSENT33 | 37 | 79 | 451 | 374 | 825 | 691 | 5658* | 495526 | 39* | 297* | 218 | 0.011 | 64.6 | 1.8 |
| ESSENT34 | 11 | 57 | 192 | 146 | 338 | 371 | 2058 | 28986 | 3 | 116 | 86 | 0.071 | 146.1 | 0.6 |
| ESSENT35 | 28 | 53 | 158 | 119 | 277 | 438 | 1756 | 55202 | 6 | 94 | 74 | 0.032 | 55.9 | 0.5 |

| | $\eta_1$ | $\eta_2$ | $N_1$ | $N_2$ | N | $\hat{N}$ | V | E | v(G) | Li | St | $\hat{L}$ | I | $\hat{B}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ESSENT36 | 48 | 64 | 450 | 230 | 680 | 652 | 4629* | 399251 | 19 | 278* | 224 | 0.012 | 53.7 | 1.4 |
| ESSENT37 | 13 | 26 | 52 | 46 | 98 | 170 | 518 | 5957 | 2 | 37 | 29 | 0.087 | 45.0 | 0.2 |
| ESSENT38 | 16 | 50 | 150 | 153 | 303 | 346 | 1831 | 44834 | 12 | 133 | 88 | 0.041 | 74.8 | 0.6 |
| ESSENT39 | 26 | 56 | 284 | 221 | 505 | 447 | 3211 | 164713 | 20 | 168 | 135 | 0.019 | 62.6 | 1.0 |
| ESSENT40 | 21 | 37 | 184 | 146 | 330 | 285 | 1933 | 80094 | 21 | 143 | 101 | 0.024 | 46.7 | 0.6 |
| ESSENT41 | 27 | 55 | 292 | 231 | 523 | 446 | 3325 | 188527 | 19 | 188 | 146 | 0.018 | 58.6 | 1.0 |
| ESSENT42 | 37 | 72 | 423 | 350 | 773 | 637 | 5232* | 470499 | 37* | 282* | 203 | 0.011 | 58.2 | 1.6 |
| ESSENT43 | 13 | 17 | 40 | 32 | 72 | 118 | 353 | 4323 | 3 | 23 | 19 | 0.082 | 28.9 | 0.1 |
| ESSENT44 | 10 | 10 | 27 | 25 | 52 | 66 | 225 | 2809 | 3 | 18 | 11 | 0.080 | 18.0 | 0.1 |
| ESSENT45 | 15 | 49 | 327 | 247 | 574 | 334 | 3444 | 13C204 | 5 | 183 | 151 | 0.026 | 91.1 | 1.1 |
| ESSENT46 | 38 | 71 | 485 | 336 | 821 | 636 | 5557* | 499632 | 22 | 270* | 222 | 0.011 | 61.8 | 1.7 |
| ESSENT47 | 23 | 50 | 221 | 167 | 388 | 386 | 2402 | 92247 | 13 | 130 | 103 | 0.026 | 62.5 | 0.8 |
| ESSENT48 | 28 | 44 | 220 | 150 | 370 | 375 | 2283 | 108955 | 13 | 124 | 94 | 0.021 | 47.8 | 0.7 |
| ESSENT49 | 34 | 55 | 247 | 169 | 416 | 491 | 2694 | 140720 | 19 | 153 | 114 | 0.019 | 51.6 | 0.8 |
| ESSENT50 | 29 | 85 | 498 | 387 | 885 | 686 | 6047* | 399216 | 40* | 306* | 228 | 0.015 | 91.6 | 1.9 |
| ESSENT51 | 25 | 51 | 240 | 193 | 433 | 405 | 2705 | 127974 | 21 | 143 | 109 | 0.021 | 57.2 | 0.8 |
| ESSENT52 | 29 | 68 | 435 | 318 | 753 | 555 | 4970* | 336992 | 32* | 268* | 193 | 0.015 | 73.3 | 1.6 |
| ESSENT53 | 26 | 32 | 107 | 79 | 186 | 282 | 1090 | 34969 | 4 | 78 | 53 | 0.031 | 34.0 | 0.3 |
| ESSENT54 | 33 | 74 | 431 | 354 | 785 | 626 | 5292* | 417715 | 39* | 309* | 223 | 0.013 | 67.0 | 1.7 |
| ESSENT55 | 24 | 41 | 178 | 143 | 321 | 330 | 1933 | 80911 | 12 | 110 | 80 | 0.024 | 46.2 | 0.6 |
| ESSENT56 | 25 | 37 | 126 | 98 | 224 | 309 | 1334 | 44158 | 8 | 74 | 57 | 0.030 | 40.3 | 0.4 |
| ESSENT57 | 32 | 52 | 290 | 233 | 523 | 456 | 3343 | 239680 | 9 | 192 | 145 | 0.014 | 46.6 | 1.0 |
| Average | 22 | 41 | 200 | 152 | 352 | 336 | 2250 | 141744 | 14 | 132 | 96 | 0.053 | 49.1 | 0.7 |
| Std Dev | 12 | 25 | 163 | 119 | 281 | 231 | 1980 | 205466 | 13 | 100 | 77 | 0.059 | 29.6 | 0.6 |

Total Scores

| $\eta_1$ | $\eta_2$ | $N_1$ | $N_2$ | v(G) | St | LOC |
|---|---|---|---|---|---|---|
| 157 | 1059 | 11393 | 8676 | 730 | 5491 | 11656 |

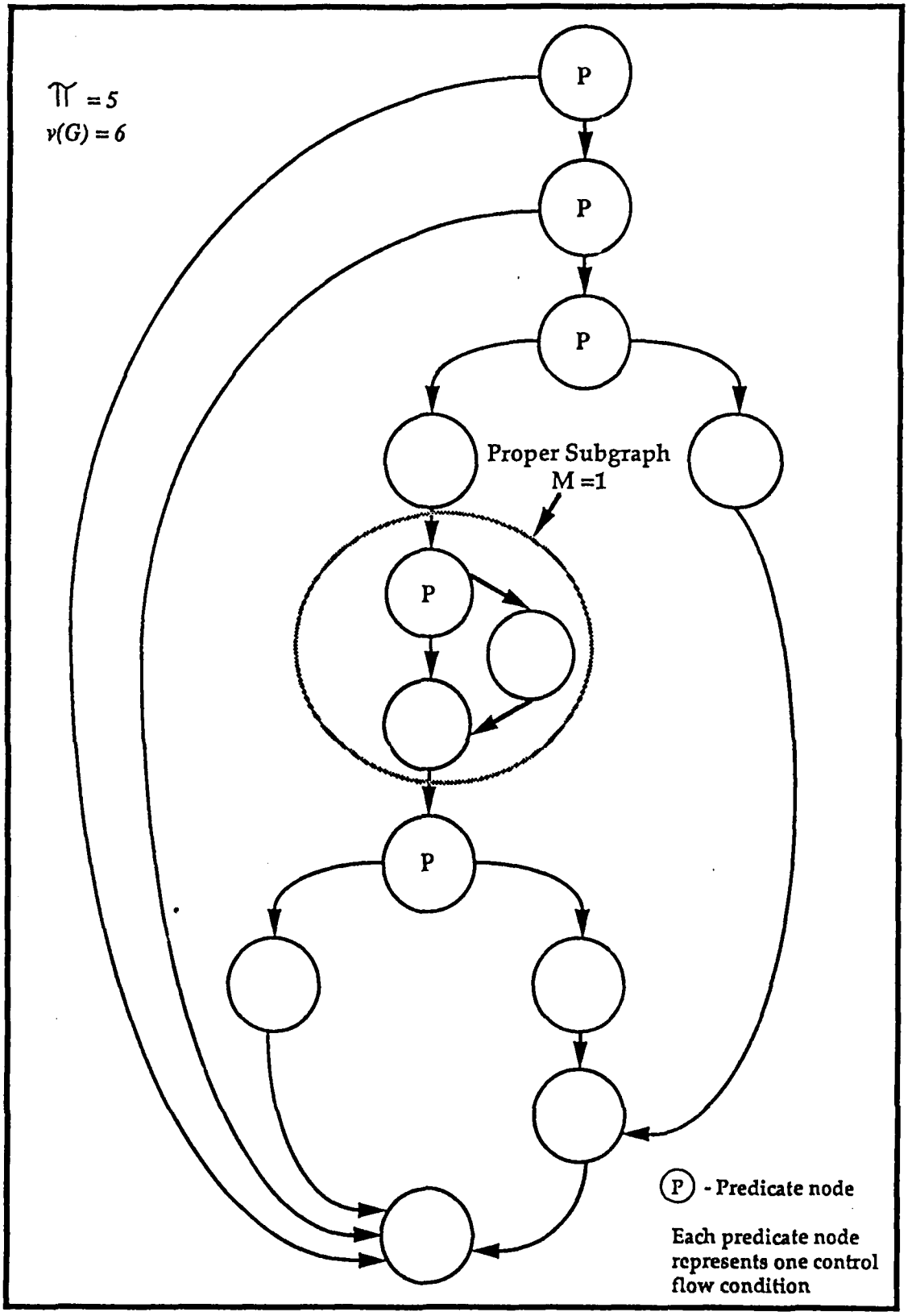
The two modules of critical code had the Halstead and McCabe measures shown in the following table. The total scores are the measures of the code when the modules are treated as one large module.

| | $\eta_1$ | $\eta_2$ | $N_1$ | $N_2$ | N | $\hat{N}$ | V | E | v(G) | Li | St | $\hat{L}$ | I | $\hat{B}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CRITIC01 | 19 | 22 | 79 | 62 | 141 | 179 | 755 | 20225 | 6 | 47 | 37 | 0.037 | 28.2 | 0.2 |
| CRITIC02 | 25 | 37 | 126 | 98 | 224 | 309 | 1334 | 44158 | 8 | 74 | 57 | 0.030 | 40.3 | 0.4 |
| Average | 22 | 30 | 103 | 80 | 183 | 244 | 1045 | 32192 | 7 | 61 | 47 | 0.034 | 34.3 | 0.3 |
| Std Dev | 3 | 8 | 24 | 18 | 42 | 65 | 290 | 11967 | 1 | 14 | 10 | 0.004 | 6.0 | 0.1 |

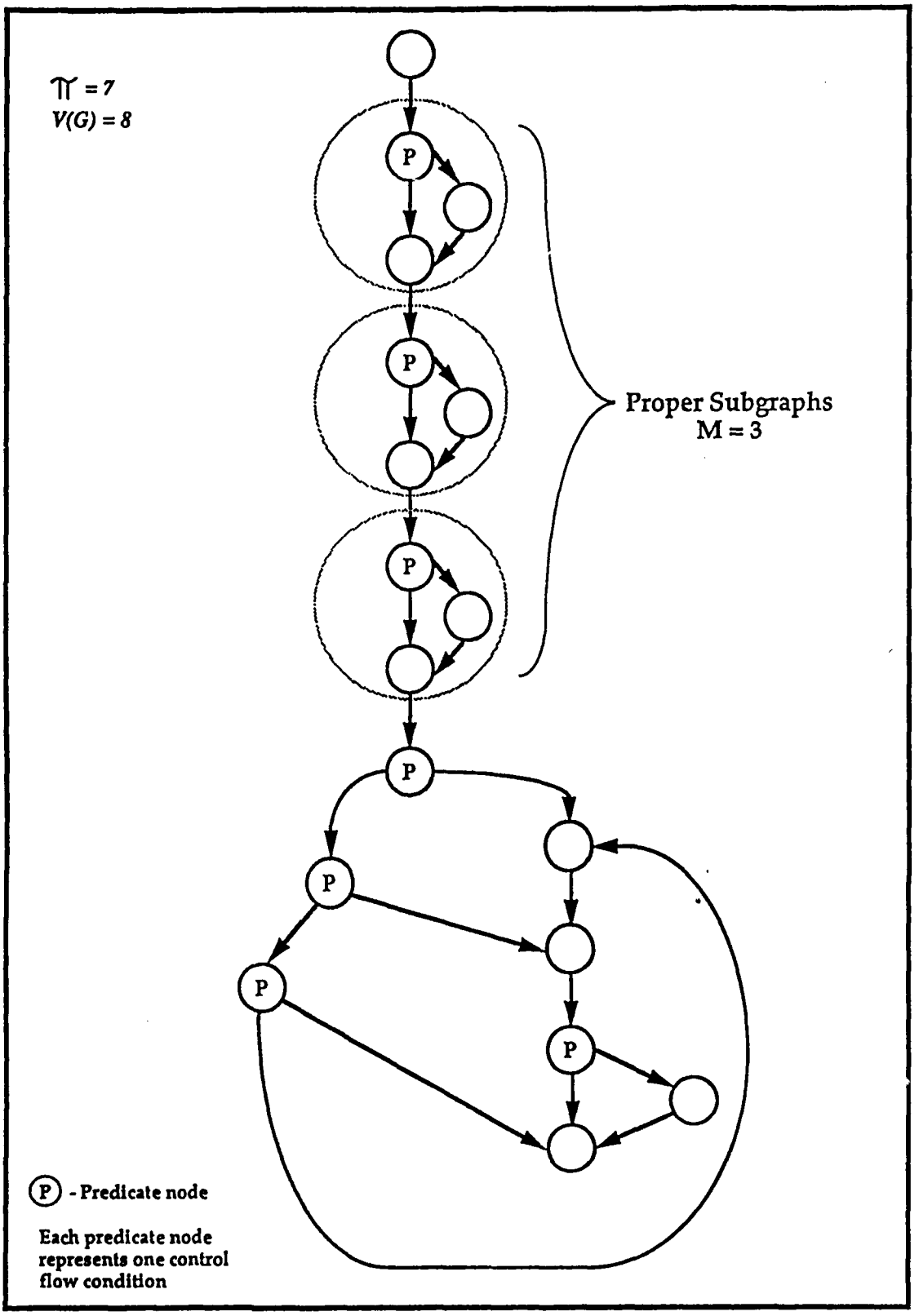
Total Scores

| $\eta_1$ | $\eta_2$ | $N_1$ | $N_2$ | v(G) | St | LOC |
|---|---|---|---|---|---|---|
| 28 | 51 | 205 | 160 | 13 | 94 | 220 |

The program control graphs for CRITIC01 and CRITIC02 are shown on the following two pages.




PROGRAM CONTROL GRAPH OF CRITICO1




PROGRAM CONTROL GRAPH OF CRITICO2

The 84 modules of nonessential code had the RADC measures shown in the following tables.

| | MO.1(3) | MO.1(7) | SD.2(1) | GE.1(1) | FS.1(2) |
|---|---|---|---|---|---|
| NONESS01 | 0 | 1 | 0 | 0 | 1 |
| NONESS02 | 1 | 1 | 1 | 0 | 1 |
| NONESS03 | 0 | 1 | 1 | 0 | 1 |
| NONESS04 | 1 | 1 | 1 | 0 | 1 |
| NONESS05 | 0 | 1 | 1 | 0 | 1 |
| NONESS06 | 1 | 1 | 1 | 0 | 1 |
| NONESS07 | 0 | 1 | 0 | 0 | 1 |
| NONESS08 | 1 | 1 | 1 | 0 | 1 |
| NONESS09 | 1 | 1 | 0 | 1 | 1 |
| NONESS10 | 1 | 1 | 0 | 0 | 1 |
| NONESS11 | 0 | 1 | 1 | 0 | 1 |
| NONESS12 | 0 | 1 | 1 | 0 | 1 |
| NONESS13 | 0 | 1 | 1 | 0 | 1 |
| NONESS14 | 1 | 1 | 1 | 0 | 1 |
| NONESS15 | 1 | 1 | 1 | 1 | 1 |
| NONESS16 | 1 | 0 | 0 | 1 | 1 |
| NONESS17 | 1 | 1 | 0 | 0 | 1 |
| NONESS18 | 0 | 1 | 1 | 0 | 1 |
| NONESS19 | 0 | 0 | 0 | 0 | 1 |
| NONESS20 | 0 | 1 | 0 | 0 | 1 |
| NONESS21 | 1 | 1 | 1 | 0 | 1 |
| NONESS22 | 1 | 1 | 0 | 0 | 1 |
| NONESS23 | 0 | 0 | 0 | 0 | 1 |
| NONESS24 | 1 | 1 | 1 | 0 | 1 |
| NONESS25 | 0 | 1 | 1 | 0 | 1 |
| NONESS26 | 1 | 1 | 0 | 0 | 1 |
| NONESS27 | 0 | 1 | 1 | 1 | 1 |
| NONESS28 | 0 | 1 | 1 | 1 | 1 |
| NONESS29 | 1 | 1 | 1 | 0 | 1 |
| NONESS30 | 1 | 1 | 0 | 0 | 1 |
| NONESS31 | 1 | 1 | 0 | 0 | 1 |
| NONESS32 | 1 | 1 | 0 | 0 | 1 |
| NONESS33 | 1 | 1 | 0 | 0 | 1 |
| NONESS34 | 1 | 1 | 0 | 0 | 1 |
| NONESS35 | 0 | 1 | 1 | 0 | 1 |
| NONESS36 | 1 | 1 | 1 | 0 | 1 |
| NONESS37 | 1 | 1 | 1 | 0 | 1 |
| NONESS38 | 1 | 1 | 0 | 0 | 1 |
| NONESS39 | 1 | 1 | 1 | 0 | 1 |
| NONESS40 | 1 | 1 | 0 | 0 | 1 |

| | MO.1(3) | MO.1(7) | SD.2(1) | GE.1(1) | FS.1(2) |
|---|---|---|---|---|---|
| NONESS41 | 0 | 1 | 1 | 0 | 1 |
| NONESS42 | 0 | 1 | 0 | 0 | 1 |
| NONESS43 | 1 | 1 | 1 | 0 | 1 |
| NONESS44 | 1 | 1 | 0 | 0 | 1 |
| NONESS45 | 1 | 1 | 1 | 0 | 1 |
| NONESS46 | 0 | 1 | 1 | 0 | 1 |
| NONESS47 | 1 | 1 | 0 | 0 | 1 |
| NONESS48 | 1 | 1 | 0 | 0 | 1 |
| NONESS49 | 1 | 1 | 1 | 0 | 1 |
| NONESS50 | 0 | 1 | 0 | 0 | 1 |
| NONESS51 | 0 | 1 | 0 | 0 | 1 |
| NONESS52 | 1 | 1 | 0 | 1 | 1 |
| NONESS53 | 1 | 1 | 0 | 0 | 1 |
| NONESS54 | 0 | 1 | 1 | 1 | 1 |
| NONESS55 | 0 | 1 | 1 | 0 | 1 |
| NONESS56 | 0 | 1 | 1 | 0 | 1 |
| NONESS57 | 0 | 1 | 0 | 0 | 1 |
| NONESS58 | 1 | 1 | 0 | 0 | 1 |
| NONESS59 | 0 | 0 | 1 | 0 | 1 |
| NONESS60 | 1 | 1 | 1 | 0 | 1 |
| NONESS61 | 0 | 1 | 1 | 1 | 1 |
| NONESS62 | 1 | 1 | 1 | 0 | 1 |
| NONESS63 | 0 | 0 | 0 | 0 | 1 |
| NONESS64 | 0 | 0 | 0 | 0 | 1 |
| NONESS65 | 0 | 1 | 1 | 0 | 1 |
| NONESS66 | 0 | 1 | 1 | 0 | 1 |
| NONESS67 | 1 | 1 | 0 | 1 | 1 |
| NONESS68 | 1 | 1 | 0 | 1 | 1 |
| NONESS69 | 1 | 1 | 0 | 0 | 1 |
| NONESS70 | 0 | 1 | 1 | 0 | 1 |
| NONESS71 | 0 | 1 | 1 | 0 | 1 |
| NONESS72 | 0 | 1 | 1 | 0 | 1 |
| NONESS73 | 0 | 1 | 0 | 0 | 1 |
| NONESS74 | 0 | 1 | 1 | 0 | 1 |
| NONESS75 | 1 | 1 | 1 | 0 | 1 |
| NONESS76 | 1 | 1 | 0 | 0 | 1 |
| NONESS77 | 1 | 1 | 0 | 0 | 1 |
| NONESS78 | 1 | 0 | 0 | 0 | 1 |
| NONESS79 | 0 | 1 | 0 | 0 | 1 |
| NONESS80 | 1 | 1 | 1 | 1 | 1 |

| | MO.1(3) | MO.1(7) | SD.2(1) | GE.1(1) | FS.1(2) |
|---|---|---|---|---|---|
| NONESS81 | 0 | 1 | 1 | 0 | 1 |
| NONESS82 | 0 | 1 | 1 | 0 | 1 |
| NONESS83 | 0 | 0 | 0 | 0 | 1 |
| NONESS84 | 1 | 0 | 0 | 0 | 1 |
| Total | 45 | 75 | 44 | 11 | 84 |

Modularity, MO = MO.1(3) & MO.1(7)

| | | |
|---|---|---|
| MO.1(3) | 45/ 84 = | 0.54 |
| MO.1(7) | 75/ 84 = | 0.89 |

DS 120/168 = DR 0.71 SO 0.71

Maintainability = MO & SD

| | | |
|---|---|---|
| MO | 120/168 = | 0.71 x 0.5 = 0.36 |
| SD | 44/ 84 = | 0.52 x 0.5 = 0.26 |

DS 164/252 = DR 0.65 SO 0.62 WSO 0.62

Expandability = GE & MO & SD

| | | |
|---|---|---|
| GE | 11/ 84 = | 0.13 x 0.6 = 0.08 |
| MO | 120/168 = | 0.71 x 0.2 = 0.14 |
| SD | 44/ 84 = | 0.52 x 0.2 = 0.10 |

DS 175/336 = DR 0.52 SO 0.46 WSO 0.33

Reusability = FS & GE & MO & SD

| | | |
|---|---|---|
| FS | 84/ 84 = | 1.00 x 0.3 = 0.30 |
| GE | 11/ 84 = | 0.13 x 0.3 = 0.04 |
| MO | 120/168 = | 0.71 x 0.2 = 0.14 |
| SD | 44/ 84 = | 0.52 x 0.2 = 0.10 |

DS 259/420 = DR 0.62 SO 0.59 WSO 0.59

The 59 modules of essential code had the RADC measures shown in the following tables.

| | MO.1(3) | MO.1(7) | SD.2(1) | GE.1(1) | FS.1(2) |
|---|---|---|---|---|---|
| ESSENT01 | 1 | 1 | 0 | 0 | 1 |
| ESSENT02 | 0 | 1 | 0 | 0 | 1 |
| ESSENT03 | 1 | 1 | 1 | 0 | 1 |
| ESSENT04 | 1 | 0 | 0 | 0 | 1 |
| ESSENT05 | 1 | 1 | 0 | 0 | 1 |
| ESSENT06 | 1 | 0 | 0 | 0 | 1 |
| ESSENT07 | 1 | 1 | 0 | 1 | 1 |
| ESSENT08 | 1 | 1 | 0 | 0 | 1 |
| ESSENT09 | 1 | 1 | 0 | 0 | 1 |
| ESSENT10 | 0 | 1 | 0 | 0 | 1 |
| ESSENT11 | 0 | 1 | 0 | 0 | 1 |
| ESSENT12 | 1 | 1 | 0 | 0 | 1 |
| ESSENT13 | 1 | 1 | 0 | 0 | 1 |
| ESSENT14 | 0 | 1 | 0 | 0 | 1 |
| ESSENT15 | 0 | 0 | 0 | 0 | 1 |
| ESSENT16 | 0 | 1 | 0 | 0 | 1 |
| ESSENT17 | 0 | 0 | 0 | 0 | 1 |
| ESSENT18 | 0 | 1 | 0 | 0 | 1 |
| ESSENT19 | 0 | 0 | 0 | 0 | 1 |
| ESSENT20 | 1 | 1 | 0 | 1 | 1 |
| ESSENT21 | 1 | 1 | 0 | 0 | 1 |
| ESSENT22 | 1 | 1 | 0 | 0 | 1 |
| ESSENT23 | 0 | 1 | 0 | 1 | 1 |
| ESSENT24 | 1 | 1 | 0 | 1 | 0 |
| ESSENT25 | 1 | 1 | 0 | 1 | 0 |
| ESSENT26 | 0 | 0 | 0 | 0 | 1 |
| ESSENT27 | 1 | 1 | 0 | 0 | 1 |
| ESSENT28 | 1 | 1 | 0 | 1 | 0 |
| ESSENT29 | 1 | 1 | 0 | 1 | 1 |
| ESSENT30 | 1 | 1 | 0 | 1 | 1 |
| ESSENT31 | 1 | 1 | 0 | 1 | 1 |
| ESSENT32 | 0 | 1 | 0 | 0 | 1 |
| ESSENT33 | 0 | 1 | 0 | 0 | 1 |
| ESSENT34 | 0 | 1 | 0 | 0 | 1 |
| ESSENT35 | 0 | 1 | 0 | 0 | 1 |
| ESSENT36 | 0 | 0 | 0 | 0 | 1 |
| ESSENT37 | 0 | 1 | 0 | 0 | 1 |
| ESSENT38 | 1 | 0 | 0 | 0 | 1 |
| ESSENT39 | 1 | 0 | 0 | 0 | 1 |
| ESSENT40 | 0 | 1 | 0 | 0 | 1 |

| | MO.1(3) | MO.1(7) | SD.2(1) | GE.1(1) | FS.1(2) |
|---|---|---|---|---|---|
| ESSENT41 | 0 | 1 | 0 | 1 | 1 |
| ESSENT42 | 0 | 1 | 0 | 0 | 1 |
| ESSENT43 | 0 | 1 | 0 | 0 | 1 |
| ESSENT44 | 1 | 1 | 0 | 1 | 1 |
| ESSENT45 | 1 | 0 | 0 | 0 | 1 |
| ESSENT46 | 0 | 1 | 0 | 0 | 1 |
| ESSENT47 | 0 | 1 | 0 | 0 | 1 |
| ESSENT48 | 0 | 1 | 0 | 0 | 1 |
| ESSENT49 | 0 | 1 | 0 | 0 | 1 |
| ESSENT50 | 0 | 1 | 0 | 0 | 1 |
| ESSENT51 | 0 | 1 | 0 | 0 | 1 |
| ESSENT52 | 0 | 1 | 0 | 0 | 1 |
| ESSENT53 | 0 | 1 | 0 | 0 | 1 |
| ESSENT54 | 1 | 0 | 0 | 0 | 1 |
| ESSENT55 | 1 | 1 | 0 | 0 | 1 |
| ESSENT56 | 0 | 1 | 0 | 0 | 1 |
| ESSENT57 | 1 | 1 | 0 | 0 | 1 |
| ESSENT58 | 1 | 1 | 0 | 0 | 1 |
| ESSENT59 | 0 | 1 | 0 | 0 | 1 |
| Total | 28 | 48 | 1 | 11 | 56 |

Modularity, MO = MO.1(3) & MO.1(7)

| | | |
|---|---|---|
| MO.1(3) | 28/ 59 = | 0.47 |
| MO.1(7) | 48/ 59 = | 0.81 |

DS 76/118 = DR 0.64 SO 0.64

Maintainability = MO & SD

| | | |
|---|---|---|
| MO | 76/118 = | 0.64 x 0.5 = 0.32 |
| SD | 1/ 59 = | 0.02 x 0.5 = 0.01 |

DS 77/177 = DR 0.44 SO 0.33 WSO 0.33

Expandability = GE & MO & SD

| | | |
|---|---|---|
| GE | 11/ 59 = | 0.19 x 0.6 = 0.11 |
| MO | 76/118 = | 0.64 x 0.2 = 0.13 |
| SD | 1/ 59 = | 0.02 x 0.2 = 0.00 |

DS 88/236 = DR 0.37 SO 0.28 WSO 0.24

Reusability = FS & GE & MO & SD

| | | |
|---|---|---|
| FS | 56/ 59 = | 0.95 x 0.3 = 0.28 |
| GE | 11/ 59 = | 0.19 x 0.3 = 0.06 |
| MO | 76/118 = | 0.64 x 0.2 = 0.13 |
| SD | 1/ 59 = | 0.02 x 0.2 = 0.00 |

DS 144/295 = DR 0.49 SO 0.45 WSO 0.47

The two modules of critical code had the RADC measures shown in the following tables.

| | MO.1(3) | MO.1(7) | SD.2(1) | GE.1(1) | FS.1(2) |
|---|---|---|---|---|---|
| CRITIC01 | 1 | 1 | 0 | 0 | 1 |
| CRITIC02 | 1 | 1 | 0 | 0 | 1 |
| Total | 2 | 2 | 0 | 0 | 2 |

Modularity, MO = MO.1(3) & MO.1(7)

| | | | |
|---|---|---|---|
| MO.1(3) | 2/ 2 = | 1.00 | |
| MO.1(7) | 2/ 2 = | 1.00 | |
| DS | 4/ 4 = DR 1.00 | SO 1.00 | |

Maintainability = MO & SD

| | | | |
|---|---|---|---|
| MO | 4/ 4 = | 1.00 x 0.5 = 0.50 | |
| SD | 0/ 2 = | 0.00 x 0.5 = 0.00 | |
| DS | 4/ 6 = DR 0.67 | SO 0.50 | WSO 0.50 |

Expandability = GE & MO & SD

| | | | |
|---|---|---|---|
| GE | 0/ 2 = | 0.00 x 0.6 = 0.00 | |
| MO | 4/ 4 = | 1.00 x 0.2 = 0.20 | |
| SD | 0/ 2 = | 0.00 x 0.2 = 0.00 | |
| DS | 4/ 8 = DR 0.50 | SO 0.33 | WSO 0.20 |

Reusability = FS & GE & MO & SD

| | | | |
|---|---|---|---|
| FS | 2/ 2 = | 1.00 x 0.3 = 0.30 | |
| GE | 0/ 2 = | 0.00 x 0.3 = 0.00 | |
| MO | 4/ 4 = | 1.00 x 0.2 = 0.20 | |
| SD | 0/ 2 = | 0.00 x 0.2 = 0.00 | |
| DS | 6/ 10 = DR 0.60 | SO 0.50 | WSO 0.50 |

## BIBLIOGRAPHY


Akiyama, Fumio, "An Example of Software System Debugging", Proceedings IFIP Congress, 1971, pp. 353-358.

Albrecht, A. J., "Measuring Application Development Productivity", Proceedings of the IBM Application Development Symposium, 1979.

Albrecht, Allan J., "Function Points Help Managers Assess Applications", Computerworld, August 26, 1985.

Albrecht, Allan J. and John E. Gaffney, Jr., "Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation", IEEE Transactions on Software Engineering, Volume SE-9, No. 6, November 1983, pp. 639-648.

Amster, S. J. et al., "An Experiment in Automatic Quality Evaluation of Software", Proceedings of the Symposium on Computer Software Engineering, New York Polytechnic, New York, NY, 1976.

Arthur, J., "Metrics: Tools to Aid Software Code Quality", Computerworld, Volume 17, No. 26, June 27, 1983.

Atwood, M. E. et al., "Annotated Bibliography on Human Factors in Software Development", ARI Technical Report, P-79-1, Science Applications, Inc., Englewood, CO, June 1979.

Bailey, C. T. and W. L. Dingee, "A Software Study Using Halstead Metrics", ACM Performance Evaluation Review - SIGMETRICS, 1981 ACM Workshop/Symposium on Measurement and Evaluation of Software Quality, March 1981, pp. 189-197.

Baker, Albert L. and Stuart H. Zweben, "A Comparison of Measures of Control Flow Complexity", Proceedings of COMPSAC 1979, IEEE, 1979.

Baker, Albert L. and Stuart H. Zweben, "The Use of Software Science in Evaluating Modularity Concepts", IEEE Transactions on Software Engineering, Volume SE-5, No. 2, March 1979, pp. 110-120.

Basili, V. R. and T. Phillips, "Evaluating and Comparing Software Metrics in the Software Engineering Laboratory", ACM SIGMETRICS (1981 ACM Workshop/Symposium Measurement and Evaluation of Software Quality), Volume 10, March 1981, pp. 95-106.

Basili, V. R., R. W. Selby, Jr., and T. Y. Phillips, "Metric Analysis and Data Validation Across FORTRAN Projects", IEEE Transactions on Software Engineering, Volume SE-9, No. 6, November 1983.

Basili, Victor R. and H. Dieter Rombach, "Implementing Quantitative SQA: A Practical Model", IEEE Software, Volume 4, No. 5, September 1987, pp. 6-9.

Basili, Victor R. and David H. Hutchens, "An Empirical Study of a Syntactic Complexity Family", IEEE Transactions on Software Engineering, Volume SE-9, No. 6, November 1983, pp. 664-672.

Basili, Victor R. and Robert W. Reiter, Jr., "Evaluating Automatable Measures of Software Development", Department of Computer Science, University of Maryland, College Park, MD, 1979.

Behrens, C., "Measuring the Productivity of Computer Systems Development Activities with Function Points", IEEE Transactions on Software Engineering, Volume SE-9, No. 6, November 1983, pp. 648-652.

Bell, D. E. and J. E. Sullivan, Further Investigations into the Complexity of Software, MTR-2874, MITRE, Bedford, MA, 1974.

Boehm, B. W. et al., Proceedings of the TRW Symposium on Reliable, Cost-Effective, Secure Software, Los Angeles, CA, March 20-21, 1974.

Boehm, B. W. et al., Characteristics of Software Quality, North-Holland Publishing Company, New York, NY, 1978.

Bowen, Thomas P., Gary B. Wigle, and Jay T. Tsai, Specification of Software Quality Attributes, RADC-TR-85-37, RADC, Griffiss Air Force Base, NY, Volumes I, II, and III, February 1985.

Brainerd, Walter S., Charles H. Goldberg, and Jonathan L. Gross, FORTRAN 77 Fundamentals and Style, Boyd & Fraser Publishing Co., Boston, MA, 1985.

Browne, Jim C., "A Proposal for Structural Models of Software Systems", Computer Science and Statistics: Proceedings of the 13th Symposium on the Interface, Springer-Verlag, Inc., New York, NY, 1981, pp. 208-210.

Buckley, Fletcher J., "Standard Set of Useful Software Metrics is Urgently Needed", Computer, Volume 22, No. 7, July 1989, pp. 88-89.

Bulut, N. and M. H. Halstead, "Impurities Found in Algorithm Implementations", SIGPLAN Notices, Volume 9, No. 3, March 1974, pp. 9-11.

Campbell, D. T. and J. C. Stanley, Experimental and Quasi-Experimental Designs for Research, Houghton Mifflin Company, Boston, MA, 1963.

Card, David N., "Major Obstacles Hinder Successful Measurement", IEEE Software, Volume 5, No. 6, November 1988, pp. 82 and 86.

Carver, D. L., "Criteria for Estimating Module Complexity", Journal of Systems Management, August 1986.

Chen, Edward T., "Program Complexity and Programmer Productivity", IEEE Transactions on Software Engineering, Volume SE-4, No. 3, May 1978, pp. 187-194.

Cote, V. et al., "Software Metrics: An Overview of Recent Results", The Journal of Systems and Software, Volume 8, No. 2, March 1988, pp. 121-131.

Coulter, Neal S., "Software Science and Cognitive Psychology", IEEE Transactions on Software Engineering, Volume SE-9, No. 2, March 1983, pp. 166-171.

Coupal, Daniel and Pierre Robillard, "Factor Analysis of Source Code Metrics", The Journal of Systems and Software, Volume 12, No. 3, July 1990, pp. 263-270.

Crawford, S. G., A. A. McIntosh, and D. Pregibon, "An Analysis of Static Metrics and Faults in C Software", The Journal of Systems and Software, Volume 5, No. 1, February 1985, pp. 37-48.

Currans, Nancy, "A Comparison of Counting Methods for Software Science and Cyclomatic Complexity", Pacific Northwest Software Quality Conference.

Curtis, B., "Measurement and Experimentation in Software Engineering", Proceedings of the IEEE, IEEE, New York, NY, Volume 68, No. 9, September 1980, pp. 1144-1157.

Curtis, B. et al., "Measuring the Psychological Complexity of Software Maintenance Tasks with the Halstead and McCabe Metrics", IEEE Transactions on Software Engineering, Volume SE-5, No. 2, March 1979.

Denicoff, Marvin, "Software Metrics: Paradigms and Processes", Computer Science and Statistics: Proceedings of the 13th Symposium on the Interface, Springer-Verlag, Inc., New York, NY, 1981, pp. 205-207.

Deutsch, Michael S., "Application of an Automated Verification System" Software Verification and Validation Realistic Project Approaches, Prentice Hall, Englewood Cliffs, NJ, 1982.

Dickson, et al., "Quantitative Analysis of Software Reliability", Proceedings of 1972 Reliability and Maintainability Symposium, Annals of Assurance Science, IEEE, Catalog No. 72CH0577-7R, pp. 148-157.

Drummond, S., "Measuring Applications Development Performance", Datamation, February 1985, pp. 103-108.

Duncan, Otis Dudley, Introduction to Structural Equation Models, Academic Press, Inc., New York, NY, 1975, pp. 1-168.

Dunsmore, H. E., "Software Metrics: An Overview of an Evolving Methodology", Information Processing and Management, Volume 20, No. 1-2, 1984, pp. 183-192.

Ejiogu[1], Lem O., "A Simple Measure of Software Complexity", Computerworld, Volume 18, No. 14, April 2, 1984.

Ejiogu[2], Lem O., "Software Structure: Its Characteristic Polynomials", SIGPLAN Notices, Volume 19, No. 12, December 1984, pp. 1-7.

Ejiogu, Lem O., "The Critical Issues of Software Metrics", Sigplan Notices, Volume 22, No. 3, March 1987, pp. 1-6.

Ejiogu, Lem O., "A Unified Theory of Software Metrics", Softmetrix, Inc., Chicago, IL, 1988, pp. 232-238.

Ejiogu, Lem O., "Beyond Structured Programming: An Introduction to the Principles of Applied Software Metrics", Structured Programming, Springer-Verlag, Inc., New York, NY, 1990.

Elshoff, James L., "A Numerical Profile of Commercial PL/I Programs", GMR-1927, General Motors Corporation Research Laboratories, Warren, MI, September 1975.

Elshoff, James L., "Measuring Commercial PL/I Programs Using Halstead's Criteria", SIGPLAN Notices, Volume 11, No. 5, May 1976, pp. 38-46.

Elshoff, James L., "The PEEK Measurement Program", GMR-4208, General Motors Corporation Research Laboratories, Warren, MI, November 15, 1982.

Elshoff, James L., "Characteristic Program Complexity Measures", GMR-4446R, General Motors Corporation Research Laboratories, Warren, MI, December 5, 1983.

Elwell, D. and N. VanSuetendael, Software Quality Metrics, DOT/FAA/CT-91/1, U.S. Department of Transportation, Federal Aviation Administration, 1991.

Evangelist, W. M., "Software Complexity Metric Sensitivity to Program Structuring Rules", The Journal of Systems and Software, Volume 3, No. 3, September 1983.

Farr, Leonard and Henry J. Zagorski, "Quantitative Analysis of Programming Cost Factors: A Progress Report", Economics of Automatic Data Processing. ICC Symposium Proceedings 1965 Rome, North-Holland, Amsterdam, Holland, 1965, pp. 167-180.

Federal Aviation Regulations, Part 25 Airworthiness Standards: Transport Category Airplanes, Subpart F, Section 25.1309.

Ferdinand, A. E., "A Theory of System Complexity", International Journal of General Systems, Volume 1, No. 1, 1974.

Feuche, Mike, "Attention is Being Generated by Complexity Metrics Tools", MIS Week, Volume 9, No. 9, February 29, 1988.

Feuer, Alan, R. and Edward B. Fowlkes, "Some Results from an Empirical Study on Computer Software", Bell Telephone Laboratories, 1979.

Fisher, R. A., "The Design of Experiments", Oliver and Boyd, London, United Kingdom, 1935.

Fitzsimmons, Ann B., "Relating the Presence of Software Errors to the Theory of Software Science", Proceedings of the Eleventh Hawaii International Conference on Systems Sciences, Western Periodicals, 1978.

Fitzsimmons, Ann B. and Tom L. Love, "A Review and Evaluation of Software Science", ACM Computing Surveys, Volume 10, No. 1, March 1978, pp. 1-18.

Funami, Y. and M. H. Halstead, "A Software Physics Analysis of Akiyama's Debugging Data", Proceedings of the MRI 24th International Symposium: Software Engineering, Polytechnic Press, New York, NY, 1976

Gaffney, Jr., John E., "Software Metrics: A Key to Improved Software Development Management", Computer Science and Statistics: Proceedings of the 13th Symposium on the Interface, Pittsburgh, PA, March 12 and 13, 1981, pp. 211-220.

Gaffney, Jr., John E. and Richard R. Reynolds, "Specifying Performance Requirements for a Degradable System - The Case of an Unmanned Weather Station", presented at CMG XIV International Conference on Computer Performance Evaluation, Crystal City, VA, December 1983.

Gaffney, Jr., John E., "Estimating the Number of Faults in Code", IEEE Transactions on Software Engineering, Volume SE-10, No. 4, July 1984, pp. 459-464.

Gaffney, Jr., John E., "The Impact on Software Development Costs of using HOL's", IEEE Transactions on Software Engineering, Volume SE-12, No. 3, March 1986.

Gaffney, Jr., John E. and Charles F. Davis, An Approach to Estimating Software Errors and Availability, SPC-TR-88-007, Software Productivity Consortium, Herndon, VA, March 1988.

Gibson, Virginia R. and James A. Senn, "System Structure and Software Maintenance Performance," Communications of the ACM, March 1989, Volume 32, No. 3, pp. 347-358.

Gilb, T., Software Metrics, Winthrop, Inc., Cambridge, MA, 1977.

Gordon, Ronald D. and Maurice H. Halstead, "An Experiment Comparing FORTRAN Programming Times with the Software Physics Hypothesis", AFIPS Conference Proceedings - National Computer Conference, Volume 45, 1976, pp. 935-937.

Gordon[1], Ronald D., "Measuring Improvements in Program Clarity", IEEE Transactions on Software Engineering, Volume SE-5, No. 2, March 1979, pp. 79-90.

Gordon[2], Ronald D., "A Qualitative Justification for a Measure of Program Clarity", IEEE Transactions on Software Engineering, Volume SE-5, No. 2, March 1979, pp. 121-128.

Gorla, Narasimhaiah, Alan C. Benander, and Barbara A. Benander, "Debugging Effort Estimation Using Software Metrics", IEEE Transactions on Software Engineering, Volume 16, No. 2, February 1990, pp. 223-231.

Gould, John D., "Some Psychological Evidence on How People Debug Computer Programs", International Journal of Man-Machine Studies, Volume 7, pp. 151-181.

Gremillion, L. L., "Determinants of Program Repair Maintenance Requirements", Communications of the ACM, Volume 27, No. 8, pp. 826ff.

Halstead, Maurice H., "Natural Laws Controlling Algorithm Structure", SIGPLAN Notices, Volume 7, No. 2, 1972.

Halstead, Maurice H. and P. M. Zislis, Experimental Verification of Two Theorems of Software Physics, CSD-TR-97, Purdue University, Lafayette, IN, June 12, 1973.

Halstead, Maurice H., Elements of Software Science, Elsevier North Holland Publishing Company, Inc., New York, NY, 1977, pp. 19-26.

Halstead, Maurice H., "Advances in Software Science", Advances in Computers, Volume 18, NY Academic, 1979.

Harrison[1], W., "MAE: A Syntactic Metric Analysis Environment", The Journal of Systems and Software, Volume 8, 1988, pp. 57-62.

Harrison[2], W., "Using Software Metrics to Allocate Testing Resources", Journal of Management Information Systems, Volume 4, No. 4, Spring 1988.

Harrison, W. A., "Applying McCabe's Complexity Measure to Multiple-Exit Programs", Software - Practice & Experience, Volume 14, No. 10, October 1984.

Harrison, W. and C. Cook, "Are Deeply Nested Conditionals Less Readable?", The Journal of Systems and Software, Volume 6, 1986, pp.335-341.

Harrison, W. and C. Cook, "Insights On Improving the Maintenance Process Through Software Measures", Oregon State University, Computer Science Department, Corvallis, OR, March 1990.

Harrison, Warren and Curtis Cook, "A Reduced Form For Sharing Software Complexity Data", Oregon State University, Computer Science Department, Corvallis, OR, 1983.

Harrison, W. and K. Magel, "A Complexity Measure Based On Nesting Level", The Journal of Systems and Software, Volume 6, 1986, pp. 335-341.

Harrison, W. et al., "Applying Software Complexity Metrics To Program Maintenance", Computer, Volume 15, No. 9, September 1988.

Hays, William L. and Robert L. Winkler, Statistics, Holt, Rinehart, and Winston, Inc., New York, NY, 1970.

Henry, S. and D. Kafura, "The Evaluation of Software Systems' Structure Using Quantitative Software Metrics", Software - Practice and Experience, Volume 14, No. 6, June 1984.

Henry, S., D. Kafura, and K. Harris, "On the Relationships Among Three Software Metrics", 1981 ACM Workshop/Symposium on Measurement and Evaluation of Software Quality, March 1981.

IEEE Computer Society, IEEE Guide for the Use of IEEE Standard Dictionary of Measures to Produce Reliable Software, IEEE Std 982.1-1988, April 1989.

IEEE Computer Society, IEEE Guide for the Use of IEEE Standard Dictionary of Measures to Produce Reliable Software, IEEE Std 982.2-1988, June 1989.

Jones, Capers, "Building a Better Metric", Computerworld, Volume 22, No. 25, June 20, 1988, pp. 38-39.

Jones, T. C., "Measuring Programming Quality and Productivity", IBM Systems Journal, Volume 17, No. 1, 1978, pp. 39-63.

Kafura, Dennis and Geereddy R. Reddy, "The Use of Software Complexity Metrics in Software Maintenance", IEEE Transactions on Software Engineering, Volume SE-13, No. 3, March 1987, pp 335-343.

Keller, Steven E., "Using Metrics to Specify Software Quality", Ada Quality Quarterly, Systems Division of Dynamics Research Corporation, Andover, MA, 1989.

Kelvin, W. T., "Popular Lectures and Addresses, 1981-1984", Workshop on Quantitative Software Models for Reliability, Complexity, and Cost, IEEE, New York, NY, 1980.

Kernighan, B. W. and P. J. Plauger, The Elements of Programming Style, McGraw-Hill, New York, NY, 1974.

Kitchenham, B. A., "Measures of Programming Complexity", ICL Technical Journal, Volume 2, No. 3, 1981, pp. 298ff.

Krause, K. W., R. W. Smith, and M. A. Goodwin, "Optimal Software Test Planning through Automated Network Analysis", Proceedings IEEE Computer Software Reliability Symposium, 1973.

Krause, K. W., R. W. Smith, and M. A. Goodwin, Optimal Software Test Planning through Automated Network Analysis, TRW-SS-73-01, TRW Engineering and Integration Division, Redondo Beach, CA, April 1973, pp. 1-5.

Kreitzberg, Charles B. and Ben Shneiderman, FORTRAN Programming: A Spiral Approach, Second Edition, Harcourt Brace Jovanovich, Inc., 1982.

Kruger, Gregory A., "Project Management Using Software Reliability Growth Models", Hewlett-Packard Journal, Volume 39, No. 3, June 1988, pp. 30-35.

Lasky, Jeffrey A., Alan R. Kaminsky, and Wade Boaz, Software Quality Measurement Methodology Enhancements Study Results, Final Technical Report, RADC-TR-89-317, RADC, Griffiss Air Force Base, NY, January 1990.

Lassez, J.-L., D. Van Der Knijff, J. Shepherd, and C. Lassez, "A Critical Examination of Software Science", The Journal of Systems and Software, Volume 2, December 1981, pp. 105-112.

Lennselius, Bo, Claes Wohlin, and Ctirad Vrana, "Software Metrics: Fault Content Estimation and Software Process Control", Microprocessors and Microsystems, Volume 11, No. 7, September 1987, pp. 365-375.

Lew, Ken S., Tharam S. Dillon, and Kevin E. Forward, "Software Complexity and Its Impact on Software Reliability", IEEE Transactions on Software Engineering, Volume 14, No. 11, November 1988, pp. 1645-1655.

Li, H. F. and W. K. Cheung, "An Empirical Study of Software Metrics", IEEE Transactions on Software Engineering, Volume SE-13, No. 6, June 1987.

Lind, Randy K. and K. Vairavan, "An Experimental Investigation of Software Metrics and Their Relationship to Software Development Effort", IEEE Transactions on Software Engineering, Volume 15, No. 5, May 1989, pp. 649-653.

Littlewood, B. (edited by), Software Reliability - Achievement and Assessment, Blackwell Scientific Publications, Oxford, United Kingdom, 1987.

Love, L. T. and A. B. Bowman, "An Independent Test of the Theory of Software Physics", SIGPLAN Notices, Volume 11, November 1976, pp. 42-49.

Low, Graham C. and D. Ross Jeffery, "Function Points in the Estimation and Evaluation of the Software Process", IEEE Transactions on Software Engineering, Volume 16, No. 1, January 1990, pp. 64-71.

Mannino, Phoebe, Bob Stoddard, and Tammy Sudduth, "The McCabe Software Complexity Analysis as a Design and Test Tool", Texas Instruments Technical Journal, Volume 7, No. 2, March-April 1990, pp. 41-53.

McAuliffe, Daniel, "Measuring Program Complexity", Computer, Volume 21, No. 6, June 1988, pp. 97-98.

McCabe, T. J., "A Complexity Measure", IEEE Transactions on Software Engineering, Volume SE-2, No. 4, 1976.

McCabe, Thomas J., Structured Testing: A Software Testing Methodology Using the Cyclomatic Complexity Metric, U.S. Department of Commerce - National Bureau of Standards Special Publication #500-99, Washington, DC, 1982.

McCabe, Thomas J., "The Cyclomatic Complexity Metric", Hewlett Packard Journal, April 1989.

McCabe, Thomas J. and Charles W. Butler, "Design Complexity Measurement and Testing", Communications of the ACM, December 1989, pp. 1415-1425.

McCall, Jim A., Paul K. Richards, and Gene F. Walters, Factors in Software Quality, RADC-TR-77-369, Volumes I, II, and III, RADC, Griffiss Air Force Base, NY, November 1977.

McClure, Carma L., "A Model for Program Complexity Analysis", Proceedings of the Third International Conference on Software Engineering, IEEE, New York, NY, 1978, p. 149-157.

Millman, P. and B. Curtis, A Matched Project Evaluation of Modern Programming Practices (Management Report), RADC-TR-80-6, Volume 1 of 2, RADC, Griffiss Air Force Base, NY, 1980.

Millman, P. and B. Curtis, A Matched Project Evaluation of Modern Programming Practices (Scientific Report), RADC-TR-80-6, Volume 2 of 2, RADC, Griffiss Air Force Base, NY, 1980.

Mohanty, Siba N., "Models and Measurements for Quality Assessment of Software", Computing Surveys, Volume 11, No. 3, September 1972, pp. 251-275.

Moranda, P. B., "Is Software Science Hard?", (in Surveyors' Forum) ACM Computer Surveys, Volume 10, No. 4, December 1978, pp. 503-504.

Murine[1], Gerald E., "The Application of Software Quality Metrics", Phoenix Conference on Computers and Communications, IEEE, Carlsbad, CA, 1983.

Murine[2], Gerald E., "Improving Management Visibility Through the Use of Software Quality Metrics", Proceedings, Seventh International Computer Software and Applications Conference, Chicago, IL, November 7-11, 1983.

Murine[3], Gerald E., "On Validating Software Quality Metrics", 4th Annual Phoenix Conference, Phoenix, AZ, March 1985.

Murine[4], Gerald E., "The Role of Software Quality Metrics in the Software Quality Evaluation Process", Proceedings, Ninth International Computer Software and Applications Conference, Chicago, IL, October 1985.

Murine, Gerald E., "Software Measurement Techniques", Proceedings, Measurement Science Conference, Long Beach, CA, January 1988.

Murine, Gerald E. and Rita Cerv, "The New Science of Software Quality Metrology", Proceedings, Measurement Science Conference, Irvine, CA, January 1986.

Musa, John D., "Faults, Failures, and a Metrics Revolution", IEEE Software, Volume 6, No. 2, March 1989, pp. 85 and 91.

Myers, G. J., Software Reliability, John Wiley and Sons, Inc., New York, NY, 1976.

Myers, G. J., "An Extension to the Cyclomatic Measure of Program Complexity", SIGPLAN Notices, Volume 12, No. 10, 1977.

Myers, G. J., Composite/Structured Design, Van Nostrand Reinhold Company, New York, NY, 1978.

Myers, G. J., The Art of Software Testing, John Wiley and Sons, Inc. New York, NY, 1979.

Ntafos, Simeon C., "A Comparison of Some Structural Testing Strategies", IEEE Transactions on Software Engineering, Volume 14, No. 6, June 1988, pp. 868-874.

Ohba, M., "Software Reliability Analysis Models", IBM Journal of Research and Development, Volume 28, No. 4, July 1984, pp. 428-443.

Oldehoeft, R. R., "A Contrast Between Language Level Measures", IEEE Transactions on Software Engineering, Volume SE-3, No. 6, November 1977, pp. 476-478.

Ottenstein, Linda M., "Quantitative Estimates of Debugging Requirements", IEEE Transactions on Software Engineering, Volume SE-5, No. 5, September 1979, pp. 504-514.

Ottenstein, Linda M., "Predicting Numbers of Errors Using Software Science", ACM Performance Evaluation Review - SIGMETRICS, 1981 ACM Workshop/Symposium on Measurement and Evaluation of Software Quality, March 1981, pp. 157-167.

Parnas, D. L., "On the Criteria to be Used in Decomposing Systems Into Modules", Communications of the ACM, Volume 15, No. 12, December 1972.

Perlis, A. J., F. G. Sayward, and M. Shaw (Editors), Software Metrics: An Analysis and Evaluation, MIT Press, Cambridge, MA, 1981.

Perrone, Giovanni, "Program Measures Complexity of Software", PC Week, Volume 5, No. 13, March 29, 1988.

Perry, W. E., "Lack of Measurements Plague IS Assessments", Information Systems News, May 2, 1983.

Pierce, Patricia, Richard Hartley, and Suellen Worrells, Software Quality Measurement Demonstration Project II Final Technical Report, RADC-TR-87-164, RADC, Griffiss Air Force Base, NY, October 1987.

Pippenger, Nicholas, "Complexity Theory", Scientific American, June 1978, pp. 114-124.

Prather, R. E., "Theory of Program Testing - An Overview", Bell System Technical Journal, Volume 62, No. 10, Part 2, December 1983.

Prather, R. E., "An Axiomatic Theory of Software Complexity Measure", The Computer Journal, Volume 27, No. 4, November 1984.

Quantitative Software Management Inc., "Hanscom AFB gets a Handle on Vendor Performance", QSM Perspectives, McLean, VA, Fall 1990.

Radio Technical Commission for Aeronautics, Software Considerations in Airborne Systems and Equipment Certification, Document No. RTCA/DO-178A, March 22, 1985.

Ramamurthy, Bina and Austin Melton, "A Synthesis of Software Science Measures and the Cyclomatic Number", IEEE Transactions on Software Engineering, Volume 14, No. 8, August 1988, pp. 1116-1121.

Reynolds, R. G., "Metrics to Measure the Complexity of Partial Programs", The Journal of Systems and Software, Volume 4, No. 1, April 1984.

Robillard, Pierre N. and Germinal Boloix, "The Interconnectivity Metrics: A New Metric Showing How a Program is Organized", The Journal of Systems and Software, Volume 10, No. 1, July 1989, pp. 29-39.

Schneidewind, N. F. and H. M. Hoffman, "An Experiment in Software Error Data Collection and Analysis", IEEE Transactions on Software Engineering, Volume SE-5, No. 3, May 1979, pp. 276-286.

Selby, Richard W. and Adam A. Porter, "Learning from Examples: Generation and Evaluation of Decision Trees for Software Resource Analysis", IEEE Transactions on Software Engineering, Volume 14, No. 12, December 1988, pp. 1743-1757.

Shatz, Sol M., "Towards Complexity Metrics for Ada Tasking", IEEE Transactions on Software Engineering, Volume 14, No. 8, August 1988, pp. 1122-1127.

Shen, Vincent Y., Samuel D. Conte, and H. E. Dunsmore, "Software Science Revisited: A Critical Analysis of the Theory and its Empirical Support", IEEE Transactions on Software Engineering, Volume SE-9, No. 2, March 1983, pp. 155-165.

Shneiderman, B., Software Psychology: Human Factors in Computer and Information Systems, Winthrop Publishing Co., Cambridge, MA, 1980, pp. 94-120.

Shumskas, Anthony F., "A Layered Software Test and Evaluation Strategy for a Layered Defense System", ITEA Journal, Volume XI, No. 3, 1990.

Siyan, Karanjit S., "Coping with Complex Programs", Dr. Dobb's Journal of Software Tools, Volume 14, No. 3, March 1989.

Society of Automotive Engineers, Fault/Failure Analysis for Digital Systems and Equipment, Aerospace Recommended Practice 1834, August 7, 1986.

Software Engineering Research Review - Quantitative Software Models, Data and Analysis Center for Software (DACS), Griffiss Air Force Base, NY, March 1979.

Sullivan, J. E., "Measuring the Complexity of Computer Software", MTR-2648, MITRE Corporation, Bedford, MA, 1973.

Sunazuka, Toshihiko, Motoei Azuma, and Noriko Yamagishi, "Software Quality Assessment Technology", Proceedings, 8th International Conference on Software Engineering, August 28-30, 1985, pp. 142-148.

Szulewski, Paul A., Mark H. Whitworth, Philip Buchan, and J. Barton DeWolf, "The Measurement of Software Science Parameters in Software Designs", ACM Performance Evaluation Review - SIGMETRICS, 1981 ACM Workshop/Symposium on Measurement and Evaluation of Software Quality, March 1981, pp. 89-94.

Taft, Darryl K., "Quality Analysis Aids Ada Programming", Government Computer News, Volume 7, No. 7, April 1, 1988.

Takahashi, Muneo and Yuji Kamayachi, "An Empirical Study of a Model for Program Error Prediction", IEEE Transactions on Software Engineering, Volume 15, No. 1, January 1989, pp. 82-86.

Thayer, T. A., M. Lipow, and E. C. Nelson, "Software Reliability Study", TRW Software Series, TRW-SS-76-03, March 1976.

U.S. Department of the Air Force, Air Force Systems Command Software Management Indicators, AFSCP 800-14, January 20, 1986.

U.S. Department of the Air Force, Air Force Systems Command Software Quality Indicators, AFSCP 800-43, January 31, 1986.

U.S. Department of Defense, Defense System Software Quality Program, Military Standard DOD-STD-2168, August 1, 1979.

U.S. Department of Defense, Defense System Software Development, Military Standard DOD-STD-2167A, June 4, 1985.

U.S. Department of Transportation, Federal Aviation Administration, "System Design and Analysis", Advisory Circular, No. 25.1309-1A, June 1988.

Van Der Knijff, D. J. J., "Software Physics and Program Analysis", The Australian Computer Journal, Volume 10, No. 3, August 1978, pp. 82-86.

Van Der Poel, K. G. and S. R. Schach, "A Software Metric for Cost Estimation and Efficiency Measurement in Data Processing System Development", The Journal of Systems and Software, Volume 3, No. 3, September 1983.

Verilog Products Catalog, Verilog USA Inc., Alexandria, VA, January 1990.

Verilog, Logiscope Automated Code Analyzer, Technical Presentation, December 1989.

Voldman, J. et al., "Fractal Nature of Software - Cache Interaction", IBM Journal of Research and Development, Volume 27, No. 2, March 1983, pp. 164-170.

Walsh, T. J., "A Software Reliability Study Using a Complexity Measure", Proceedings of the National Computer Conference, AFIPS, 1979.

Walters, G. F., "Applications of Metrics to a Software Quality Management (QM) Program", Software Quality Management, Petrocelli, New York, NY, 1979.

Ward, W. T., "Software Defect Prevention Using McCabe's Complexity Metric", Hewlett Packard Journal, April 1989.

Warthman, James L., Software Quality Measurement Demonstration Project I, Final Technical Report, RADC-TR-87-247, RADC, Griffiss Air Force Base, NY, December 1987.

Weyuker, Elaine J., "Evaluating Software Complexity Measures", IEEE Transactions on Software Engineering, Volume 14, No. 9, September 1988, pp. 1357-1365.

Woodfield, S. N., V. Y. Shen, and H. E. Dunsmore, "A Study of Several Metrics for Programming Effort", Journal of Systems and Software, Volume 2, December 1981, pp. 97-103.

Woodward, Martin R., Michael A. Hennell, and David Hedley, "A Measure of Control Flow Complexity in Program Text", IEEE Transactions on Software Engineering, Volume SE-5, No. 1, January 1979, pp. 45-50.

Yu, Tze-Jie, Vincent Y. Shen, and Hubert E. Dunsmore, "An Analysis of Several Software Defect Models", IEEE Transactions on Software Engineering, Volume 14, No. 9, September 1988, pp. 1261-1270.

Zislis, Paul M., An Experiment in Algorithm Implementation, CSD-TR-96, Department of Computer Science, Purdue University, Lafayette, IN, June 1973.

Zweben, S. H., "A Study of the Physical Structure of Algorithms", IEEE Transactions on Software Engineering, Volume SE-3, No. 3, May 1977, pp. 250-258.

Zweben, Stuart H., Software Physics: Resolution of an Ambiguity in the Counting Procedure, TR 93, Purdue University, Lafayette, IN.

## GLOSSARY

ANTAGONISTIC QUALITY FACTORS. Quality Factors with conflicting attributes.

BINARY SEARCH. A searching algorithm in which the search population is repeatedly divided into two equal or nearly equal sections.

BITS. Binary digits.

CODE. The subset of software which exists for the sole purpose of being loaded into a computer to control it.

COMPLEMENTARY QUALITY FACTORS. Quality Factors with interrelated attributes.

CONTROL STRUCTURES. Programming constructs which direct the flow of control.

EQUIVALENCE STATEMENT. A FORTRAN statement which equates two variable names.

HARDWARE. The physical components of a computer.

METRIC. A measure.

MODULE. A unit of code which implements a function.

MONOTONIC FUNCTION. A function in which a certain change in the measure always represents a certain change in the property being measured, where either change is simply an increase or decrease in magnitude.

OBJECT CODE. The translation of source code that is loaded into a computer.

OPERANDS. The variables or constants on which the operators act.

OPERATORS. Symbols which affect the value or ordering of operands.

OPTIMIZING COMPILER. A computer program which, while translating source code into object code, removes inefficiencies from the code.

PROGRAM. A detailed set of instructions for accomplishing some purpose.

QUALITY MEASURE. A repeatable, monotonic relationship relating measures of objects (a set of numbers) to subjective qualities.

SOFTWARE METRIC. A measure of software objects.

SOFTWARE QUALITY FACTOR. Any software attribute that contributes either directly or indirectly, positively or negatively, toward the objectives for the system in which the software resides.

SOFTWARE QUALITY METRIC. (1) A measure that relates measures of the software objects (the symbols) to the software qualities (quality factors). (2) The measure of a software quality factor.

SOFTWARE. Computer programs and the documentation associated with the programs.

SOURCE CODE. Code that can be read by people.

STROUD NUMBER. The total number of elementary mental discriminations that a person makes per second.

SUBROUTINE. A self-contained body of code which can be called by other routines to perform a function.

WELL-BEHAVED FUNCTION. A smooth mathematical relationship.

## ACRONYMS AND ABBREVIATIONS

| | |
|---|---|
| $\lambda$ | Language Level |
| $1/E_0$ | Average number of discriminations a person is likely to make for each bug introduced into the code. |
| AC | Advisory Circular |
| ACT | Analysis of Complexity Tool |
| ARP | Aerospace Recommended Practice |
| $\hat{B}$ | Number of Bugs (Estimated) |
| BAT | Battlemap Analysis Tool |
| CE | Certification Engineer |
| CFR | Code of Federal Regulations |
| CR/LF | Carriage Return/Line Feed |
| CSCI | Computer Software Configuration Item |
| D | Program Difficulty |
| DR | Direct Ratio (Average) |
| DS | Direct Score |
| E | Programming Effort |
| EOF | End of File |
| FAA | Federal Aviation Administration |
| FAR | Federal Aviation Regulation |
| FP | Function Point |
| HOL | High Order Language |
| I | Intelligence Content |
| IEEE | Institute of Electrical and Electronics Engineers |
| IFC | Information Flow Complexity |
| ISO | International Standards Organization |
| L | Program Level |
| $\hat{L}$ | Estimated Program Level |
| $\hat{N}$ | Estimated Length |
| $\eta$ | Vocabulary of a Program |
| $\eta_1$ | Number of Unique Operators |
| $\eta_2$ | Number of Unique Operands |
| $\eta_1^*$ | Minimum Number of Unique Operators |
| $\eta_2^*$ | Number of Different Input and Output Parameters |
| N | Implementation Length of a Program |
| $N_1$ | Total Number of Operator Occurrences |
| $N_2$ | Total Number of Operand Occurrences |
| $N_1^*$ | Minimum Number of Operators |
| $N_2^*$ | Minimum Number of Operands |
| PC | Processing Complexity |
| PCA | Processing Complexity Adjustment |
| PROM | Programmable Read-Only Memory |
| RADC | Rome Air Development Center |
| RAM | Random-Access Memory |
| RTCA | Radio Technical Commission for Aeronautics |
| S | Stroud Number |

| | |
|---|---|
| SAE | Society of Automotive Engineers |
| SO | Second Order (Average) |
| SQF | Software Quality Factor |
| SQM | Software Quality Metrics |
| SQPP | Software Quality Program Plan |
| SRS | Software Requirement Specification |
| SSS | System/Segment Specification |
| $\hat{T}$ | Estimated Programming Time |
| V | Volume |
| $V^*$ | Potential Volume |
| WSO | Weighted Second Order (Average) |



☆U.S. GOVERNMENT PRINTING OFFICE: 1992 - 684-061/60053

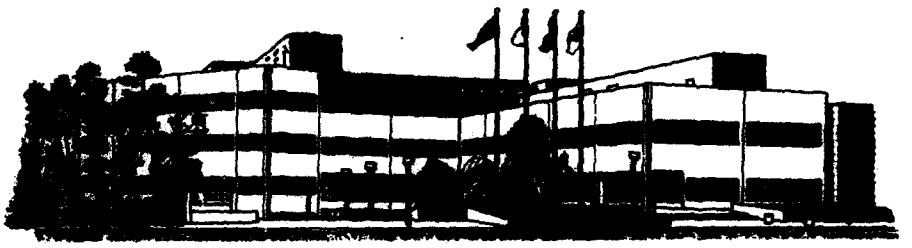
AD-A275 323

U.S. Department of Transportation
Federal Aviation Administration

DOT/FAA/CT-88/10

# HANDBOOK - VOLUME II
# DIGITAL SYSTEMS VALIDATION

## CHAPTER 18
## AVIONIC DATA BUS
## INTEGRATION TECHNOLOGY



DTIC
ELECTE
FEB 01 1994
S B D

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

94-03085

FEDERAL AVIATION ADMINISTRATION
TECHNICAL CENTER
ATLANTIC CITY INTERNATIONAL AIRPORT, NEW JERSEY 08405

94 1 31 191

Best
Available
Copy

## NOTICE

This document is disseminated under the sponsorship of the U. S. Department of Transportation in the interest of information exchange. The United States Government assumes no liability for the contents or use thereof.

The United States Government does not endorse products or manufacturers. Trade or manufacturers' names appear herein solely because they are considered essential to the objective of this report.

Technical Report Documentation Page

| 1. Report No. | 2. Government Accession No. | 3. Recipient's Catalog No. |
|---|---|---|
| DOT/FAA/CT-88/10 | | |

| 4. Title and Subtitle | 5. Report Date |
|---|---|
| Handbook - Volume II<br>Digital Systems Validation<br>Chapter 18<br>Avionic Data Bus Integration Technology | November 1993 |
| | 6. Performing Organization Code |
| 7. Author(s) | 8. Performing Organization Report No. |
| D. Elwell, L. Harrison, J. Hensyl, and N. VanSuetendael | DOT/FAA/CT-88/10-Vol-2-Ch-18 |
| 9. Performing Organization Name and Address | 10. Work Unit No. (TRAIS) |
| Computer Resource Management, Inc.<br>200 Scarborough Drive, Suite 108<br>Pleasantville, New Jersey 08232 | 11. Contract or Grant No.<br>DTFA03-86-C-00042 |
| | 13. Type of Report and Period Covered |
| 12. Sponsoring Agency Name and Address | Tutorial<br>Handbook Chapter 18 |
| U.S. Department of Transportation<br>Federal Aviation Administration<br>Technical Center<br>Atlantic City International Airport, NJ 08405 | 14. Sponsoring Agency Code<br>ACD-230 |

15. Supplementary Notes

Pete Saraceni, FAA Technical Center, Program Manager, (609) 485-5577

(Note: This Tutorial is a condensed version of FAA Technical Center Final Report DOT/FAA/CT-91/19)

16. Abstract

As multiple digital avionic systems were introduced into aircraft, there arose a need for digital communications between systems. In the early 1970's, many different digital data bus designs were used to provide this communication. Because these digital systems proved to be reliable and cost effective, their popularity increased. Proliferation led to standardization, particularly in the air transport category of aircraft, which allowed communications between line replaceable units (LRUs) to become more complex. The LRUs began to rely more heavily on each other to reduce the amount of equipment required. Sensor data and systems data could be shared among multiple systems, rather than each system requiring its own private source.

Integrated digital avionics are increasingly being used to implement essential and critical functions that cannot be sufficiently reproduced by conventional means. The safety of such aircraft is highly dependent upon the computer software, hardware, and data buses connecting the systems. The newest concerns relate to the problems that are unique to highly integrated systems. There is no standard with which to assess the possible impact of these bus-based systems on aircraft safety. These and other advanced avionic systems will result in specific safety assessment problems when the appropriate data packages are submitted to the Federal Aviation Administration during the certification process.

| 17. Key Words | 18. Distribution Statement |
|---|---|
| Avionics, Data Bus, Integration, Buffer, Controller, Network, Protocol, Digital, Software, Error, Fault, Frame, Interrupt, Parity, Station, Token, Multiplexing | Document is available to the U.S. public through the National Technical Information Service, Springfield, VA 22161 |

| 19. Security Classif. (of this report) | 20. Security Classif. (of this page) | 21. No. of Pages | 22. Price |
|---|---|---|---|
| Unclassified | Unclassified | 180 | |

Form DOT F 1700.7 (8-72) Reproduction of completed page authorized

TABLE OF CONTENTS


| Section | | | Page |
|---|---|---|---|
| 1. | INTRODUCTION | | 18-1 |
| | 1.1 | Background | 18-1 |
| | 1.2 | Scope | 18-2 |
| 2. | BUS-INTEGRATED AVIONIC SYSTEMS | | 18-5 |
| | 2.1 | Avionic System Architectures | 18-5 |
| | 2.2 | Avionic Data Buses | 18-9 |
| | 2.3 | Aircraft Implementations | 18-11 |
| 3. | CERTIFICATION PROCEDURES FOR BUS-INTEGRATED SYSTEMS | | 18-13 |
| | 3.1 | Types of Certification | 18-13 |
| | 3.2 | Conducting Certification Testing | 18-15 |
| | | 3.2.1 Approaches to Bus Reliability | 18-16 |
| | | 3.2.2 Testing Data Buses | 18-16 |
| | 3.3 | Certification Concerns | 18-17 |
| 4. | RELATED REGULATIONS AND STANDARDS | | 18-19 |
| | 4.1 | Relevance of Formal Guidelines to Bus-Integrated Systems | 18-19 |
| | | 4.1.1 Bus-Integrated Avionic Systems and Federal Aviation Regulations | 18-20 |
| | | 4.1.2 Bus-Integrated Avionic Systems and Advisory Circulars | 18-22 |
| | | 4.1.3 Bus-Integrated Avionic Systems and Special Conditions | 18-24 |
| | 4.2 | Relevance of Informal Guidelines to Federal Regulations | 18-26 |
| | | 4.2.1 Radio Technical Commission for Aeronautics DO-160C | 18-27 |
| | | 4.2.2 Radio Technical Commission for Aeronautics DO-178A | 18-27 |
| | | 4.2.3 Society of Automotive Engineers ARP 1834 | 18-30 |
| | 4.3 | Relevance of Manufacturer Testing to Federal Regulations | 18-31 |
| | | 4.3.1 ARINC 429 Data Bus | 18-32 |
| | | 4.3.2 Commercial Standard Data Bus | 18-33 |
| | | 4.3.3 ARINC 629 Data Bus | 18-33 |
| | | 4.3.4 Avionics Standard Communications Bus | 18-35 |
| | | 4.3.5 Summary | 18-36 |

TABLE OF CONTENTS
(Continued)


| Section | | Page |
|---|---|---|
| 5. | BUS-INTEGRATED SYSTEMS TECHNOLOGY | 18-39 |
| 5.1 | System Integration Concerns | 18-39 |
| 5.1.1 | Architecture Related Concerns | 18-41 |
| 5.1.2 | Protocol Related Concerns | 18-47 |
| 5.1.3 | Data Integrity Concerns | 18-60 |
| 5.2 | Bus Hardware-Software Interaction | 18-81 |
| 5.2.1 | Bus Interface Units and Central Processing Units | 18-82 |
| 5.2.2 | Hardware-Software Interaction Faults | 18-85 |
| 5.2.3 | Fault Detection | 18-90 |
| 5.2.4 | Fault Correction | 18-94 |
| 5.2.5 | Summary | 18-102 |
| 5.3 | Bus Protocol Specification and Verification Methods | 18-102 |
| 5.3.1 | A Protocol Specification Guideline | 18-102 |
| 5.3.2 | Protocol Specification Content | 18-103 |
| 5.3.3 | Protocol Specification Methods | 18-104 |
| 5.3.4 | Protocol Verification Methods | 18-108 |
| 5.3.5 | Application to Avionic Data Buses | 18-110 |
| 5.3.6 | Summary | 18-113 |
| 5.4 | Bus Integration Standards, Guidelines, and Techniques | 18-114 |
| 5.4.1 | Levels of Integration | 18-114 |
| 5.4.2 | The Ideal Bus Integration Standard | 18-116 |
| 5.4.3 | Bus Integration Standards and Guidelines | 18-118 |
| 5.4.4 | Bus Integration Techniques | 18-118 |
| 5.4.5 | FAA Certification and Bus Integration | 18-132 |
| 5.4.6 | Summary | 18-138 |
| 6. | CONCLUSIONS | 18-139 |
| 6.1 | Certification Procedures for Bus-Integrated Systems | 18-139 |
| 6.2 | Related Regulations and Standards | 18-140 |
| 6.3 | Bus-Integrated Systems Technology | 18-141 |
| 6.3.1 | System Integration Concerns | 18-141 |
| 6.3.2 | Bus Hardware-Software Interaction | 18-141 |
| 6.3.3 | Bus Protocol Specification and Analysis | 18-142 |

TABLE OF CONTENTS
(Continued)


| Section | | Page |
|---|---|---|
| 6.3.4 | Bus Integration Standards, Guidelines, and Techniques | 18-142 |
| 6.3.5 | FAA Certification and Bus Integration | 18-143 |
| 6.4 | Summary | 18-144 |
| APPENDICES | | |
| A - | DYNAMIC TIME SLOT ALLOCATION PROTOCOL | 18-145 |
| B - | HIGH-LEVEL DATA LINK CONTROL PROTOCOL | 18-149 |
| C - | CHECKLIST FOR ANALYSIS OF DATA BUS HARDWARE AND SOFTWARE | 18-151 |
| BIBLIOGRAPHY | | 18-153 |
| GLOSSARY | | 18-169 |
| ACRONYMS AND ABBREVIATIONS | | 18-175 |



DTIC QUALITY INSPECTED 5

Accession For
NTIS GRA&I ☑
DTIC TAB ☐
Unannounced ☐
Justification
By
Distribution/
Availability Codes
Dist | Avail and/or Special
A-1

LIST OF ILLUSTRATIONS


| Figure | | Page |
|---|---|---|
| 2.1-1 | DATA BUS COMPONENTS | 18-5 |
| 2.1-2 | UNIDIRECTIONAL BUS ARCHITECTURE | 18-6 |
| 2.1-3 | AVIONIC SYSTEM USING UNIDIRECTIONAL BUSES | 18-6 |
| 2.1-4 | BIDIRECTIONAL BUS ARCHITECTURE | 18-7 |
| 2.1-5 | AVIONIC SYSTEM USING A BIDIRECTIONAL BUS | 18-7 |
| 2.1-6 | BIDIRECTIONAL BUS ARCHITECTURE, CENTRAL CONTROL | 18-8 |
| 2.1-7 | BIDIRECTIONAL BUS ARCHITECTURE, DISTRIBUTED CONTROL | 18-8 |
| 5.1-1 | COMMON DATA BUS TOPOLOGIES | 18-42 |
| 5.1-2 | LINEAR DATA BUS TOPOLOGIES | 18-43 |
| 5.1-3 | GATEWAY AND BRIDGE USED IN AVIONIC SYSTEMS | 18-46 |
| 5.1-4 | PERIODIC ACCESS FOR THREE BUS USERS | 18-51 |
| 5.1-5 | APERIODIC ACCESS FOR THREE BUS USERS | 18-51 |
| 5.1-6 | HDLC FRAME FORMAT | 18-53 |
| 5.1-7 | ASCB FRAME FORMAT | 18-54 |
| 5.2-1 | DATA BUS HARDWARE-SOFTWARE INTERFACE | 18-83 |
| 5.2-2 | SHARED INTERFACE RAM | 18-85 |
| 5.2-3 | DATA FRAMING | 18-88 |
| 5.2-4 | INPUT VOTING | 18-93 |
| 5.2-5 | OUTPUT VOTING | 18-94 |
| 5.2-6 | SELF-CHECKING PAIRS | 18-100 |
| 5.3-1 | STATE MACHINE | 18-105 |
| 5.3-2 | COUPLED STATE MACHINES | 18-106 |
| 5.3-3 | PETRI NET WITH FOUR STATES AND FOUR TRANSITION BARS | 18-107 |
| 5.3-4 | ACCESS PROTOCOL OVERVIEW FOR ARINC 629 BUS | 18-112 |
| 5.4-1 | TYPICAL FAULT TREE | 18-125 |
| 5.4-2 | QUANTITATIVE FAULT TREE ANALYSIS | 18-126 |
| 5.4-3 | HAZARD ANALYSIS WORKSHEET HEADER | 18-131 |

LIST OF TABLES


| Table | | Page |
|---|---|---|
| 2.2-1 | CURRENT AVIONIC DATA BUSES | 18-10 |
| 2.2-2 | NEW AVIONIC DATA BUSES | 18-11 |
| 2.3-1 | DATA BUSES, LISTED BY AIRCRAFT | 18-12 |
| 4.1-1 | FEDERAL AVIATION REGULATIONS APPLICABLE TO DIGITAL AVIONIC SYSTEMS | 18-20 |
| 4.1-2 | ADVISORY CIRCULARS APPLICABLE TO DIGITAL AVIONIC SYSTEMS | 18-23 |
| 5.1-1 | LTPB CHARACTERISTICS | 18-57 |
| 5.1-2 | HSRB CHARACTERISTICS | 18-58 |
| 5.1-3 | LTPB MESSAGE CHARACTERISTICS | 18-68 |
| 5.1-4 | LTPB MESSAGE PRIORITIES | 18-68 |
| 5.1-5 | HSRB MESSAGE LENGTH VERSUS INFORMATION WORDS | 18-70 |
| 5.1-6 | HSRB EFFICIENCY VERSUS INFORMATION WORDS | 18-71 |
| 5.2-1 | BUS INTERFACE UNIT INTEGRATED CIRCUITS | 18-84 |
| 5.2-2 | DATA BUS HARDWARE-SOFTWARE INTERACTION PROBLEMS | 18-86 |
| 5.3-1 | PROTOCOL SPECIFICATION GUIDELINES | 18-103 |
| 5.4-1 | INTEGRATION STANDARDS AND GUIDELINES, BY BUS (2 PARTS) | 18-115 |
| 5.4-2 | INTEGRATION TECHNIQUES DOCUMENTS | 18-119 |
| 5.4-3 | FMEA QUALITATIVE ANALYSIS REPORT | 18-127 |
| 5.4-4 | FMEA QUANTITATIVE ANALYSIS REPORT | 18-127 |
| 5.4-5 | FMECA ANALYSIS REPORT | 18-128 |
| 5.4-6 | SYSTEM SAFETY ANALYSIS METHODOLOGY | 18-130 |

# 1. INTRODUCTION

## 1.1 Background

Fixed and rotary wing civilian aircraft have used digital flight control and avionic systems since the late 1960s. One of the earliest digital systems was the Inertial Navigation System. Subsequently, other digital systems were added (Spradlin 1983). As multiple systems were introduced into aircraft there arose a need for digital communications between systems. In the early 1970s, many different digital data bus designs were used to provide this communication. Because these digital systems proved to be reliable and cost effective, their popularity increased.

Proliferation led to standardization, particularly in the air transport category of aircraft. In 1976, the air transport industry approved the Aeronautical Radio, Incorporated, (ARINC) Mark 33 Digital Information Transfer System (DITS) for digital data bus communications between Line Replaceable Units (LRUs) that conformed to the ARINC 500-Series Equipment characteristics. In the early 1980s, the General Aviation (GA) industry began using two data bus standards unique to its requirements.

Standardization of digital communications allowed communications between LRUs to become more complex. LRUs began to rely more heavily on each other to reduce the amount of equipment required. Sensor data and systems data could be shared among multiple systems, rather than each system requiring its own private source. The tighter coupling of systems led to the introduction of systems that were previously too complex or too cumbersome to produce. Complete Automatic Flight Control and Flight Management systems were implemented. Cockpits produced in the 1980s consisted of flight control electronics and avionics composed primarily of digital systems.

Although today's aircraft primarily use digital systems, the issue of whether digital systems can be relied upon for the safety of the aircraft, crew, and passengers has been avoided. Modern aircraft are certificated as safe for air transport use based on the assumption that any computer system may fail without producing a life threatening hazard. This is true because modern aircraft continue to rely on conventional mechanical, hydraulic, and analog electronic back-up systems to provide the minimum performance necessary to ensure safe flight and landing.

Civilian aircraft presently being developed can no longer be certificated on this basis. Complex digital systems are being used to implement essential and critical functions that cannot be sufficiently reproduced by conventional means. The X-29 military aircraft, with forward swept wings, is an example of what lies ahead for commercial aircraft. This aircraft is an inherently unstable design that requires computer control to keep it stable; a pilot could not fly it by standard means. It would be pointless to provide conventional back-up systems.

The safety of such aircraft is highly dependent upon the computer software, hardware, and the data buses connecting the systems. These aspects of digital systems have undergone, individually, much study and improvement over the years. The newest concerns relate to the problems that are unique to complex, highly integrated, systems. In particular, the modern bidirectional data buses will be heavily relied upon, yet at the same time, become more complex. There is no standard with which to assess the possible impact of these bus-based systems on aircraft safety. These and other advanced flight control and avionic systems will result in specific safety assessment problems when the appropriate data packages are submitted to the Federal Aviation Administration (FAA) during the certification process.

## 1.2 Scope

This handbook chapter addresses the concerns related to reliable communication on the serial digital data buses used to integrate digital systems in civilian aircraft. The reliability needed for buses used in essential and critical systems is particularly addressed. The communication on the parallel backplane buses used within LRUs is not addressed. Topics discussed include the following:

- The process followed by the FAA to certify that aircraft digital systems are safe.
- The formal and informal regulations that aircraft digital systems must satisfy.
- Safety concerns related to system integration based on current avionic data bus standards for air transport and GA aircraft.
- Safety concerns related to system integration based on new avionic data bus standards for air transport and GA aircraft.
- How data bus software-hardware interaction relates to aircraft safety.
- Data bus protocol specification and verification methods for ensuring proper operation.
- The extent to which data bus integration is controlled by data bus standards.
- Safety lessons that can be learned from current and new avionic data bus standards for military aircraft.
- The relationship of data bus standards to the certification process and regulatory standards.

This handbook chapter is provided to serve as a guide to Certification Engineers (CEs). It should help the CEs evaluate the material submitted for review when

they are asked to approve bus-integrated systems. For additional details on specific avionic standards and protocols, see the technical report, "Avionic Data Bus Integration Technology" (Elwell et al. 1992).

# 2. BUS-INTEGRATED AVIONIC SYSTEMS

## 2.1 Avionic System Architectures

An avionic system may perform a major cockpit function, like flight control, flight management, navigation, communications, autopilot, or autoland. Each system consists of a suite of electronic units that each perform a particular function needed by the system. These electronic units are usually called LRUs. (Entire systems are not considered replaceable units under routine maintenance.) LRUs typically transfer digital information among themselves and other systems on serial data buses. Each LRU, or bus user, usually consists of a host Central Processing Unit (CPU) interfaced to the bus by a Bus Interface Unit (BIU). The configuration is shown in figure 2.1-1.

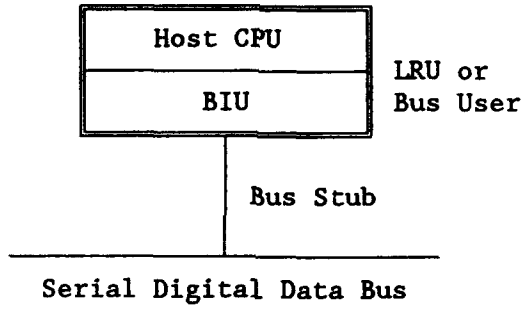



FIGURE 2.1-1. DATA BUS COMPONENTS

There are two primary types of bus-integrated avionic systems: those based on unidirectional data buses and those based on bidirectional data buses. A typical unidirectional bus architecture is shown in figure 2.1-2. The transmitting LRU controls the bus protocol and provides the bus message data. The protocol is very simple; it primarily consists of a standard message format. When the transmitting LRU broadcasts its messages onto the data bus, each of the other LRUs connected to the bus monitors the broadcast messages in order to detect and read the messages required.

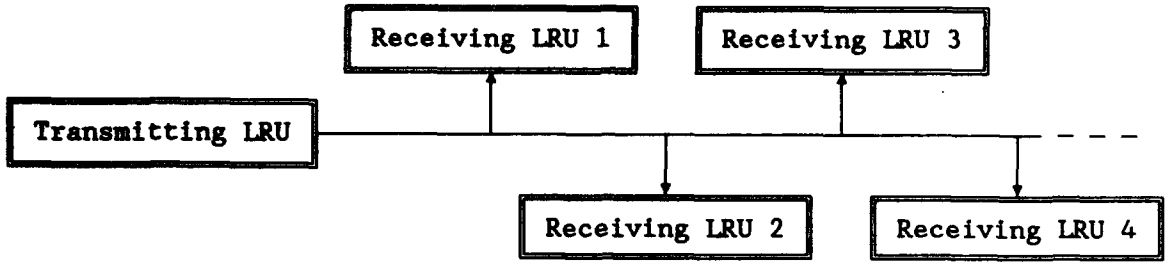



FIGURE 2.1-2. UNIDIRECTIONAL BUS ARCHITECTURE

When unidirectional data buses are used to integrate a system, the bus network is usually complex and requires large amounts of wire. Every LRU that needs to transmit data must have a unique data bus for its messages. Each LRU may need to have several bus interfaces to receive messages from multiple buses. For example, the navigation system shown in figure 2.1-3 requires five buses.

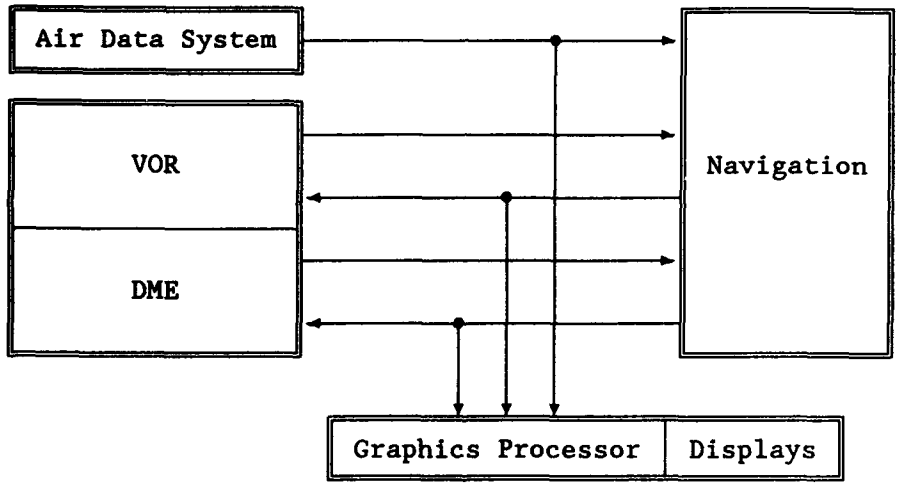



FIGURE 2.1-3. AVIONIC SYSTEM USING UNIDIRECTIONAL BUSES
(Hitt 1986)

Since each required message is made available by a direct connection, a system level design of the data bus network is unnecessary. The final bus network in an aircraft could be simply the configuration that results after every LRU has individually satisfied its information requirements.

A typical bidirectional data bus architecture is shown in figure 2.1-4. All LRUs can transmit and/or listen on one bus. Messages are time multiplexed. Each LRU only needs to have one bus interface and the bus network is reduced to a single data bus.

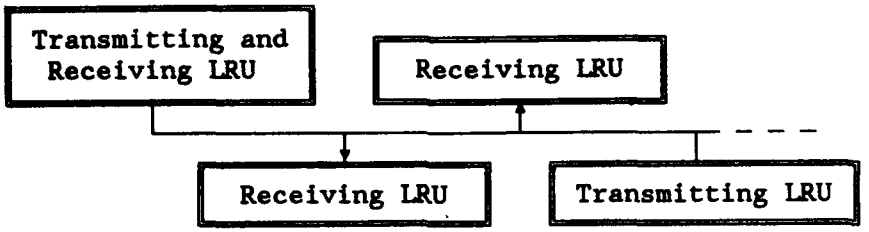



FIGURE 2.1-4. BIDIRECTIONAL BUS ARCHITECTURE

When bidirectional data buses are used, the physical network is usually simple, as shown in figure 2.1-5. On the other hand, the bus control is quite complex. The protocol must not only provide standard messages, but also arbitrate data bus transmissions to ensure that only one LRU transmits at a time and that listeners are listening at the proper time. The communication for LRUs integrated into a single system by a bidirectional data bus requires a system level design for successful operation. If each LRU attempted to independently satisfy its information requirements, the bus communications would never work.

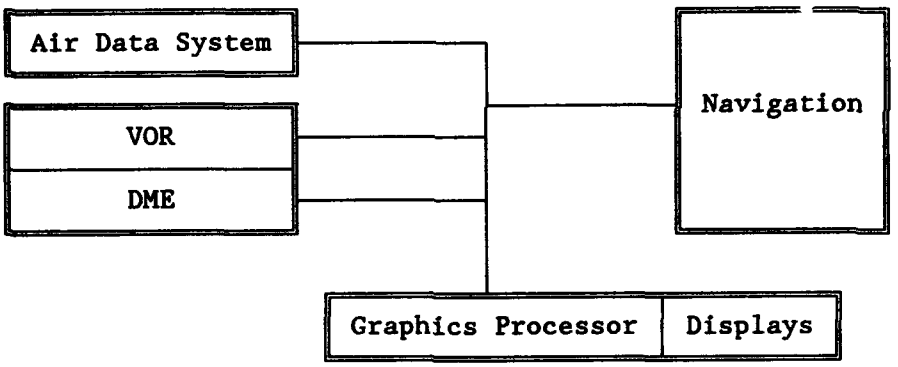



FIGURE 2.1-5. AVIONIC SYSTEM USING A BIDIRECTIONAL BUS

Because bus control is much more complex for bidirectional data buses, many different architectures may be employed for bus control. The two fundamental approaches in these architectures are central and distributed control. Figure 2.1-6 shows the bus control provided by a central Bus Controller (BC). The BIU portion of each LRU is explicitly shown.

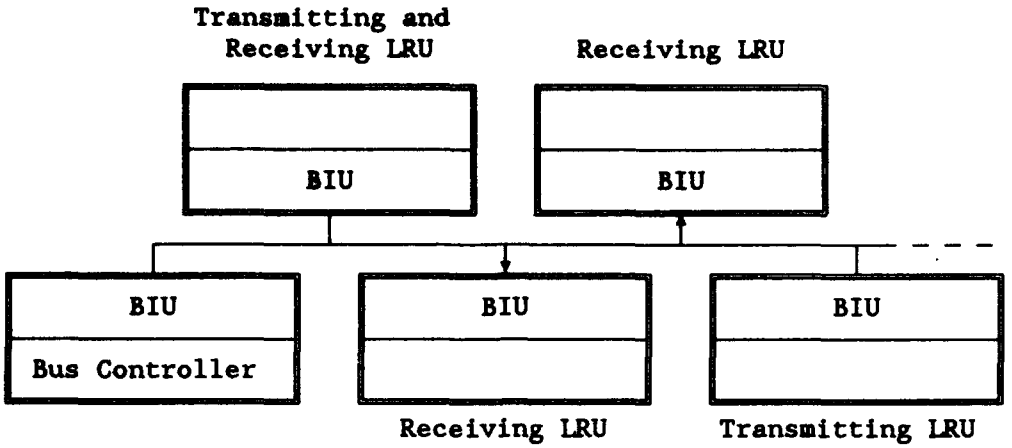



FIGURE 2.1-6. BIDIRECTIONAL BUS ARCHITECTURE, CENTRAL CONTROL

The main advantage of central bus control is that only one bus component ever has control of the bus operation. All data bus users can only use the bus as directed by the BC. The controller can be a tightly coupled system, with minimal interaction with outside influences. Another advantage is that when the data bus configuration changes, only the BC must be changed to support the new configuration. Other LRUs usually remain unaffected. Furthermore, system integration issues are necessarily addressed explicitly when the BC is designed. The main disadvantage of a bus which is centrally controlled is that the BC represents a single point of failure. Advanced designs attempt to solve this problem by using redundant controllers and redundant data buses.

Figure 2.1-7 shows a bidirectional data bus that relies on distributed control. The BIU of each transmitting LRU must recognize when it is its turn to control the bus. It then transmits its messages and relinquishes control.

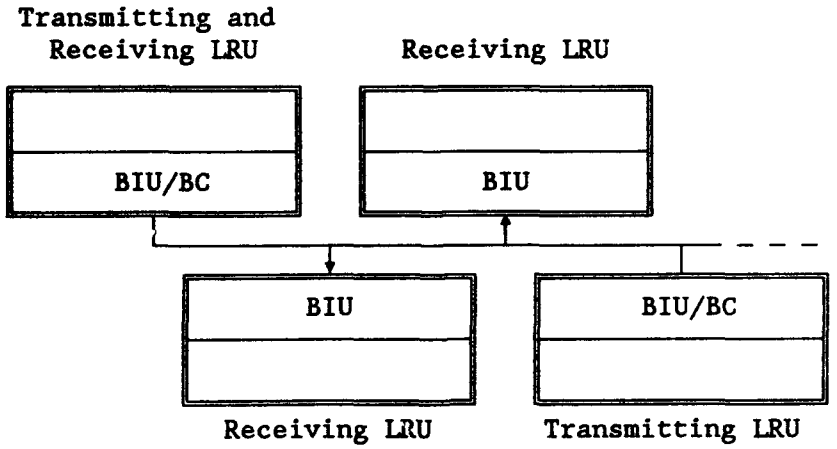



FIGURE 2.1-7. BIDIRECTIONAL BUS ARCHITECTURE, DISTRIBUTED CONTROL

Typically, a bus that uses distributed control has the primary advantage that, if an LRU controls the bus improperly, the remainder of the bus users can continue to communicate unaffected. However, distributed control is weak on the very points that are advantages for central control. Since every BIU is a BC, bus control must be coordinated among LRUs. Also, changes to the bus configuration may require a change to every BIU. Distributed control can cause the designer of a BIU to take a narrow approach, concentrating on bus control during the window available to the one LRU. System design becomes an independent task that must be delegated, rather than an inescapable task, as it is for central control.

The implications of these architectural variations for the safety of data bus-integrated systems is discussed in detail in subsequent sections.

## 2.2 Avionic Data Buses

Currently, three digital data buses predominate in civilian aircraft. One is used in the large transport aircraft and two in the smaller business and private GA aircraft.

Transport category aircraft primarily use the unidirectional data bus standardized by ARINC. It is defined in ARINC Specification 429, "Mark 33 DITS" (1990). Data on the Mark 33 DITS are transmitted, at a bit rate of either 12.5 or 100 kilobits per second, to up to 20 LRUs monitoring the bus messages. Nearly every transport aircraft has a large network of ARINC 429 data buses connecting avionics within and between the major systems.

GA aircraft use the unidirectional Commercial Standard Data Bus (CSDB), developed by the Collins General Aviation Division of Rockwell International, and the bidirectional Avionics Standard Communications Bus (ASCB), developed by Honeywell, Incorporated. The bus used in a particular aircraft is determined by which company the airframe manufacturer chooses to supply the avionics. Both companies are major contributors to avionics today. However, in 1989, only about one-third of the GA fleet used guidance and control avionics that likely used data buses ("Avionics Market Data," 1991).

A CSDB can be either a low- or high-speed bus. Data are transmitted at a bit rate of 12.5 kilobits per second on a low-speed bus and 50 kilobits per second on a high-speed bus. Up to 10 receivers can be attached to one bus.

The ASCB is a centrally controlled, bidirectional bus. The basic configuration consists of one BC directing the operation of two, otherwise isolated, buses. Each bus can support up to 48 users. Data are transferred at a bit rate of two-thirds of a megabit per second. LRUs may transmit on one bus and listen to either bus. This isolation allows less critical systems to receive data from more critical systems without being able to affect their operation. The BC synchronizes the activity of the LRUs on both buses. The ASCB pair may also be fitted with a standby controller whose operation is coordinated with the active controller.

In military aircraft, one data bus predominates. Since about 1970, military aircraft have used the MIL-STD-1553 Digital Time Division Command/Response

Multiplex Data Bus. Because the bus has been used extensively for so long, and in critical systems, many important lessons have been learned that should be applied to data buses used in civilian aircraft. This data bus is being fully relied upon in fly-by-wire aircraft, like the X-29. It has found its way into civilian aircraft only in isolated cases.

The MIL-STD-1553 data bus is a bidirectional, centrally controlled data bus. This bus can support 31 users and data are transmitted at a bit rate of 1 megabit per second. Many implementations use it in a dual, fully redundant, configuration. All activity can be replicated on either bus since each bus is controlled by identical controllers.

A fiber optic implementation of the MIL-STD-1553 bus has been defined. It is the MIL-STD-1773 (1983) bus. It has not been used in commercial aircraft.

The predominant data buses in use are summarized in table 2.2-1. These buses are analyzed in this chapter with regard to their use in integrating digital systems.

TABLE 2.2-1. CURRENT AVIONIC DATA BUSES

| Data Bus | Usage |
|---|---|
| ARINC Specification 429-12, "Mark 33 Digital Information Transfer System (DITS)" | Air Transport |
| Commercial Standard Data Bus (CSDB) | General Aviation |
| Avionics Standard Communications Bus (ASCB) | General Aviation |
| MIL-STD-1553, "Digital Time Division Command/Response Multiplex Data Bus" | Military |
| MIL-STD-1773, "Fiber Optics Mechanization of an Aircraft Internal Time Division Command/Response Multiplex Data Bus" | Military |

Some recent experimental air transport aircraft have used a new data bus developed by the Boeing Commercial Airplane Company (BCAC). The BCAC version is known as the Digital Autonomous Terminal Access Communication (DATAC) data bus. This bus has been made an air transport standard under ARINC Specification 629, Part 1 (1990). It will be used in the Airbus 340 and Boeing 777, as well as subsequent air transports.

The ARINC 629 bus is a bidirectional bus utilizing distributed control. This bus can support up to 120 users. Data are transmitted at a bit rate of 2 megabits per second. It supports the higher data rate and large message

transfers needed in highly integrated digital systems. It is intended that this bus will be relied upon in essential and critical systems.

Two other data buses are being developed and standardized, primarily for military aircraft. They are targeted to be the primary buses used in military aircraft, replacing the MIL-STD-1553 bus. Because they are very high-speed buses, they may also find application in civilian aircraft that require a greater data bus throughput than an ARINC 629 bus can supply. These buses are the Society of Automotive Engineers (SAE) AS4074.1 Linear Token Passing Bus (LTPB) and the AS4074.2 High Speed Ring Bus (HSRB). Both transfer data at a bit rate of 50 megabits per second. They are multi-transmitter buses that operate under distributed control. Messages can be sent bidirectionally, but not in the conventional sense.

The LTPB is a linear bus and bus users can either transmit or receive, but messages are passed in a logical ring. The HSRB is configured in both a physical and logical ring. Bus users can either transmit or receive, but messages are passed around the ring until they reach their destination.

The prominent data buses being newly used or developed are summarized in table 2.2-2. These buses are also analyzed in this chapter with regard to their use for integrating digital systems.

TABLE 2.2-2. NEW AVIONIC DATA BUSES

| Data Bus | Usage |
|---|---|
| ARINC Specification 629, "Multi-Transmitter Data Bus" | Air Transport |
| SAE AS4074.1, Linear Token Passing Bus (LTPB) | Military |
| SAE AS4074.2, High Speed Ring Bus (HSRB) | Military |

## 2.3 Aircraft Implementations

This section gives a sample of the mix of data buses and the aircraft in which they are installed. The list in table 2.3-1 is not comprehensive.

TABLE 2.3-1. DATA BUSES, LISTED BY AIRCRAFT

| Aircraft | Data Bus | Reference |
|---|---|---|
| Airbus A310/A320 | ARINC 429 | Shaw and Sutcliffe 1988<br>Clifton |
| Airbus A330/A340 (being developed) | ARINC 629 | ARINC Specification 629 Part 3, 1989 |
| Bell Helicopter | ARINC 429 | Clifton |
| Boeing 727 | CSDB has been used in retrofits | Rockwell International (Collins Division) |
| Boeing 737 | ARINC 429<br><br>DATAC was retrofitted to the NASA TSRV 737<br><br>CSDB has been used in retrofits | Clifton<br><br>Shaw and Sutcliffe 1988<br>Holmes 1986<br><br>Rockwell International (Collins Division) |
| Boeing 747 | ARINC 429 | Clifton |
| Boeing 757 | ARINC 429 | Shaw and Sutcliffe 1988 |
| Boeing 767 | ARINC 429 | Shaw and Sutcliffe 1988 |
| Boeing 777 (being developed) | ARINC 629 | Bailey 1990 |
| Cessna Citation | ASCB | FAA, Atlanta ACO |
| Dassault Falcon 900 | ASCB | FAA, Atlanta ACO |
| DeHavilland-8 | ASCB | FAA, Atlanta ACO |
| Gulfstream IV | ASCB | FAA, Atlanta ACO |
| McDonnell-Douglas DC-8 | CSDB has been used in retrofits | Rockwell International (Collins Division) |
| McDonnell-Douglas MD-11 | ARINC 429 | Spitzer[1] 1986 |

## 3. CERTIFICATION PROCEDURES FOR BUS-INTEGRATED SYSTEMS

Certification is the process of obtaining FAA approval for the design, manufacture, and/or sale of a part, subsystem, system, or aircraft, by establishing that it complies with all applicable government regulations. The purpose of certification is to demonstrate and record that the total aircraft is suitable and safe for civilian use. The FAA does this by requiring that aircraft products (aircraft, engines, and propellers) be Type Certificated (TCed). Major avionic systems that are to be manufactured for use in an aircraft are certificated individually under an aircraft type certification program. The requirements for the certification of avionic systems are covered in the Federal Aviation Regulations (FARs), as follows:

- Part 21, "Certification Procedures for Products and Parts"
- Part 23, "Airworthiness Standards: Normal, Utility, and Acrobatic Category Airplanes"
- Part 25, "Airworthiness Standards: Transport Category Airplanes"
- Part 27, "Airworthiness Standards: Normal Category Rotorcraft"
- Part 29, "Airworthiness Standards: Transport Category Rotorcraft"
- Part 33, "Airworthiness Standards: Aircraft Engines"
- Part 91, "General Operating and Flight Rules"
- Part 121, "Certification and Operation: Domestic, Flag, and Supplemental Air Carriers and Commercial Operators of Large Aircraft"
- Part 135, "Air Taxi Operators and Commercial Operators of Small Aircraft"

Data buses, on the other hand, are not explicitly certificated because they have been viewed simply as the connectors of the systems. Certification procedures need to be expanded to include reviews and tests for data buses used by digital systems.

### 3.1 Types of Certification

There are two approaches to approving an avionic system, depending on whether the system is an original design or an independent design of a previously approved product. When a major design effort is required to develop a system, the integrity of the aircraft into which it will be installed is in question. Thus, one of two "type certification" processes must be followed to receive a certificate. For totally new designs, or changes that are so extensive as to require a complete reinvestigation of the design, the developer must follow the

process required to obtain a TC for the aircraft. For major changes (as defined in FAR Part 21, section 93) to a system previously approved under a TC, the developer can follow a simpler process to obtain a Supplemental Type Certificate (STC). In either case, after the certificate is issued, the manufacturer may also obtain a Production Certificate approval to manufacture additional systems, whose type certification is based on conformity to the type design rather than tests of each system.

When a manufacturer wishes to produce modification or replacement parts (i.e., parts not previously approved by a TC or an STC) for sale or installation on a TCed aircraft, simpler approvals are sufficient. The manufacturer who holds the TC or STC for the design can request an amendment to their certificate. A manufacturer who wishes to produce such a part for an aircraft, but does not hold the TC or STC, can obtain a Parts Manufacturer Approval (PMA). Such a second party manufacturer can also apply for a Technical Standard Order (TSO) Authorization. The FAA publishes TSOs that establish the minimum performance requirements for such interchangeable parts. Any manufacturer can obtain this specification and build a part that satisfies it. If the manufacturer is given a TSO Authorization, the parts may be stamped with the TSO number, showing compliance with the requirements of the TSO. The parts can then be legally sold or installed in aircraft. It is the responsibility of the installer to ensure that they are used in an application that does not exceed performance requirements.

The system to be certificated can be a component or several components. It can be simple or complex. The FARs stipulate which process must be followed in each case. Although the manufacturer may refer to the FARs to decide which approval should be sought, often a CE recommends which application the manufacturer should submit. The authority for determining whether a change constitutes a modification or a redesign, and whether a redesign is minor or major, rests with the Aircraft Certification Office (ACO).

By way of example, the all new Boeing 777 is being developed under a TC program. On the other hand, the FAA has required that the entire fleet of commercial aircraft (all aircraft that operate under FAR Part 121 rules) be retrofitted with a Traffic Alert and Collision Avoidance System (TCAS). This system was developed under STC programs.

Whenever an applicant presents a situation that is not covered by the existing rules, the ACO can request direction from the Directorate by using an Issue Paper. The Directorate may need to rule on an issue because the applicant believes they are in compliance, but the ACO does not. In this case, the Directorate gives their conclusion on the Issue Paper, sustaining the ACO's position, overruling the ACO's position, or presenting an alternative position.

The ACO submits an Issue Paper when an applicant claims to provide an equivalent level of safety by means other than provided in the regulations. If the claim is substantiated, the FAA Directorate issues a Finding of Equivalent Safety.

An Issue Paper may also be presented when an applicant's design is sufficiently new that no regulation seems to apply. In this case, the request for action by the Directorate results in a Special Condition (SC) being issued.

## 3.2 Conducting Certification Testing

In the past, to certificate an airplane, inspectors and engineers had to understand avionics based on analog electronic systems driving mechanical, pneumatic, and/or hydraulic systems. Today's digital systems are more complex. Data buses within these systems perform their own functions and could be considered separate systems, not simply wires. Existing requirements do not cover the expanded functions that data buses perform.

The environmental tests described in "Environmental Conditions and Test Procedures for Airborne Equipment" (RTCA/DO-160C, 1989) address electronic component tests, such as magnetic effects, voltage spikes, and induced voltages. These general tests can be performed on any electronic component. For example, the ARINC 429 bus has been subjected to these tests because it has been used to connect electronic components. While these tests are necessary, they are not sufficient. Bidirectional data buses require new tests that should be addressed in RTCA/DO-178. The ASCB, for example, allows signals to be both transmitted and received over the same wire. This two-way communication requires complex digital electronics to control bus transmissions. BCs, software in the controllers, and protocols must now be tested.

The FAA relies on the manufacturer to conduct testing. If the FAA adopts new bus tests, the manufacturer must comply with them. In general, to show compliance with the FARs, a component must be subjected to environmental tests, software tests, and failure analyses.

For certificating systems containing data buses, the manufacturer should test the bus to ensure that it is reliable and performs its intended function. If the bus relies on a back-up system, it also should be tested.

FAR Part 25, section 1309, shows the objectives the tests are designed to meet for transport category airplanes. The airplane systems and associated components considered separately and in relation to other systems must be designed to ensure that the following conditions are met:

- The occurrence of any failure condition which would prevent the continued safe flight and landing of the airplane is extremely improbable.

- The occurrence of any other failure condition which would reduce the capability of the airplane or the ability of the crew to cope with adverse operating conditions is improbable.

For electrical systems and equipment design (and thus for the subset of digital avionic systems containing data buses) critical environmental conditions must be considered. Digital avionic equipment must comply with this section, unless the equipment is already covered by TSOs that include environmental test procedures and test designs for meeting the two requirements above.

Section 4 addresses related standards for testing, as provided in the Advisory Circulars (ACs) and SCs published by the FAA. These documents address testing for lightning and High Intensity Radiated Frequency (HIRF) susceptability.

### 3.2.1 Approaches to Bus Reliability

For certificating systems that will be used for flight-critical functions, designers can take one of two approaches. The first approach, "safe-life," means the component is designed to keep its strength and integrity throughout its life. The second method, "fail-safe," means safety is assured by having a redundant or back-up part that will work if the first component fails (Improving Aircraft Safety, 1980).

The fail-safe approach has been adopted for digital avionic systems containing data buses. Because of the risk of applying complex technology to critical or essential functions, data buses used to support either category usually consist of a pair of redundant buses, and the entire digital system has a back-up.

In the Gulfstream IV airplane, for example, the ASCB ties together a sophisticated navigation system that drives navigation displays and provides steering inputs to a digital autopilot. The ASCB is controlled by redundant BCs (Jennings 1986). For this aircraft, the controllers are built into the two fault warning computers. By design, if one computer fails the other takes control, so the system still operates correctly. The entire digital system is backed up by an electromechanical system.

Eventually, if redundancy can ensure that the probability of an unsafe event occurring is acceptable (not greater than $1x10^{-9}$ for critical functions), digital systems may supersede older ones and may not need mechanical back-ups. In the new F-16C and F-16D, the current Advanced Fighter Technology Integration F-16's triple-redundant digital computers, each with analog back-up, will be replaced by quadruple-redundant digital computers (Spitzer[1] 1986) without back-up. Pilots will rely solely on the digital systems in the cockpit. *Hence, redundancy* becomes more important, and testing the redundant systems to ensure that they will operate as intended becomes critical.

### 3.2.2 Testing Data Buses

A manufacturer who plans to use an existing data bus differently, or would like to certificate equipment that uses a new bus, should thoroughly document any new tests. For example, while the ASCB has been in the field for over a decade, it has not been used on flight-critical systems. Also, in many cases, it has not been used to its fullest capability, i.e., bidirectionally. This function may need to be tested during integration testing. When no regulations and standards exist to create the tests, the manufacturer must devise them and submit them in the test plan.

As data buses are being designed to carry more functions than in the past, low-level considerations, such as the message formats, become important. Also, depending on the architecture of the data bus, other components may need to be tested. For example, National Semiconductor is developing a bus controller Integrated Circuit (IC) that will be installed in the BIU of an ARINC 629 data bus user. The controller interfaces a linear, serial bus with a parallel, 16-bit subsystem bus. The manufacturer must develop tests for the IC using RTCA/DO-178A and RTCA/DO-160C for guidance. In addition to normal factory tests

of the IC, the ARINC 629 BIU, data buses, and connected equipment should all be tested as a system at a validation or simulation facility.

### 3.3 Certification Concerns

The TSO Authorization method of approving components was developed to allow manufacturers to substitute equivalent components "off-the-shelf" without jeopardizing the existing TC. Since a TSO Authorization request must be processed within 30 days and does not require integration testing, manufacturers use this method rather than Type Certification whenever possible.

In the days of simpler aircraft design, TSO Authorizations were adequate. Now, however, digital avionics systems are more complex and require involved integration testing procedures. In some cases, if a manufacturer substitutes one black box for another (by using the TSO method), the FAA risks having a system certificated with potential safety risks. While the new black box may function perfectly in a laboratory setting, it may not have the required protocol to interact effectively with the rest of the digital system. Hence, this failure could result in a system failure.

To improve the TSO approval process, ACOs are becoming more involved in approving new digital systems. They are reviewing Verification and Validation (V&V) plans for TSO packages, and are working more closely with the manufacturers. The ACOs are suggesting that system integration test plans be required for substitutions in integrated digital systems. Additionally, sections of RTCA/DO-178A addressing certification issues are being rewritten to address these integrated systems.

As data buses become more complex, the manufacturer must ensure that the data bus will function as intended within its operating environment. Manufacturers' validation facilities will play a greater role in establishing the requirements for integration testing, since the functions of digital avionic systems will be simulated there. The certification requirements will expand for such systems to reflect the FAA's concern that they safely perform their functions once the systems are installed in an aircraft.

## 4. RELATED REGULATIONS AND STANDARDS

The CE's job has become more complex due to rapid growth in the microelectronics industry. Breakthroughs in hardware and software technology have made it difficult for CEs to determine what avionic data bus standards are permissible. For example, the CE must ensure that both the data bus and the method for testing the bus (e.g., simulation, fault analysis) meet predetermined regulations.

Because few specific certification procedures exist, the CE has only a general approach for certificating new and upgraded digital data buses. As a result, the CE must consult many sources for certification information.

Fortunately, associations like the American Institute of Aeronautics and Astronautics (AIAA) and the Institute of Electrical and Electronics Engineers (IEEE) hold conferences and produce publications addressing certification issues. These publications often state requirements that a specific data bus should meet. Other articles presented by aircraft associations list standards, guidelines, and test procedures which may be adopted by individual manufacturers or federal agencies.

ARINC and the General Aviation Manufacturers Association (GAMA) publish data bus standards. They include descriptions of specific bus topologies and protocols. Subcommittees within these associations often publish guidelines that an avionic system manufacturer can follow, like ARINC Project Papers 617 (1990) and 651 (1990). Although these two guidelines have not been formally accepted by the FAA and are currently in draft form, manufacturers may refer to them for guidance during a system's design process.

Associations like the SAE and RTCA publish analysis and test procedures. They address failure analyses (SAE Aerospace Recommended Practice [ARP] 1834) and environmental testing (RTCA/DO-160C). These procedures are used by manufacturers to demonstrate their system's reliability and functionality.

Before the above standards are applied in certification, they are compared with federal regulations. The only regulations applicable to digital data buses and integrated avionic systems are the FARs, ACs, and SCs. The relevant FARs are Parts 23, 25, 27, 29, and 33, while ACs and SCs are means of showing compliance with the FARs.

### 4.1 Relevance of Formal Guidelines to Bus-Integrated Systems

The following sections present FARs applicable to the certification of data buses and integrated avionic systems. Additional FAR sections, which address HIRF requirements, are forthcoming. When they take effect, they should also be considered. ACs and SCs, and their relationship to the FARs, are then discussed.

### 4.1.1 Bus-Integrated Avionic Systems and Federal Aviation Regulations

FARs are published by the U.S. Government to regulate civil aviation activities. They range from Part 1, "Definitions and Abbreviations," to Part 189, "Use of Federal Aviation Communication Systems." Each FAR part is separated into sections. Within some of these sections are rules that avionic system manufacturers must follow during a system's design process.

Whether the system is used in an airplane or rotorcraft will determine which of the sections the system must satisfy. For example, if an avionic system is to be put in a normal category rotorcraft, it must satisfy FAR Part 29, section 1309. On the other hand, if a system is to be installed in a transport category airplane, it must satisfy FAR Part 25, section 1309. Table 4.1-1 shows which sections within these FARs should be considered during an avionic system's certification. Additional sections, which address HIRF requirements, are forthcoming. When they take effect, they should also be considered.

TABLE 4.1-1. FEDERAL AVIATION REGULATIONS APPLICABLE TO DIGITAL AVIONIC SYSTEMS

| Regulation | Title |
|---|---|
| FAR 23.1309 | Equipment Systems and Installations |
| FAR 23.1431 | Electronic Equipment |
| FAR 25.581 | Lightning Protection |
| FAR 25.1309 | Equipment Systems and Installations |
| FAR 25.1431 | Electronic Equipment |
| FAR 27.610 | Lightning Protection |
| FAR 27.1309 | Equipment Systems and Installations |
| FAR 29.610 | Lightning Protection |
| FAR 29.1309 | Equipment Systems and Installations |
| FAR 29.1431 | Electronic Equipment |
| FAR 33.75 | Safety Analysis |
| FAR 33.91 | Engine Component Tests |

FAR Parts 23, 25, 27, and 29, section 1309, require that systems and equipment be designed to perform their intended functions under any foreseeable operating

conditions. These sections also address failure conditions by defining how many failures are allowed throughout a specified time period. A failure is any condition that could inhibit the continued safe flight and landing of the aircraft. As stated in section 23.1309:

> "The occurrence of any failure condition that would prevent the continued safe flight and landing of the aircraft must be extremely improbable,"

and

> "the occurrence of any other failure condition that would reduce the capability of the aircraft or the ability of the crew to cope with adverse operating conditions is improbable."

An AC provides the failure rates for these requirements. Extremely improbable failures have a probability of $1x10^{-9}$ or less. Improbable failures have a probability of $1x10^{-5}$ or less, but greater than $1x10^{-9}$.

FAR Parts 25 and 29, section 1309, state similar requirements for their related aircraft. FAR Part 27, section 1309, however, does not go into as much detail; this section merely states that "the equipment, systems, and installations of a multi-engine rotorcraft must be designed to prevent hazards to the rotorcraft in the event of a probable malfunction or failure," and "equipment, systems, and installations of a single-engine rotorcraft must be designed to minimize hazards to the rotorcraft in the event of a probable malfunction or failure."

These requirements have a direct impact on the design of data buses and avionic equipment because the manufacturer must develop a scheme to satisfy them. Usually, manufacturers employ laboratory, ground, flight, and simulator tests to meet section 1309.

FAR Parts 23, 25, 27, and 29, section 1309, also contain short statements of how to comply with certain requirements in those FARs. Section 25.1309 states that one must use environmental tests to evaluate the electrical system's design and installation, except when the component is authorized under a TSO. Part 23 also states that environmental testing must be used for compliance, and additionally, that it should include analyses for radio frequency (RF) energy and lightning effects. In addition to environmental, laboratory, ground, flight, and simulator tests, manufacturers can show compliance by referencing previous comparable service experience on other aircraft.

FAR Parts 23, 25, and 29, section 1431, discuss requirements for electronic equipment. In sections 23.1431 and 29.1431, the requirements deal strictly with radio communication and navigation systems. FAR section 25.1431, however, discusses radio and electronic equipment. This particular section says that "radio and electronic equipment, controls, and wiring must be installed so that the operation of any one unit or system of units will not adversely affect the operation of any other radio or electronic unit, or system of units".

FAR Part 25, section 581, also must be considered during the avionic system design process. It expresses a need for lightning protection, and is more

specific than FAR Part 23, section 1309. Part 25, section 581, states that equipment should be designed so that a lightning strike will not endanger the aircraft. It also suggests eliminating the threat of lightning damage by diverting the electrical current. FAR Parts 27 and 29, section 610, describe lightning requirements for transport and normal category rotorcraft. Both parts recount the same requirements as FAR Part 25, section 581.

Electronic Engine Controls (EECs) using data buses are addressed differently. FAR Part 33, section 75, requires a safety analysis to determine that no probable failure or improper operation of an engine can cause an engine to catch fire, burst, generate excessive loads, or lose its ability to be shut down. EECs certainly require this analysis. Furthermore, section 33.91a requires additional tests for those components for which reliable operation cannot be adequately substantiated by the endurance tests of section 33.82. The FAA has followed the recommended Notice of Proposed Rulemaking, No. 85-6 (1985), as guidance for the safety analysis and tests of EECs, including Full Authority Digital Electronic Controls.

All systems which employ data buses and avionic equipment are subject to these requirements. However, no test or design procedures for data buses or integrated avionic equipment are directly mentioned in the FAR sections listed in table 4.1-1.

### 4.1.2 Bus-Integrated Avionic Systems and Advisory Circulars

To assist the manufacturer in meeting the requirements of certain FAR sections, the FAA publishes ACs. ACs address specific sections of the FARs, and "describe various acceptable means for showing compliance" with the FARs (AC 25.1309-1A, 1988). The ACs are not mandatory; manufacturers may opt to meet the FARs by different means. This decision, however, requires that the manufacturer's techniques be validated by the FAA.

Table 4.1-2 shows ACs which may be used to help manufacturers comply with the FAR sections listed in table 4.1-1. A new AC is being developed which is also of interest: AC-XX-XX, "Certification of Aircraft Electrical/Electronic Systems for Operation in the High Intensity Radiated Fields (HIRF) Environment" (1991). A user's manual will accompany the AC ("User's Manual for AC-XX-XX," 1992). A similar user's manual is being developed for AC-20-136.

AC 20-115A describes how RTCA/DO-178A is used in connection with TSO, TC, and STC authorizations. The AC says that since future avionic equipment will rely heavily on software and microcomputer techniques, a manufacturer may use RTCA/DO-178A to secure approval of computer software. The AC also says that if other ACs, which better outline the relationship between the criticality level and the software level, are published by the FAA, those ACs take precedence over RTCA/DO-178A. RTCA/DO-178A's primary use is to satisfy FAR Parts 21, 23, 25, 27, 29, and 33.

TABLE 4.1-2. ADVISORY CIRCULARS APPLICABLE TO DIGITAL AVIONIC SYSTEMS

| Advisory Circular | Title |
|---|---|
| AC 20-115A | RTCA/DO-178A |
| AC 20-136 | Protection of Aircraft Electrical and Electronic Systems Against the Indirect Effects of Lightning |
| AC 21-16C | RTCA/DO-160C |
| AC 23.1309-1 | Equipment, Systems, Installations in Part 23 Airplanes |
| AC 25.1309-1A | System Design and Analysis |

To help manufacturers satisfy all FAR Parts that address the need for lightning protection, AC 20-136 was published, and an accompanying user's manual is under development. The AC describes how a manufacturer can cope with the hazards inherent in a lightning environment. Methods pointed out by the AC include the following:

- Determining the lightning strike zones for the aircraft
- Establishing the external lightning environment for the zones
- Establishing the internal lightning environment
- Establishing transient control and design levels

Manufacturers who wish to achieve compliance with the FAA's lightning requirements should begin by submitting a certification plan to the appropriate ACO. An outline and explanation for the lightning effects certification plan are presented on pages 5, 6, and 7 of AC 20-136. Once the plan is approved, the manufacturer may begin analysis. Since RTCA/DO-160C contains test criteria for evaluating the indirect effects of lightning, it may be employed in this step.

AC 21-16C describes how RTCA/DO-160C is used in conjunction with TSO authorizations. RTCA/DO-160C describes environmental test procedures that can be used to satisfy AC 25.1309-1A and AC 23.1309-1. RTCA/DO-160C also satisfies criteria presented in FAR Part 25, section 1309. Since data buses and related digital equipment are sometimes certified within a TSO, this document can be applied during a certification procedure. No procedures or guidelines are pointed out in this document; it only states that RTCA/DO-160C should be considered.

AC 25.1309-1A describes design procedures and failure analyses for meeting the requirements of FAR section 25.1309. Techniques such as redundancy, isolation,

and error tolerance improve the safety of the system (more techniques are listed on page 3 of the AC). Usually, at least two of these techniques are needed. Also included in AC 25.1309-1A is the FAA's Fail-Safe Design Concept, as follows:

> "In any system or subsystem, the failure of any single element, component, or connection during any one flight should be assumed. Such failures should not prevent continued safe flight and landing, or reduce the capability of the airplane or crew to cope with the resulting failure conditions." (AC 25.1309-1A, 1988).

Examples of failure condition analysis and design procedures are provided in appendix A of the technical report, Avionic Data Bus Integration Technology (Elwell et al. 1992).

The ultimate goal of AC 25.1309-1A is to ensure that all failure conditions for all systems are considered. AC 23.1309-1 discusses similar, scaled down procedures for meeting the requirements in FAR Part 23, section 1309.

4.1.3 Bus-Integrated Avionic Systems and Special Conditions

Requests for SCs are submitted to the FAA in accordance with FAR Parts 11 and 21. One purpose of an SC can be to supplement the FARs when the FARs do not explicitly define adequate safety measures for "novel and unusual design features on aircraft" (SC 23-ACE-49, 1990). This section does not discuss why SCs are adopted; it merely states what an SC is and gives examples of SCs which have been applied to integrated avionic equipment. SCs for one aircraft can be considered for another aircraft, if the other aircraft uses similar components or systems.

SCs which are published for integrated avionic systems usually do not mention data buses. However, because data buses can be a part of the system that requires the SC, buses are implicitly subject to the SC's criteria.

SC 25-ANM-35 (1990) includes two special conditions, each with two subparts, that concern the McDonnell-Douglas MD-11 aircraft. Following is a summary of each subpart:

- Lightning

  Each electronic system that performs flight-critical functions must be designed and installed to ensure that the operation of these functions is not affected when the airplane is exposed to lightning.

  Each essential function, carried out by new or modified electronic equipment, must be protected to ensure timely recovery of the function after a lightning strike.

  Systems that perform essential functions must be protected to ensure that failures, due to a lightning strike, will not result in an unacceptable cockpit crew workload.

- **Protection from Unwanted Effects of RF Fields**

  **Electronic systems that perform flight-critical functions must be designed and installed to ensure that the operation of these functions is not adversely affected when the airplane is exposed to High Energy Radio Frequency (HERF) fields.**

**SC 25-ANM-35 is meant to supplement the FARs because the FARs do not contain adequate safety standards for protection from lightning and the unwanted effects of RF fields.**

**To meet the requirements of SC 25-ANM-35, the MD-11 must undergo specific analyses for lightning and RF fields. (One such analysis is presented in RTCA/DO-160C, section 22.) A need for lightning effects analysis is pointed out in FAR Part 23, section 1309.**

**The subparts of SC 25-ANM-35, discussed above, can be indirectly applied to data buses. If the bus were to be exposed to lightning effects or RF fields it could lose data, produce erroneous data, or fail completely. The bus could also act as a path for current, and that current could adversely affect LRUs connected to the bus.**

SC 23-ACE-49 (1990) is similar to SC 25-ANM-35, and is published on the SOCATA Model TBM-700 Series aircraft. The TBM-700 aircraft is required to meet SC 23-ACE-49 because it contains an Electronic Attitude Director Indicator and an Electronic Horizontal Situation Indicator, in place of the original mechanical and electromechanical displays. SC 23-ACE-49 contains the same special conditions as SC 25-ANM-35, but adds a special condition which requires failure analysis.

SC 23-ACE-49 amends the FARs on the installation of electronic displays which could be adversely affected by a single failure or malfunction. It also provides requirements for verifying that these flight-critical systems are adequately designed.

This SC is similar to the one issued for the MD-11 airplane. The following special conditions are issued as part of the type certification basis for the SOCATA TBM-700 airplane:

- Electronic Flight Instrument Display (EFID)

  The systems using EFIDs must be examined separately, and in relation to other airplane systems, to determine if the airplane is dependant on the system's function for safe flight and landing. If so, the system must satisfy the following requirements (SC 23-ACE-49, 1990):

  - "It must be shown that there will be no single failure or probable combination of failures under any foreseeable condition that would prevent the continued safe flight and landing of the airplane, or it must be shown that such failures are extremely improbable."

- "It must be shown that there will be no single failure or probable combination of failures under any foreseeable condition that would significantly reduce the capability of the airplane or the ability of the crew to cope with adverse operating conditions, or it must be shown that such failures are improbable."

- "Warning information must be provided to alert the crew to unsafe system operating conditions and to enable them to take appropriate corrective action. Systems, controls, and associated monitoring and warning means must be designed to minimize initiation of crew action that would create additional hazards."

- Electronic Flight Instrument System (EFIS) Lightning and HERF Protection

  - "Each system that performs critical functions must be designed and installed to ensure that the operation and operational capabilities of these critical functions are not adversely affected when the airplane is exposed to: (1) lightning and (2) high energy radiated electromagnetic fields external to the airplane."

  - "Each essential function of the system must be protected to ensure that the essential function can be recovered after the airplane has been exposed to lightning."

The descriptions above show how integrated digital avionic systems can be addressed by SCs. They also show how guidelines like RTCA/DO-160C could be used to satisfy the SCs, and indirectly, FAR Parts 23, 25, 27, and 29, section 1309, as well as FAR Part 33, sections 75 and 91.

4.2 Relevance of Informal Guidelines to Federal Regulations

This section shows what documents are used by data bus and integrated avionic equipment manufacturers to meet the requirements of FAR Parts 23, 25, 27, and 29, section 1309. For the purpose of this section, these documents are termed "informal guidelines."

The FAA has informally adopted RTCA/DO-160 as a means of complying with the environmental requirements of FAR Parts 23, 25, 27, and 29, section 1309. For example, in 1978, systems using the ARINC 429 data bus were submitted to the tests in RTCA/DO-160A. Today, systems that use the ARINC 429 bus are still subject to RTCA/DO-160, now called RTCA/DO-160C. Integrated systems and data buses that need to satisfy FAR Parts 23, 25, 27, and 29, section 1309, usually meet the requirements in RTCA/DO-160C.

If a data bus or an avionic system involves software, the software can be validated using the procedures in RTCA/DO-178. RTCA/DO-178 was published in 1982 specifically for the purpose of assisting with certification of complex avionic software. It was updated in 1985 and renamed RTCA/DO-178A. Again, data bus software and avionic software are usually submitted to the procedures in RTCA/DO-178.

Another informal guideline is the SAE's ARP 1834. It defines fault and failure analysis (F/FA) techniques for digital hardware. Since digital systems are fault prone, the FAA has decided that fault analysis should be employed during the certification process. FAR Parts 23, 25, 27, and 29, section 1309, and AC 25.1309-1A express the need for fault analysis, and ARP 1834 has provided a means for conducting such an analysis.

Even though FARs do not specifically mention these informal guidelines, their procedures are useful to manufacturers during the design of their systems. These informal guidelines address the appropriate regulations and have been well researched by organizations such as ARINC, SAE, and the FAA. Manufacturers may use the informal guidelines to evaluate complex parts within their systems, like data buses and their associated circuitry.

A data bus system must undergo many tests and analyses to meet the FARs. These tests are designed to ensure integrity and quality; help define redundancy and back-ups; help isolate systems, components, and elements; verify reliability; meet designed failure effect limits; and define error tolerance. Any document that addresses tests of this nature may be used as an informal guideline to satisfy the FARs. These include RTCA/DO-160, RTCA/DO-178A, and SAE ARP 1834.

### 4.2.1 Radio Technical Commission for Aeronautics DO-160C

Electromagnetic Emission and Susceptibility (EES) tests are conducted in accordance with RTCA/DO-160 to determine if certain waveforms are maintained in an electromagnetic interference environment. These tests were initially needed to satisfy AC 25.1309-1A, which describes a means of complying with FAR section 25.1309. With the addition of section 22, the tests also address the requirements of AC 20-136. EES testing should prove that certain environmental conditions which can adversely affect the aircraft will not cause single-point failures.

There are four sections in RTCA/DO-160C that may be used to satisfy FAR Parts 23, 25, 27, and 29, section 1309, and FAR Part 33, sections 75 and 91: Section 19, "Induced Signal Susceptibility," Section 20, "Radio Frequency Susceptibility - Radiated and Conducted," Section 21, "Emission of Radio Frequency Energy," and Section 22, "Lightning Induced Transient Susceptibility." Each section, and the tests contained therein, is described in Elwell et al. (1992). Although these tests can be used, others may be developed. Any other test should yield results that parallel RTCA/DO-160C, section 1, "Applicable Equipment Performance Standards." Further information about EES tests can be found in RTCA/DO-160C, or acquired from RTCA Special Committee 135.

### 4.2.2 Radio Technical Commission for Aeronautics DO-178A

Avionic systems that utilize software should be subjected to the procedures in RTCA/DO-178A, "Software Considerations in Airborne Systems and Equipment Certification." This document was developed by the European Organization for Civil Aviation Electronics, Working Group 12, and helps satisfy the FARs and ACs. RTCA/DO-178A presents procedures which verify that software failures in digital equipment and systems will not affect the aircraft in which they are installed. RTCA/DO-178A also shows specific methods and techniques to help the

designer with software design, testing, configuration, and documentation. Alternative methods for complying with RTCA/DO-178A can be used if the manufacturer shows that the techniques are parallel to the ones in RTCA/DO-178A.

It is beyond the scope of this paper to explain every aspect of RTCA/DO-178A. This section only covers procedures which can be used with FAR Parts 23, 25, 27, and 29, section 1309, and their associated ACs, AC User's Manuals, and SCs, as well as FAR Part 33, sections 75 and 91. The procedures are discussed below; only a brief description of each is provided since most are system dependant.

4.2.2.1 Developing a System which is Software Based

Two steps should be followed when defining a system that is to be certified and is software based. First, establish the system's criticality category; second, translate the criticality category to the software level.

To determine a system's criticality category, the manufacturer should assess the system's application and all failures which could result from a system malfunction. Flight-critical, flight-essential, and flight-nonessential are the accepted categories. A system is defined by its most critical function. The manufacturer may use simulations, similarity tests, ground and flight tests, and/or other appropriate methods to ascertain this information.

Software levels adopted by RTCA/DO-178A are Levels 1, 2, and 3. Generally, Level 1 corresponds to software used in flight-critical functions, Level 2 to that in flight-essential functions, and Level 3 to flight-nonessential functions. Once the software level is established, system development can begin.

System development begins with extracting the software requirements from the system requirements. This involves defining what the software should do, rather that how it should do it. Since this section of the development process is unique to the system, the manufacturer must be sure that the system requirements are well understood.

After software requirements are extracted and defined, software development can continue. See RTCA/DO-178A, section 5, for more information.

4.2.2.2 Software Development, Verification, and Validation

Once the software requirements are established, the manufacturer should develop, verify, and validate the system's software. The first part of this procedure requires that the manufacturer submit a software development plan to the regulatory agency. The plan should define the software functions; the criticality of each function and software level; hardware and software interfaces; microprocessor characteristics; built-in test (BIT) and monitoring requirements; what functional losses could occur as a result of software failure; and timing, test, and partitioning requirements (RTCA/DO-178A, 1985). An approach to help formulate the software development plan is shown in RTCA/DO-178A, figure 6-1.

After the software is developed according to the approved plan, the manufacturer can begin to verify the software through testing. Discussed in RTCA/DO-178A are module tests, module integration tests, and hardware and software integration tests. Because these tests can be lengthy and are all system dependent, only an explanation of module testing is provided below. Module integration testing, hardware and software integration testing, and assurance of each are described on pages 24 through 30 of RTCA/DO-178A.

Module tests include logic and computation tests which verify that the module performs its intended function. Logic testing is used to detect illogical sequences and constructs. Typical errors that logic tests detect are halted execution, executions trapped in a loop, incorrect logic decisions, lack of logic to handle certain input conditions, and missing input data.

Computation tests are used to detect errors. The errors can appear in a computational sequence or numerical algorithm. A computational test may consider an algorithm's reaction to data within a specified range, data outside a specified range, and data that is on the border of a specified range. For example, an altitude-measuring algorithm may produce results based on digital data from a flight computer. If, for some reason, the algorithm receives data that is not within the specified range, should the algorithm assume a zero value or should the algorithm repeat its function again with the next data? These are typical questions which a computational test should address.

Many types of computational tests can be selected since they are dependant on the system's parameters. It remains the responsibility of the manufacturer to properly define and execute these tests.

For flight-critical systems, all verification results must be retained and all problems logged. For flight-essential systems, only a Statement of Compliance is required as a summary of the verification process. No documentation is required for nonessential systems.

Once a system's development and verification tests are complete, the system's validation may begin. System validation usually includes an evaluation and testing process, and may be done in accordance with system verification and development testing. System validation should demonstrate the following:

- System requirements comply with the appropriate regulations. (This can be confirmed by simulations or environmental and performance analyses.)

- The system functions properly under adverse operating and failure conditions.

As with system development and verification, system validation will vary in complexity and extent depending on the system's characteristics and criticality category.

4.2.2.3 Software Configuration Management and Software Quality Assurance

Systems involving software must also undergo Software Configuration Management (SCM) and Software Quality Assurance (SQA). These methods describe how to

improve identification, control, and auditing of software. SCM and SQA methods in RTCA/DO-178A are drawn directly from proven methods of hardware control.

As with software verification, SCM requires the use of an SCM plan. This plan may be part of the overall SQA plan. The SCM plan includes a description of how SCM will be implemented and followed throughout the system's certification process. It should further discuss how SCM will be applied during the service life of the equipment.

The SCM should include documentation, identification, and change control and status accounting. Documents which satisfy the documentation part of the SCM are included in RTCA/DO-178A, section 8.

The SQA plan should identify and evaluate quality problems and ensure corrective action (RTCA/DO-178A, 1985). An SQA plan should include the purpose; quality assurance functions; documentation; policies, procedures, and practices; reviews and audits; configuration management; medium control; testing; supplier control; and appropriate records. A brief description of each is provided on pages 39 and 40 of RTCA/DO-178A.

SCM and SQA procedures are interrelated. Therefore, their plans should be coordinated to eliminate unnecessary redundancy. The procedures outlined above are fully explained in RTCA/DO-178A.

### 4.2.3 Society of Automotive Engineers ARP 1834

Failure analysis on data buses and integrated avionic equipment can be accomplished using procedures in ARP 1834, "Fault/Failure Analysis for Digital Systems and Equipment" (1986). The need for failure analysis techniques is pointed out in AC 25.1309-1A and FAR Parts 23, 25, 27, and 29, section 1309, and is implied by FAR Part 33, section 75. ARP 1834 has been adopted as an informal guideline for meeting these requirements. ARP 1834's analyses are specifically meant to identify digital equipment hardware faults.

ARP 1834 is not an exhaustive or universally accepted method for applying F/FA. It is used merely to present cost effective, industry acceptable means for identifying failure modes and failure effects.

Manufacturers who wish to use ARP 1834 as a certification guideline should discuss their reasoning with the regulatory agency early in the process. This is because variations of approaches presented in ARP 1834 will need to be employed under different circumstances. For systems that are flight-critical or flight-essential in nature, one approach might be to develop design techniques for a fault tolerant system. Design techniques most often employed in this situation are as follows:

- Similar or dissimilar redundancy, signal consolidation, and hardware functional partitioning.

- Fault detection and isolation that uses comparison monitoring of redundant elements, along with in-line tests, monitoring, and reasonableness checks.

- Fault response with system reconfiguration and shutdown, and operational mode changing.

It is the system designer's responsibility to establish the system's objective.

When selecting an F/FA, one must decide whether to employ a top-down or bottom-up approach. The top-down approach begins at the system level and proceeds down to the component design. Here, the failures that produce a particular system malfunction effect can be found (SAE ARP 1834, 1986). Fault Tree Analysis (FTA) is an example of the top-down approach.

The bottom-up approach begins at the part or component level, and moves upward to the system level. This allows failure effects on the next higher level to be identified. Failure Mode and Effects Analysis (FMEA) is an example of the bottom-up approach.

Other factors that help the manufacturer select an F/FA approach are furnished on page 14 of ARP 1834. Descriptions, as well as applications to top-down and bottom-up approaches, are provided. For evaluating flight-critical or flight-essential functions, both top-down and bottom-up approaches should be used.

Certifying a digital avionic system for flight-critical or flight-essential operation could require an F/FA such as FTA and FMEA. This is pointed out in ARP 1834, but procedures for FTA or FMEA are not given. Section 5.4 of this chapter describes these methods in detail, as well as chapter 3 (Curd 1989) of this handbook. Also, MIL-STD-1629A presents steps for a military FMEA, from which procedures can be drawn to satisfy AC 25.1309-1A.

ARP 1834 points out basic methods of F/FA, including analyses of digital, processor-based systems. Pages 30 through 37 of ARP 1834 show a detailed F/FA procedure for these systems. Special methods include fault insertion using hardware, emulation, and computer simulation. Also, appendices A, B, and C of ARP 1834 contain examples of top-down, bottom-up, and emulation F/FA approaches, respectively.

## 4.3 Relevance of Manufacturer Testing to Federal Regulations

Most every manufacturer who produces data buses or integrated avionic equipment follows RTCA/DO-160, and forms of RTCA/DO-178 and SAE ARP 1834. This is because the FAA has dubbed them "acceptable means for showing compliance" with the FARs (AC 25.1309-1A, 1988). When manufacturers run across something that has not been addressed in the informal guidelines, they must develop their own validation techniques to show compliance. These validation techniques are usually chosen to satisfy FAR Parts 23, 25, 27, and 29, section 1309, as well as FAR Part 33, sections 75 and 91.

This process was followed for the ARINC 429 data bus. Environmental tests on the original bus were conducted by the BCAC in accordance with RTCA/DO-160A. In addition to the test procedures in RTCA/DO-160A, BCAC conducted other tests on the bus's components. This was necessary because RTCA/DO-160A did not address all aspects of the data bus. The tests are outlined in ARINC Specifica-

tion 429-12. Honeywell's Sperry Commercial Flight Systems Group and Rockwell's Collins Division (both in conjunction with GAMA) have adopted similar procedures for the CSDB and ASCB, respectively.

Other tests (like those performed by BCAC) are developed to address bus requirements that the informal guidelines miss. For the purpose of this section, these tests are broken into two categories: external and internal. External tests could be either laboratory tests or computer simulations, while internal tests are used by components to check themselves (e.g., verify data words, labels, or characters). Internal tests include monitoring, error detection, and synchronization, and may go down to the bit level.

External and internal tests are not defined by the FARs, but are considerations that help ensure that the bus performs its intended function. Without them the bus may still function, but its integrity would be significantly decreased.

The following four sections discuss how the informal guideline tests and these manufacturer's tests are applied to avionic data buses. Because there are many of these tests, and some are proprietary to the manufacturer, only brief discussions are provided.

### 4.3.1 ARINC 429 Data Bus

The ARINC 429 bus is a digital broadcast data bus made up of a transmitter, receivers, and wire. It was developed by the Airlines Electronic Engineering Committee's (AEEC) Systems Architecture and Interfaces (SAI) subcommittee. The AEEC, which is sponsored by ARINC, released the first publication of ARINC *Specification 429 in 1978. At that time, the specification* contained the basic philosophy of the bus, as well as data transfer and format characteristics.

Included in the original specification were tests of the bus and its interface circuitry. Environmental testing was conducted in accordance with RTCA/DO-160 (this was the only informal guideline in this section that the ARINC 429 bus satisfied). The ARINC 429 bus also underwent external tests such as receiver data detection techniques, laboratory tests, and computer simulations to prove that the bus was fully operational.

Laboratory tests and computer simulations were used to assess pulse distortions on the data bus. For the laboratory tests, the bus was configured with Number 20 American Wire Gauge cable in a typical Boeing 747. A pulse was generated by an ARINC 429 bus transmitter and viewed at the outputs of the transmitter and at a receiver. The results were viewed with an oscilloscope. Computer simulations modeled the whole bus, with the bus's model being drawn from the wire characteristics. The computer simulation included analyzing voltage waveforms and transmitter impedance. More detailed descriptions of these tests are provided in appendix 1 of ARINC Specification 429-12.

Internal tests are done by the bus on itself (these are included in ARINC Specification 429-12). The tests include data word counts, parity checks, and cyclic redundancy checks (CRCs).

Word counts are used by ARINC 429 LRUs to verify that the number of words at the receiver is the number of words expected. If the number of words does not match, the receiver notifies the transmitter within a specified amount of time.

Parity checks use one bit of the 32-bit ARINC 429 data word. Odd parity was chosen as the accepted scheme for ARINC 429-compatible LRUs. If a receiving LRU detects odd parity in a data word, it continues to process that word. If the LRU detects even parity, it ignores the data word. (Parity checks are described in detail in section 5.1 of this chapter).

CRCs are used by ARINC 429 LRUs to verify groups of data words or data strings. A description of the CRC is given in section 5.1.

This section described how some external and internal tests were used to verify the ARINC 429 bus's operation, and, indirectly, satisfy FAR Parts 23, 25, 27, and 29, section 1309. Today, many similar tests are being developed and executed on the ARINC 429 data bus.

### 4.3.2 Commercial Standard Data Bus

The CSDB is GA's ARINC 429 bus. It connects avionic LRUs point-to-point to provide an asynchronous broadcast method of transmission. More information about the bus's operating characteristics is contained in the standard, which is available through GAMA.

Before the bus could be used in an avionic environment, it was put through validation tests similar to those used on the other buses. These included the environmental tests presented in *RTCA/DO-160* and *failure analyses*. Most environmental tests were done transparently on the bus after it was installed in an aircraft.

As with the other buses, Rockwell's Collins Division had to develop external tests to show that the bus satisfied specifications in the standard. Test procedures of this nature are not included.

Internal bus tests that the CSDB standard describes include a checksum test and a parity check. Both of these are used to ensure the integrity of the bus's data. Care should be taken when using these tests because their characteristics do not allow them to be used in systems of all criticality levels. Further information about both tests is provided in section 5.1.

These are not the only external and internal tests that the CSDB manufacturer can perform. Many more characteristics which may require testing are presented in the CSDB specification. Again, it remains the manufacturer's responsibility to prove that exhaustive validation testing (VT) of the bus and its related equipment has met all the requirements of the FARs.

### 4.3.3 ARINC 629 Data Bus

The ARINC 629 data bus is a high-speed, bidirectional data bus, which uses a bus protocol that supports both periodic and aperiodic data. It was developed by BCAC prior to 1981.

Much information has been published on the ARINC 629 bus over the last 10 years. The data bus has been the focus of many technical papers and symposiums. ARINC's SAI subcommittee, which published part one of the bus standard, is currently working on parts two, three, and four. These drafts are called the Applications Guide, Data Standards, and Test Plan, respectively. Each of these parts has been distributed by ARINC in draft form.

Part four of the Test Plan contains a "complete" set of external tests for ARINC 629 bus components, or for groups of components within the data link and physical layers of the bus. It also contains a section explaining the environmental tests considered for the ARINC 629 bus.

External tests in the Test Plan address the bus's components. The Current Mode Coupler (CMC), Serial Interface Module (SIM), and terminal are all components considered by the Test Plan. The Test Plan also states that each of these components will be subjected to different tests. A list of the component tests is included in Attachment 1 of the Test Plan. Once the single units complete their testing, they should be tied together and tested in conjunction with one another. This hierarchial approach makes general test cases easier to identify. No formal external test procedures are presented here because they are not specified in the draft of the Test Plan.

Internal tests used by the ARINC 629 bus range from simple ones that verify parity to complicated ones that ensure a bus user, or terminal, will not broadcast out of turn. Since there are many internal tests which can be performed, only a few examples are given.

One internal test involves monitoring performed by a BIU. There are three types of terminal monitoring: receive data monitoring, transmission monitoring, and protocol checking. Only the protocol check is discussed here.

A protocol check is used by a BIU subsequent to transmission. The purpose of this check is to ensure that a transmitter will not place data on the bus at the wrong time. In this way, orderly periodic and aperiodic transmission occurs between terminals. The protocol check requires the transmitter to satisfy the following three conditions between two transmissions (if these conditions are not met, transmission is inhibited [Shaw and Sutcliffe 1988]):

- A Transmit Interval (TI) must have passed. This TI is common to all terminals on the bus.

- A quiet period called the Synchronization Gap (SG) must have passed. This SG is also common to all terminals.

- A quiet period called the Terminal Gap (TG) must have passed since the SG and since the end of any other terminal's transmission. This TG is unique to each ARINC 629 terminal.

Other internal tests that the ARINC 629 bus performs are parity checking, data format, and modulation. These tests are performed in the data link layer, and are done on each label and data word. Parity checking on the ARINC 629 bus

parallels the ARINC 429 bus's parity checking. The ARINC 629 bus parity check is accompanied by a modulation check.

Two other internal tests that are performed are checksum and CRC. Since these discussions parallel the one given in section 4.3.1, they are not restated here.

Many more external and internal tests are required for the ARINC 629 bus because it is a complicated bus. They are pointed out in the ARINC specification, technical papers, and symposiums. BCAC and associated manufacturers will continue this type of testing long after the ARINC 629 bus specification is complete. However, the point of the tests remains unchanged; both internal and external tests are required to show the FAA that the ARINC 629 bus can be reliably implemented.

### 4.3.4 Avionics Standard Communications Bus

The ASCB is primarily used on GA aircraft, such as business jets and commuter turboprop aircraft. Because integrated avionic systems in these aircraft still need to satisfy the FAA's requirements for airworthiness, testing similar to the other buses must be performed.

There are three versions of the ASCB: A, B, and C. Version A was designed for use in flight-nonessential systems, Version B for flight-essential systems, and Version C for flight-critical systems. Only Versions A and B are implemented on aircraft and covered in the current GAMA specification. Version C is currently under development.

As with the ARINC 429 bus, the ASCB had to undergo tests outlined in *RTCA/DO-160*, as well as others defined by the manufacturers. These tests are more detailed than those of the ARINC 429 bus because the ASCB uses a bidirectional (half-duplex) architecture. Tests that address the ASCB's BC and waveform tests are examples of external tests that can be performed by the manufacturer.

The ASCB is controlled by a BC. Because the BC provides central bus control, the ASCB incorporates a redundant BC in case the primary BC fails. An external test that involves these BCs should verify that control is properly transferred from one BC to the other in the amount of time specified by the standard, and that the primary BC will relinquish control in the event of a failure (e.g., power interruption).

A waveform test should also be performed on the ASCB. Here, combinations of stub lengths and unterminated stubs are subjected to bit-errors and signal alterations. This external test shows whether bus data is affected by the medium's characteristics.

Internal tests, like those pointed out for the ARINC 429 bus, are performed by the BIU. These are applied to ensure that the bus conforms to the standard. Tests of this nature include CRCs and Transmission Validation.

The tests discussed above are not the only external and internal tests that can be performed by the manufacturer. Many more bus characteristics that require testing are presented in the ASCB specification and throughout various technical

papers. Whether a test is conceived by the manufacturer or drawn from another document, an ASCB manufacturer must prove to the FAA that the bus and its related equipment has met all of the requirements of the FARs.

### 4.3.5 Summary

Section 4 showed which FARs and ACs are applicable to the certification of data buses and integrated avionic systems. It then discussed SCs and their relationship to the FARs.

After the federal regulations were defined, section 4 discussed the informal guidelines that showed what documents are used by data bus and integrated avionic equipment manufacturers to meet the requirements of FAR Parts 23, 25, 27, and 29, section 1309, and FAR Part 33, sections 75 and 91. Tests presented in these informal guidelines are designed to ensure the system's integrity and quality; verify reliability; help specify redundancy and back-ups; help isolate systems, components, and elements; and help define error tolerance. Documents presented in this section included RTCA/DO-160C, RTCA/DO-178A, and SAE ARP 1834. Other documents may be used (as informal guidelines) to satisfy the FARs if their procedures meet the same ends.

Although civilian aircraft satisfy different requirements than military aircraft, analyses for integrated avionic systems in civilian aircraft can be drawn from military documents. For example, MIL-STD-1692A discusses an FMEA for military avionic systems. Since FMEA is an integration standard for civilian aviation, concepts from this document can be applied to the certification of civilian aircraft.

If manufacturers run across something not addressed by the informal guidelines, they must develop their own validation techniques to show compliance. These validation techniques are usually chosen to comply with FAR Parts 23, 25, 27, and 29, section 1309, as well as FAR Part 33, sections 75 and 91.

Developing proper validation techniques should be a main concern of the integrated avionic system manufacturer. These techniques must consider all failure modes of the system, even ones that are unique and infrequent. Failure to do this could result in hazardous conditions, even if the system is mature. Lessons can be drawn from the MIL-STD-1553 data bus and its associated equipment (Earhart 1991).

For example, the MIL-STD-1553 has undergone extensive tests over the last decade. Throughout this time period, the MIL-STD-1553 has been accepted as the data bus for most military equipment. Even with all the testing and validation, there is still some apprehension about validating MIL-STD-1553 and its associated electronics. Much of this apprehension is the result of poor VT.

One reason errors occur is because some manufacturers feel that total system VT is not necessary. Total validation requires testing single components first, and then testing them in conjunction with each other. A manufacturer who tests only the single components could easily overlook system-wide errors.

Another reason errors occur is because some manufacturers only test a system once and use the results for subsequent systems. Just because a system functioned properly throughout the first tests does not mean that each similar system will yield the same results. This is especially true if the system is to be installed on a different aircraft or controlled by different software or firmware. Some MIL-STD-1553 terminals tested by Test Systems of Phoenix, Arizona, have been found to contain incorrect transformers and transceivers.

VT should also verify that operating LRUs satisfy the bus's standard. Just because an LRU works in a system does not mean it meets the bus's standard. Tests like this should be presented in each LRU's test plan; this plan helps manufacturers verify results and defines the LRU's error margins and tolerances. Tests should also check margins and tolerances that are not considered under normal operation or operational testing (Earhart 1991).

The above example shows how poor VT could impact certification of well-known products. Often, proper tests for digital avionic equipment are not established until unique failure conditions appear. This is one reason that implementation of complex avionic systems usually follows years of design. Although the ARINC 429 data bus was developed prior to 1978, it has taken years to achieve the current level of reliability. On the other hand, the ARINC 629 data bus was developed prior to 1980 and is still not being used in production aircraft. To breach this design barrier, the avionic system's manufacturer should collaborate with the FAA early in the design process and thoroughly validate all aspects of their systems. This type of process will represent a challenge for both the system expert and the FAA.

# 5. BUS-INTEGRATED SYSTEMS TECHNOLOGY

This section focuses on technical issues related to the use of data buses in avionic systems. Particular emphasis is placed on issues specific to the integration of systems using data buses.

Section 5.1 introduces data bus architectures and examines the integration issues. Concerns relating to avionic system and BIU interaction are examined in Section 5.2, Bus Hardware-Software Interaction. Methods for protocol development and verification are presented in Section 5.3, Protocol Specification and Verification Methods. Finally, the guidelines used for bus integration are identified and examined in Section 5.4, Bus Integration Standards, Guidelines, and Techniques.

## 5.1 System Integration Concerns

Factors such as weight, power consumption, maintainability, reliability, flexibility, and the cost of ownership are just a few of the general concerns when evaluating a system design. This section examines the specific concerns relating to the use and integration of avionic data buses. Different bus architectures and protocols are addressed first, then particular integrity issues, and, finally, the issues of data bus monitoring and maintenance.

Data buses used in aircraft have distinct advantages over point-to-point wiring. One advantage is the reduction in the number of wires and connectors, and another is the flexibility gained when adding, deleting, or modifying the system.

There are two basic types of data buses: unidirectional and bidirectional. Although there are many areas of concern common to both types, a bidirectional bus has additional areas of concern. These are related to the data bus access protocol. This determines when and how often a transmitter may gain control of the bus. A discussion of access protocols is contained in section 5.1.2. In a unidirectional data bus, which has only one transmitter, there is no need for control to be relinquished, hence, there is no concern over an access protocol.

Following are four major areas of concern that have been identified as relating to bus interfaces (Hecht and Hecht 1985):

- Address errors
- Internal inconsistencies
- Denial of access
- "Babbling" transmitters

Address errors are a corruption of the address field of a transmitted message. On bidirectional buses there may be address fields in a message for both the source and destination. This is especially true if the protocol requires an acknowledgement message to be returned to the sender. In this case, an error which occurs in a source address field will be easily detected since the acknowledgement will not be received.

An internal inconsistency exists when the data passed between bus users fails to adhere to the predefined format. These formats are given in detail by the particular bus specification and should be tested by the receiver for conformity. Data words may contain fields for error checking, sign bit, status bits, address bits, data bits, etc. The receiver typically uses one or more forms of error detection, such as a CRC or a parity check, as a basis for message acceptance or rejection. A rejected message may be retransmitted, or a default value or alternate data source used.

Denial of access is a problem associated with bidirectional buses that needs careful attention during the design, implementation, and operation of a bus. A bus user is denied access when the user has information to send but the bus is not available due to an error. One such error could be another bus user failing to terminate its bus transmission. A failure in this area renders the bus useless to one or more bus users. Therefore, access protocols and bus interface hardware need to be carefully designed.

Babbling transmitters are those that fail to abide by the access protocol rules. Due to a failure of bus interface hardware or software, the transmitter is activated during another transmitter's access time. If not terminated, this type of failure denies all other users access to the bus.

These four concerns are not only data bus interface concerns, but specific integration concerns as well. New bus users must be tested to ensure that if they are incorrectly addressed because of an address error, they discard the message. They must follow the predefined format of the data bus specification. Users not in compliance due to an internal inconsistency problem will either generate errors or not detect them when they occur. Denial of access can become a problem for bidirectional data buses when new users are added. This can happen if not enough bus capacity is allowed for new users. Babbling may occur if new users are not configured with the correct protocol parameters.

Another concern is that of specification completeness. Integration of equipment on the same data bus may involve equipment from separate manufacturers. When this occurs, certain parameters which may be undefined or incompletely defined in the bus specification are subject to differing interpretations. This difference of interpretation may later cause a bus failure. This concern is specifically addressed in section 5.3.

The areas of concern are addressed in the following sections as appropriate. Protocols, although defined as part of a system architecture, are examined in a separate section that deals more specifically with protocol concerns.

### 5.1.1 Architecture Related Concerns

To understand data bus integration problems it is helpful to first understand the different data bus architectures used. Data buses are increasingly referred to as networks by those who work with and around them. There are fundamental differences between avionic data bus networks and computer networks. These differences are generally dictated by the intended use of the network.

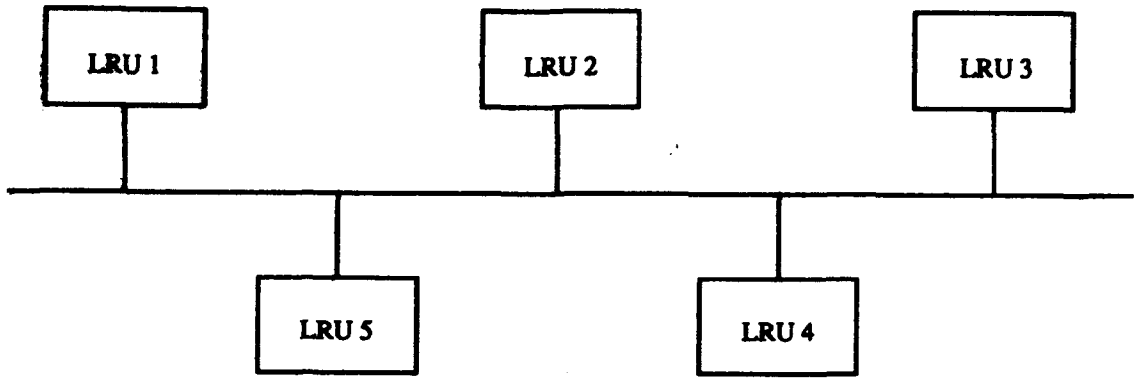
Computer networks are designed for purposes such as database access, integrated voice and data transmission, resource sharing, file transfer, process control, and general communication. On the other hand, avionic data buses were viewed only as a way to save wiring and weight and enhance system performance by sharing common resources. The function of the bus was to transfer certain variables from one bus user to another at a fixed update rate. With enhancements in protocols and advancements in IC densities, data bus performance has risen. So has the interest in using the data bus for purposes that resemble computer networks. For example, ARINC Specifications 429-12 and 629 define protocols for transferring files among bus users.

Networks use many different architectures. Some network architectures are defined on the basis of response time; others are defined on the basis of security, reliability, cost, or a combination of these. Where data buses are used in flight-essential or flight-critical applications, the architecture is designed with throughput and reliability as key factors.

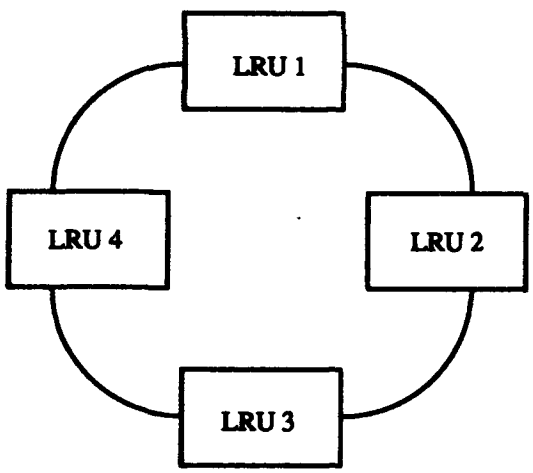
#### 5.1.1.1 Basic Bus Architectures

One technique useful in defining a bus architecture is the physical layout. The physical arrangement of bus users in a network is called a topology. Various methods have been used to connect bus users with data buses. Some common topologies are illustrated in figure 5.1-1. In a linear topology, LRUs are added by sequentially attaching them to the data bus. All LRUs can listen to any transmission on the bus. For a ring topology, the ring must be broken to add new LRUs. Messages are passed sequentially from one LRU to the next. In the star topology, the LRUs are connected to a central hub. A message from an LRU passes through the hub to any or all other LRUs on the hub.

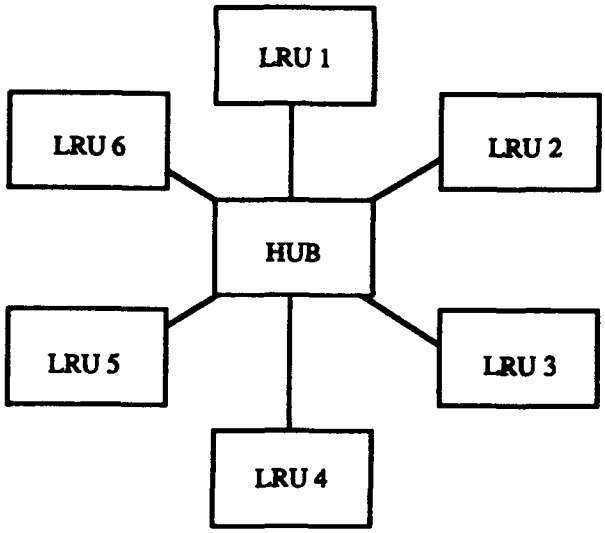
Some of the possible topologies for connecting one data bus to another are shown in figure 5.1-2. When one data bus is controlled by another it is called a hierarchical topology. This is common in a military data bus topology, which uses the MIL-STD-1553 bus. In civilian aircraft, it is common for buses to be equal and share data as required ("MIL-STD-1553 Designer's Guide," 1982).




**LINEAR**




**RING**




**STAR**

FIGURE 5.1-1. COMMON DATA BUS TOPOLOGIES

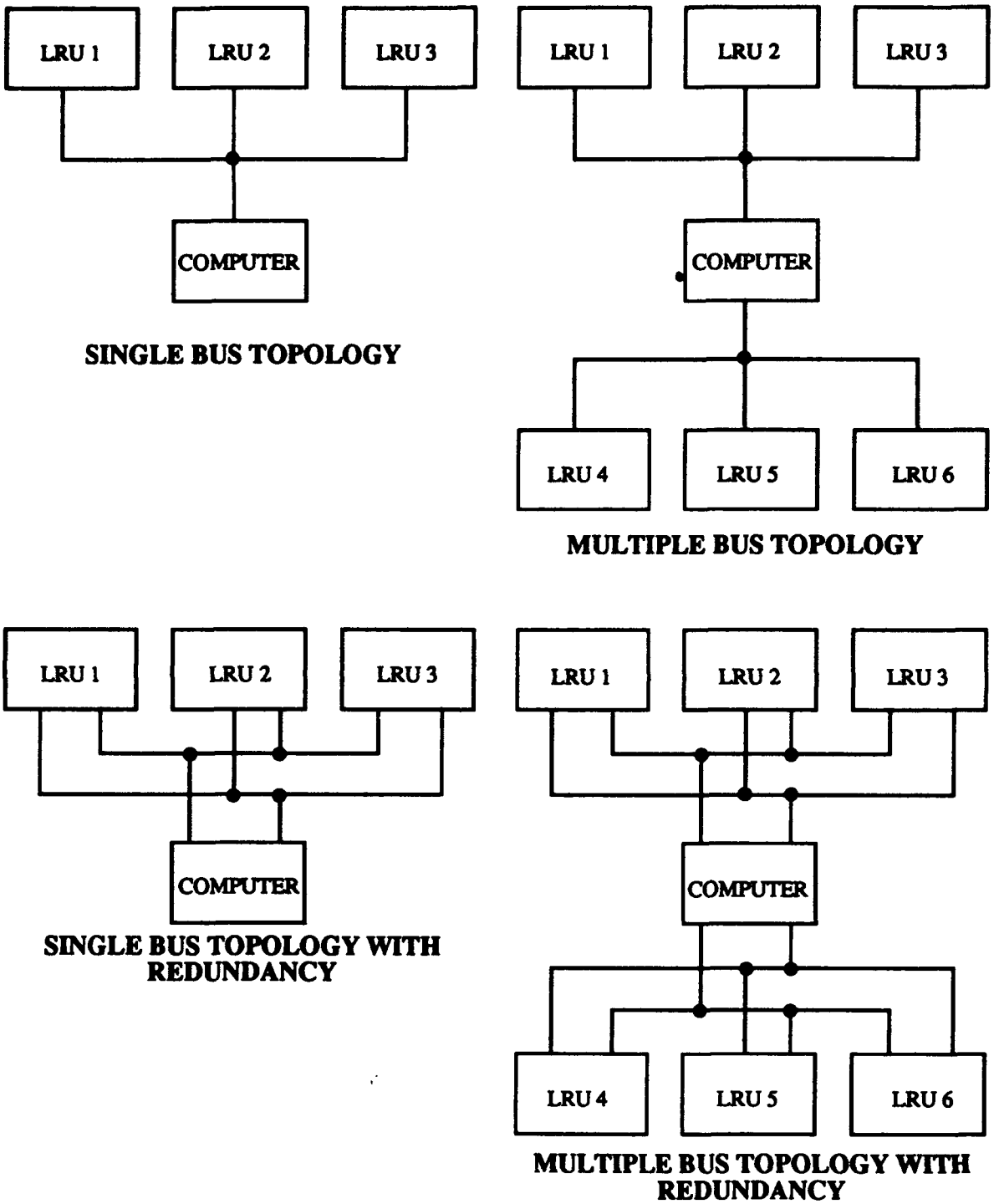



FIGURE 5.1-2. LINEAR DATA BUS TOPOLOGIES

Redundancy is used in a data bus architecture to provide continued operation on one data bus if there is a failure on another, regardless of the cause of failure and whether or not the error is a recoverable type. Redundancy is implemented both physically and functionally. Physical redundancy requires two or more of the same item. If one fails, the other is used. For this type of redundancy to work successfully, a means of failure detection and system reconfiguration is required. Functional redundancy requires that the function be duplicated, but in a dissimilar way. The implementation of redundancy is vital in systems that provide flight-critical functions. Redundant designs require careful attention by the system designer.

#### 5.1.1.2 Control Architectures

How a data bus is controlled has an affect on the bus architecture. There are two types of control which dominate avionic data bus systems: distributed and centralized. With centralized control, a single controller directs all the activity of the data bus. There are no transmissions from any bus user unless directed by the BC. This controller will have a list of the addresses of all the bus users and will transmit a command to each user at the designated rate, giving each user a chance to access the bus and send any data required by other bus users. The ASCB and MIL-STD-1553 bus use centralized control.

Distributed control refers to a system that is not centrally controlled. Instead of the access control being contained in a central controller, it is programmed into each device which is connected to the data bus. Each user has been programmed to follow an identical set of access rules without variation. The ARINC 629 bus uses distributed control.

#### 5.1.1.3 Functionally Partitioned Architectures

Another technique used in defining the bus architecture is functional partitioning. This means that data buses are defined by functions which they perform and are grouped accordingly. For example, in the ARINC 629 bus implementation (planned for use on the Boeing 777) systems are partitioned according to their function, such as fly-by-wire, system, and display functions (Bailey 1990). Data sharing among bus users is more easily accomplished when the users requiring the data are on the same bus as the users supplying the data. When this is not done, some method of linking the data buses together is required. This can be accomplished with gateways and bridges.

#### 5.1.1.4 Multiple Bus Architectures

The use of gateways and bridges is another facet of integration concerns associated with a data bus architecture. Avionic systems that are required to share data may use different data bus protocols. A gateway is used to connect two or more data buses so that a user on a bus using protocol X may communicate with a user of another data bus, which uses protocol Y. A gateway may be a standalone interface or part of an LRU. The gateway functions as a protocol converter, converting data packets, wordstrings, or frames from one format to another. A gateway used between two buses is required to perform two data conversions, protocol X to protocol Y, and protocol Y to protocol X.

When it is necessary to share information between data buses which use the same protocol but must remain isolated, a bus bridge is used. Figure 5.1-3 shows examples of how buses may be connected by gateways and bridges.

An example of an avionic device which fits the gateway definition is found in ARINC Specification 429-12, appendix 2, where it is referred to as a "data exchange buffer." The specification describes an interface between the MIL-STD-1553 command/response data bus and the ARINC 429 broadcast data bus. Some of the possible conversions between these two buses could be changing the destination label or address, changing the ordering of bits, or generating and testing the error checking mechanism used on the particular bus.

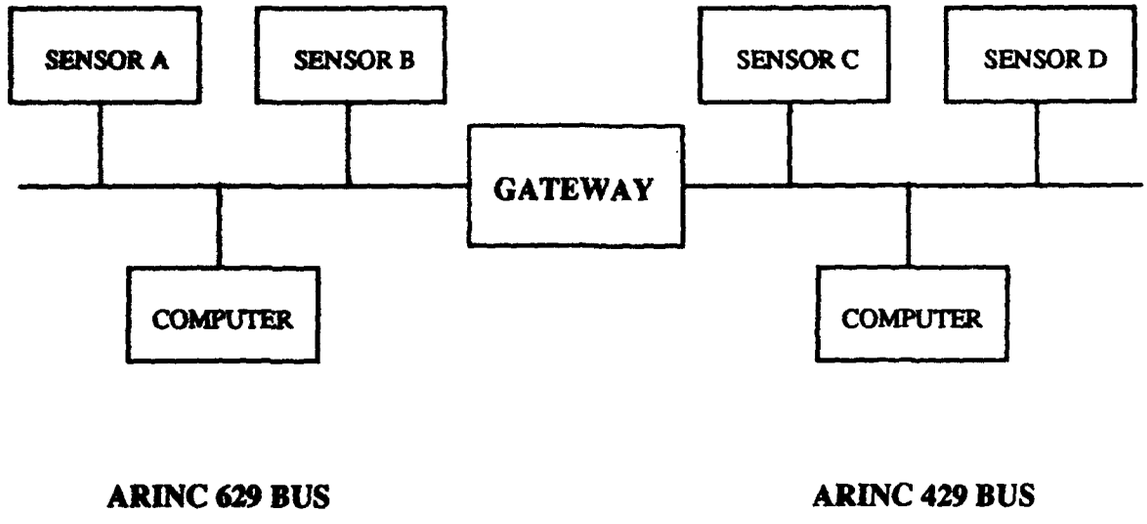
One possible implementation of a gateway may require a conversion from parity error detection to a CRC detection technique requiring the generation of a CRC check word. A gateway may implement this conversion in hardware or software and will have a throughput delay based on the particular implementation chosen. In general, a software technique would produce a lower cost with a higher delay, and a hardware technique would produce a lower delay, but at a higher cost.

A gateway is more complex than the user bus interface since it needs to deal with protocols for two different buses and their associated data formats. Data latency is increased in a configuration which uses gateways, due to the time it takes a variable to pass from one bus to another through the gateway. If the gateway or bridge causes data to be "momentarily stored" (ARINC Specification 429-12, appendix 2, 1990), then the system performance could be affected due to a "stale data" condition.

It should be the goal of any gateway design to keep the storage time small in relation to the other time parameters.

Careful consideration should be given not only to the data latency problem but also to the handling of bus errors through the gateway. Should an error that was detected by the ARINC 429 bus interface be passed through the gateway to the MIL-STD-1553 bus so that the intended receiver will detect it and take appropriate action? Should there be a bit reserved in the data format to handle this situation? Should the old data be stored in the gateway and used until a correct error free update is received? The particular error recovery method that is used should be consistent with the particular standard and have minimal impact on system operation.

A protocol that uses an acknowledgement response from the receiver for verifying correct receipt of data, will have additional constraints when used in a system containing a gateway. If the protocol at the receiving end is not required to issue an acknowledgement, but the sender requires it, then the end-to-end integrity is broken at the gateway interface. If an acknowledgement is required by both the sender and receiver, then the timeout value for the sender should take into account the round trip delay introduced by the gateway.



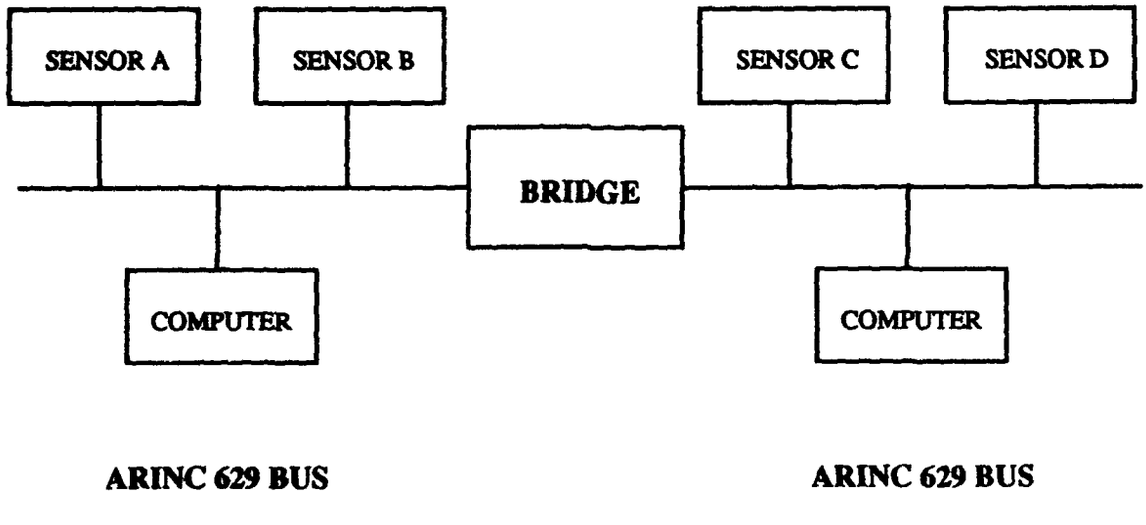




FIGURE 5.1-3. GATEWAY AND BRIDGE USED IN AVIONIC SYSTEMS

Periodicity is an attribute which may be affected by a gateway implementation. A periodic bus is one in which data arrive at the receiver at regular time intervals. Different protocols meet at the gateway. Each protocol by itself may be periodic, but when linked to another protocol the result appears as an aperiodic bus. This is due to the fact that the two protocols are not synchronous. A wordstring that arrives at the gateway from bus X may have just missed the transmission for bus Y. A later transmission from bus X may be just in time for bus Y. Operation in this manner means that at certain unspecified times the data will be fresh and at other times it will be stale. Systems that require information to be updated at certain rates need to be analyzed closely to determine if the introduction of a gateway will degrade the system operation.

In addition to the problems mentioned above, hardware-software interaction problems, discussed in section 5.2, also apply. This is because, to each bus the gateway resembles an LRU with a bus interface and host CPU.

### 5.1.2 Protocol Related Concerns

Multiple transmitters can use one bus by using time-based multiplexing. This multiplexing requires that a bus access protocol be defined to ensure that, at any one time, only one user is transmitting. The bus access protocol is a set of rules by which all bus users must abide to access the bus and ensure its specified operation. The basic types of access protocols which could be considered for use with bidirectional data buses are as follows:

- Contention
- Time slot allocation
- Command/response
- Token passing

With the contention protocol any bus user may transmit on the bus at any time after the bus becomes idle. If two bus users start transmitting at the same time, a collision of data occurs and the data are corrupted. Collisions are a normal event with this type of protocol. This protocol works well under light use but tends to collapse under heavy loading due to numerous collisions.

The time slot allocation protocol assigns a unique, predefined time slot during which each bus user may access the bus. Each user listens to the bus for a period of inactivity. When the assigned time occurs for a particular bus user, it may take control of the bus. Access to the bus is not attempted again until the necessary time passes, which allows all other users to access the bus.

In a command/response protocol no bus user may transmit without receiving permission from the BC. There is only one BC active at any time. The failure of one BC should cause the activation of an alternate BC.

A token passing protocol allows a bus user to transmit only after it receives the unique bit pattern, referred to as the "token." It receives the token, sends any waiting message(s), and passes the token on to the next user.

There are also variations of these protocols that make the differences between them unclear. For instance, a command/response protocol can operate with a single central controller using a redundant standby controller for recovery. Under the same command/response protocol there can be a large number of BCs attached to a bus and all but one will be in the inactive state. Control of the bus can be passed from one controller to the next as each requires bus use. This technique closely resembles the operation of the token passing bus with a distributed control architecture. This is a more complex protocol.

Certain attributes have been identified as being highly desireable for avionic data bus protocols. These are fault tolerance, efficiency, simplicity, data integrity, support of synchronous and asynchronous data transfer, and predictability (Rich et al. 1983). Though they are not the only desireable features for a protocol, they do identify areas where major concerns have been expressed.

Fault tolerance describes the ability of a protocol to handle errors. Some protocols simply identify the fact that an error occurred; others are able to recover, possibly by a retransmission of the same data. Another recovery technique is a bus user isolating itself from the bus after detecting self-generated errors. A failure of the protocol should be identified by bus users and the reestablishment of order should be possible with little delay.

With respect to data transfer, efficiency is a measure of how much useful data are transmitted on the bus compared to the total number of bits transmitted. A large amount of overhead required for operation decreases the capability of the bus.

Simplicity is a measure of how understandable the protocol is. An easily understood protocol will benefit all areas of development, testing, and operation.

Data integrity depends on how well errors are detected to ensure that correct transmissions are made on the bus. The use of some form of error detection is necessary to achieve data integrity. Various methods are available to implement this feature and each differs in the types of errors which it will detect. The efficiency of the protocol is usually affected by the particular method used.

Most avionic data buses were designed to handle data that is synchronous. The handling of control inputs, along with more recent applications, such as on the Boeing 777 where the data bus may function as a general-purpose computer network, require that the data bus be able to handle asynchronous demands as well. This requirement necessitates that the overall throughput have the capacity to handle the uncontrolled load of aperiodic devices.

A deterministic protocol is one which is highly predictable. The specification states exactly how it will perform under all foreseeable conditions, and it can be verified that it does act according to this predetermined behavior. Asynchronous transfers detract from this characteristic and the truly random access protocols based on collision detection are nondeterministic. Protocols for avionic use are chosen because they are highly predictable.

A protocol which has the capability to deny access to a bus user is not a deterministic protocol. Even for a protocol that does not deny access, there may be errors that have the same effect as access denial. For example, a transmitter hardware failure may cause the transmitter to babble continuously, thereby denying other transmitters access to the bus.

In the following sections, some of the basic protocols are discussed and evaluated with respect to these preferred attributes.

5.1.2.1 Contention Protocols

A contention protocol in its pure form is nondeterministic. The term Carrier Sense Multiple Access (CSMA) describes the predominant form of this protocol. Multiple users are listening to the bus. When one has a message to send, it waits for the bus to be not busy and makes a transmission. From this simple description of CSMA operation one can easily see potential problems. This uncoordinated access protocol cannot guarantee that a message will ever be successfully transmitted. As more users are added to the bus, the CSMA protocol suffers from an increasing number of collisions during the contention period and, hence, wasted bandwidth. Unless some variation is made to this protocol to avoid contention, it will be plagued with poorer performance as new users are added. Therefore, many modifications have been made to this protocol in order to increase its reliability. For further details on contention protocols, see Elwell et al. (1992).

Since this protocol is not deterministic, it is not used in flight-essential or flight-critical avionic applications.

5.1.2.2 Time Slot Allocation Protocols

In a time slot allocation protocol, each user is given a preallocated time slot. The ARINC 629 bus uses a time slot allocation protocol that also accommodates asynchronous transmission. As with other bidirectional data bus protocols, this protocol uses collision detection, but collisions are not normal events as in a contention protocol.

The Time Division Multiple Access (TDMA) approach is the simplest form of a time slot allocation protocol. In pure TDMA, the time of occurrence and the duration of each user's time slot are predetermined. When one user's time has transpired, another user is given access to the bus. A more advanced form allows users to determine the time of access based on bus activity. The standard implementation is called the Dynamic Time Slot Allocation (DTSA) protocol. Although it is not an avionic data bus protocol, it is included to help understand the ARINC 629 bus. The details of the DTSA operation are given in appendix A of this chapter. The ARINC 629 bus implements a special form of DTSA.

In contrast to CSMA, which is based on a random access method, a time slot protocol relies on a unique and predefined access method for each user. Each user is guaranteed that, during its time slot under error-free conditions, it has sole access to the bus. This access method lends itself to a high bus efficiency, even under heavy loading conditions. Throughput under the CSMA

protocol, however, rapidly deteriorates with increasing access demands by its users.

5.1.2.2.1 ARINC 629 Bus

ARINC Specification 629, Part 1, defines a TG count which is similar to the count duration, $T_c$, for DTSA. It defines a unique value for each user based on a delay count. Bus access is permitted only after this count is satisfied. Another DTSA parameter, the Frame Time, $T_F$, is similar to the Minor Frame of the ARINC 629 bus specification in that it defines the cycle time of one user, from the start of transmission x to the start of transmission x+1. For either protocol, if transmission lengths are allowed to vary then the sequence of user accesses is maintained, but not the periodicity.

TDMA protocols operate in a cyclic fashion with the transmission of any user being predictable as far as the time slot is concerned. The ARINC 629 bus cycle, however, is more complex because three timers must be satisfied for bus access. Also, variations such as aperiodic transmissions are permitted.

ARINC Specification 629, Part 1, defines two basic modes of protocol operation. One is the Basic Protocol (BP), where transmissions may be periodic or aperiodic. Normal transmissions on the bus are periodic, but a condition such as bus overloading may force the protocol into an aperiodic mode. Transmission lengths are fairly constant, but can vary somewhat without causing aperiodic operation if sufficient overhead is allowed. In the Combined Protocol (CP) mode transmissions are divided into three groups for scheduling:

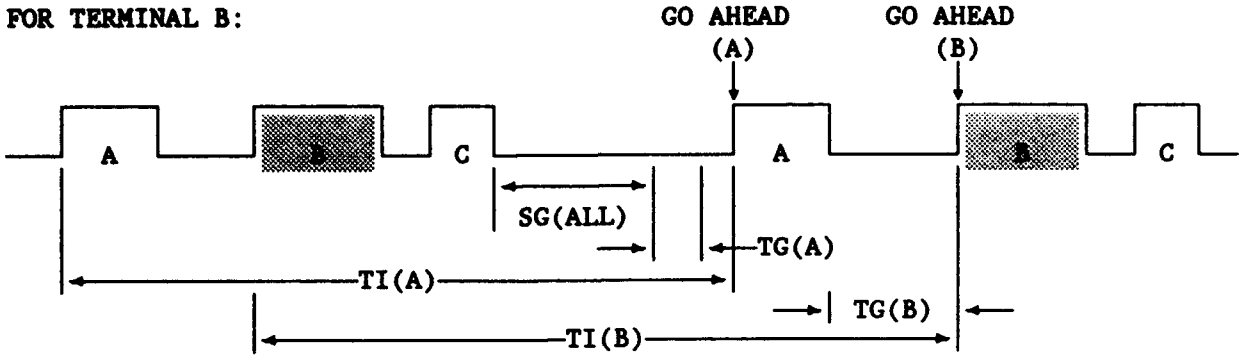
- Level 1 is periodic data (highest priority)
- Level 2 is aperiodic data (mid-priority)
- Level 3 is aperiodic data (lowest priority)

Level one data are sent first, followed by level two and level three. Periodic data are sent in level one in a continuous stream until finished, after which there should be time available for transmission of aperiodic data.

With this protocol there are three conditions which must be satisfied for proper operation. They are the occurrence of a TI, the occurrence of an SG, and the occurrence of a TG. These values are based on bus quiet time and are implemented as timers in each bus user. Figure 5.1-4 shows the access timing when the bus is operating in the periodic mode.

The TI defines the minimum period that a user must wait to access the bus. It is set to the same value for all users. In the periodic mode, it defines the update rate of every bus user. The SG is also set to the same value for all users and is defined as a bus quiet time greater than the largest TG value. The SG can take on four different values and is set larger than the greatest TG value. Every user is guaranteed bus access once every TI period. The TI and SG times are not reset by bus activity. The TG is a bus quiet time which corresponds to the unique address of a bus user. The TG, however, is reset by

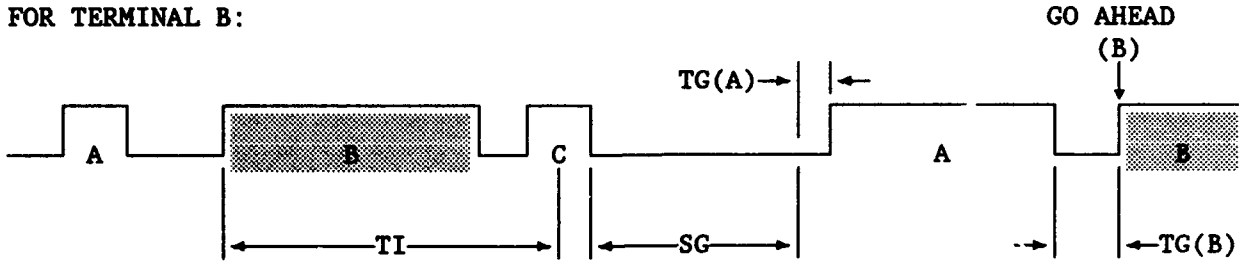
any bus activity. Once all three timers have expired for a user, it may access the bus.




TI is the controlling parameter.
TG prevents collisions due to clock drift.
SG is not a factor.

FIGURE 5.1-4. PERIODIC ACCESS FOR THREE BUS USERS
("ARINC 629 Symposium View Foils," 1991)

When a bus user or users exceed the time required for all transmissions to fit within the TI value, the protocol becomes aperiodic. During this overload condition, transmissions still continue but periodicity is not maintained. Figure 5.1-5 shows the access timing when the bus is operating in the aperiodic mode.




TG and SG are the controlling parameters.
TI is not a factor.

FIGURE 5.1-5. APERIODIC ACCESS FOR THREE BUS USERS
("ARINC 629 Symposium View Foils," 1991)

Since the TI value is exceeded, it is no longer the controlling parameter for bus access. The SG and TG now become the controlling factors and the TI is not a factor. This operation ensures all users bus access, although not at regular intervals.

According to Part 1 of the ARINC 629 bus specification, the system integrator is tasked with the selection of values for the TI, SG, and TG. Once the number of users is known, the range of TG values can be assigned and the SG and TI values determined. The TI is given by the following formula (ARINC Specification 629, Part 1, 1990):

$$TI = 0.5(\text{Binary Value of TG})_{10} + 0.5005625 \text{ ms}$$

When adding users to the bus it becomes necessary to review these bus parameters step-by-step, as was done in the initial design. Even if the bus capacity is not a problem, the values of the TG and SG may require modification if many users are added to the system. A recalculation of all timing parameters, along with changes in the hardware straps and Programmable Read-Only Memories (PROMs) for each user, may be required. The PROMs of all LRUs will also require updating if new labels are added to the bus.

Additionally, when more users are added, bus efficiency is reduced because of the increase in the TG required to address the new user. Adding user 126 to a bus consumes almost 128 microseconds every TI, whereas the addition of user 10 consumes only about 12 microseconds. One way to avoid problems when adding users is to maintain unassigned TGs with low values for this very purpose. If utilization of these TGs is planned from inception, then the integration impact will be minimal. Also, the SG value may need to increase when new users are added if the largest TG value approaches the value of the SG.

In a TDMA-based protocol with fixed time slots, overload of the bus is not possible since all users have access to the bus only in their own time slots. If variable length transmissions are permitted and a bus user sends data longer than is allotted, bus overload occurs. The data bus is still fully in use, but it becomes asynchronous and established update times for periodic variables are not met. This shift to the aperiodic mode is not detected by the bus hardware and needs to be implemented at a higher level for detection.

The ARINC 629 bus specification allows the use of variable length wordstrings and, therefore, the aperiodic mode is also defined in the specification. An ample amount of free time should be provided in the initial design to allow for integration of new users.

For the ARINC 629 bus, bus inactive time is measured and used by LRUs as a parameter in the access protocol. When a protocol is based on the bus inactive time, and the difference in inactive periods (which represent addresses) approaches bus propagation time, care must be exercised in the physical layout and address assignment of the individual users. Otherwise, a conflict may arise due to two users responding at the same time, both assuming they have access to the bus. To deal with this problem, section 4.2.1.3 of the specification states the following:

> "In general, for wire media, the total media length (stub/bus/stub) between terminals [bus users] with consecutive TGs should not exceed 60 meters." (ARINC Specification 629, Part 1, 1990).

This requires that LRUs and unassigned TGs be physically grouped so that this requirement is not violated. Also, any physical changes to the data bus that affect propagation parameters need to be considered carefully.

The ARINC 629 system designer or system integrator is tasked with many decisions concerning integration and operation of all the systems. The selection of the particular protocol mode, BP or CP, is one decision which must be made and which affects protocol complexity. If the CP mode is selected, then all LRUs must conform to whatever standard is proposed for that mode. If undefined areas exist in the bus specification, such as using the bus for file transfer, then the system designer is essentially tasked with completing the undefined sections and developing, testing, and implementing them as well.

#### 5.1.2.3 Command/Response Protocols

A command/response protocol is one in which a central controller manages all transmissions on the data bus. Bus users needing to send data are periodically addressed by the controller and given permission to access the bus for a specified message. No transmissions may be initiated without this permission.

An advantage of the centrally controlled architecture is that integration changes are carried out only in the active and standby controllers. Users do not require modification unless they are involved in the change. For further information on command/response protocols, see Elwell et al. (1992).

##### 5.1.2.3.1 High-Level Data Link Control Protocol

The High-Level Data Link Control (HDLC) protocol can operate in a command/response mode and is the basis of the ASCB data bus operation. HDLC was defined by the International Standards Organization (ISO) for the purpose of replacing character-oriented protocols.

HDLC is a bit-oriented protocol where data appears as a continuous stream of "ones" and "zeros." The beginning and end of the data bit stream are defined by using a flag at the beginning and end of the bit sequence. Once this is done it is referred to as a frame. Any information sent using the HDLC protocol uses the format shown in figure 5.1-6.

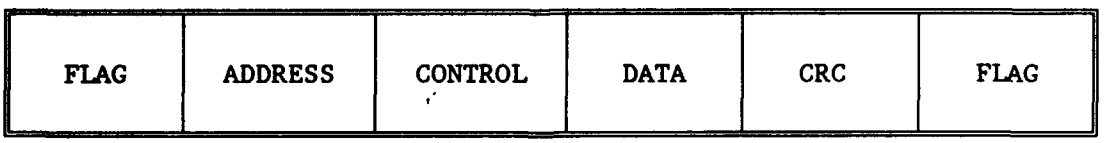



FIGURE 5.1-6. HDLC FRAME FORMAT

Operation of the HDLC protocol is described in terms of the capabilities of the bus users, or stations, and their cooperation. Intelligent stations can be connected to several very simple stations. The management of the bus, which requires more capabilities, is usually located in the more intelligent station.

This station is called a primary station while the others are called secondary stations (Meijer and Peeters 1982). When there is a primary station with more than one secondary station, it is referred to as an unbalanced configuration.

There are two modes by which the stations interact under the HDLC protocol: Normal Response Mode and the Asynchronous Response Mode. The Normal Response Mode specifies that the only time a secondary station can transmit is in response to a command or poll from the primary station. The Asynchronous Response Mode specifies that a station may transmit any time the bus is inactive. This applies to both primary and secondary stations. Operation in this mode means that collisions on the bus will be a normal occurrence.

In certain configurations it is necessary for all stations to have the same capabilities. In this case, each station will have the capacity to function as a primary or secondary station. This type of configuration is referred to as a balanced configuration.

Based on the bus user capabilities and response modes there are three classes of procedures defined in HDLC:

- Unbalanced Asynchronous Configuration (UAC)
- Unbalanced Normal Configuration (UNC)
- Balanced Asynchronous Configuration (BAC)

Further details of the HDLC protocol are given in appendix B of this chapter.

5.1.2.3.2 Avionics Standard Communications Bus

The ASCB is a centrally controlled, unbalanced implementation of the HDLC protocol using the Normal Response Mode. This configuration is the UNC. The ASCB message frame uses the leading flag, address, data, CRC, and terminating flag fields defined by the HDLC standard. Added to this are a checksum on the data field and "SYNC" and "MARK" fields at the beginning and end of the message, respectively. Figure 5.1-7 shows the frame format for the ASCB message.

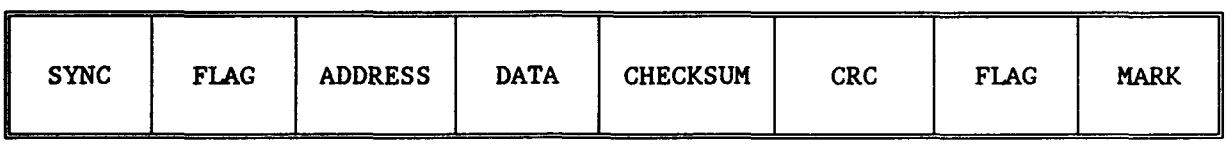



FIGURE 5.1-7. ASCB FRAME FORMAT

The ASCB eliminates the control byte, as defined in HDLC, from both the send and receive messages. The ASCB specification, however, defines a control word and a counter field in the control word (GAMA ASCB, section V, paragraph 4.3.1, 1987):

"Three bits are reserved in the control word of each user to implement a ... counter. This counter is incremented by the user each time it transmits."

This counter is used to verify that the data received is a new transmission and not the same data as the last transmission. Under HDLC, this field would be used in a store-and-forward network where a message may be broken up into smaller packets and sent to the destination, possibly over differing routes. The receiver would then be required to reconstruct the message in the order in which it was sent by using the three-bit counter field for correct ordering. Since there is only one route defined for ASCB users, this field is used as a data update indicator.

By not using the full implementation of the counter field as defined by HDLC, the ASCB protocol avoids the complexity of returning an acknowledgement frame to the sender for every message received. This is an important difference because, by doing this, the protocol is greatly simplified and proper operation is more easily verified and monitored. With the elimination of the control field, the HDLC information frames, supervisory frames, and unnumbered frames are also eliminated.

Since no acknowledgement is returned, there must be a way to recover from errors on the bus where data are lost and, therefore, nonrecoverable. Assuming all transmission errors are detectable, there are three basic ways to handle them:

- Use the last data received
- Request retransmission
- Use a stored or simulated value

Since retransmission is not a mode of operation for the ASCB, the system designer must choose one of the other methods for error handling.

Transmissions on the bus are considered valid messages by users if they contain a valid flag and address at the beginning and a valid flag and mark at the end. Anything else which appears on the bus is to be ignored by all bus users.

It may be difficult to make significant additions of users to an avionic system that uses the ASCB bus, since the message lengths are predefined and must fit within a 25 milliseconds cycle time. The use of a central controller, however, minimizes the difficulty since only the controller, and not all of the users, need to be updated when this is done. Since users only respond when they are addressed by the central controller, only the controller and any redundant controllers need their user lists updated.

There is no provision in the ASCB for separate handling of high priority data or messages. All transmissions are treated the same. If certain information on the bus is in higher demand by a bus user, the designer should ensure that it appears on the bus more frequently than other data. This can be accomplished by having a particular message sent every cycle time, as opposed to every other

cycle time. Since the ASCB uses a predefined cycle time, which is 25 milliseconds, the possibility of bus overload is nonexistent for this protocol.

#### 5.1.2.4 Token Passing Protocols

This protocol is based on a token, or special bit pattern, which circulates around a ring bus to each user. When a user receives the token it has exclusive access to the bus. When no users have messages to send, the ring is idle and the token circulates freely.

When a user wants to send a message, it waits until it receives the token. It functionally removes the token from the bus by altering the bit pattern. The message is then sent on its way around the ring along with the modified token, which is called a "connector."

If the tokens or messages were completely received and retransmitted by each user to the next user, it would be an inefficient protocol. The time to circulate a message around the loop would be the product of the time for one complete transfer multiplied by the number of users. Instead, the message is retransmitted bit-by-bit; each user only introduces a one-bit delay. This reduces the retransmission overhead to only one bit-time multiplied by the number of users.

When the number of users connected to the ring is large, the one-bit delay and propagation time become significant. If the ring is small, then the number of users is limited by the number of bits contained in the token. There must be enough users to allow the entire token to be placed on the ring. Another factor which requires consideration is when a user is removed from the ring. The number of users remaining active needs to equal or exceed the number of bits in the token. If a user is removed, it may be necessary for the interface logic to remain attached to the ring so that a one-bit delay is maintained.

A problem associated with this protocol is that if the token is ever lost, for instance by a noise burst modifying the token pattern, operation will cease. Users can monitor for this condition and, after a period of inactivity, start a token circulating again. If variable length messages are permitted, the timeout period needs to satisfy the worst case scenario, of all bus users transmitting the maximum length message, to avoid having more than one token in circulation.

Two avionic data buses that may be classified in the category of token rings are the LTPB and the HSRB. The LTPB uses a linear topology with token passing for access control. The HSRB uses a ring topology with token passing. Both buses operate at high data rates (50 megabits per second) and are designed primarily for military aircraft.

For further information on token passing protocols, see Elwell et al. (1992).

##### 5.1.2.4.1 Linear Token Passing Bus

The LTPB is a recently defined data bus. Two types of media are defined for use by the LTPB. They are fiber optic and wire media. Bus lengths of up to 1000

meters can be accommodated by the LTPB. Some of the bus characteristics are listed in table 5.1-1.

TABLE 5.1-1. LTPB CHARACTERISTICS

| | |
|---|---|
| Media | Fiber Optic or Wire |
| Word Size | 16 Bits |
| Message Size | 0 to 4096 Words |
| Number of Physical Addresses | 128 |
| Priority Levels | 4 |
| Topology | Linear |

Although the physical bus is linear, the protocol uses a token which is cyclicly addressed to each bus user, in sequence, around a logical ring. The token is a Token frame which consists of an address field and a frame check field. The address field contains the address of the BIU for which the token is intended. The frame check field is a CRC which ensures token integrity. The BIU that receives the token is granted access to the bus.

Since the token contains the destination address, any BIU may alter the sequence of BIUs by modifying this field. This feature allows a ring to easily reconfigure itself when an individual BIU becomes inactive. At power-up, or when the token is corrupt, the logical ring sequence is established by a predefined contention method.

If a BIU does not respond in a reasonable amount of time to a token passed to it, the sending BIU will again send the token to the same bus user. If there is no response, the sending BIU increments the destination address field of the token and again transmits it on the bus. This process continues until a successor is found or the destination address wraps around and equals the sending address (AS4074.1, 1988).

The addition of new members to the ring and reentering bus users that were momentarily dropped from the ring is accomplished by the Ring Admittance Timer (RAT). Each BIU maintains a RAT. When the timer expires, the BIU attempts to pass the token to a bus user with an address between its own and that of its current successor. A token is sent to the succeeding address two times. If no response is received, the address is increased by one and a new token is issued. This process continues until a successor is established. If the current successor already has the next physical address, the RAT is ignored. A RAT should be used only on a lightly loaded bus. The throughput of a moderately loaded bus would be significantly decreased (AS4074.1, 1988).

The LTPB allows message prioritizing. There are four categories of messages: priority 0 through priority 3. Priority 0 messages are the highest in priority, priority 3 messages are the lowest.

When a BIU receives the token, it sends all priority 0 messages first. A Token Holding Timer (THT) is maintained by the BIU to control the maximum amount of time the token may be held. It is reset upon reception of a token. The token is passed on to the next user when the THT expires, even if there are messages remaining to be sent.

Before the THT expires, Token Rotation Timers (TRTs) determine the window for sending messages with priorities of 1 through 3. Each BIU maintains a TRT for each of the three priority levels. After priority 0 messages are sent, priority 1 messages are sent until finished or until expiration of the priority 1 TRT. This procedure is followed for each message priority level. If all of the lowest priority messages are sent, the token is passed to the next bus user. The TRT and THT ensure small latencies for high priority messages (AS4074.1, 1988).

5.1.2.4.2 High Speed Ring Bus

The HSRB is another recently defined data bus. It is a unidirectional ring bus that sequentially passes the token from one bus user to the next to control bus access. The BIU that receives the token modifies the token, originates a message, removes the message when it returns around the loop, and then emits a new token. All other BIUs in the ring simply repeat messages that they receive. In the normal mode of operation, the BIU holding the token sends only one message before issuing a new token. Some of the bus characteristics are listed in table 5.1-2.

TABLE 5.1-2. HSRB CHARACTERISTICS

| | |
|---|---|
| Media | Fiber Optic or Wire |
| Word Size | 16 Bits |
| Message Size | 1 to 4096 Words |
| Number of Physical Addresses | 128 |
| Priority Levels | 8 |
| Topology | Ring |

A BIU connected to the HSRB has two functional parts: the Ring Interface Unit (RIU) and the Ring Interface Module (RIM). The RIM interfaces to the medium and either allows the RIU to be connected to the bus or isolates the RIU from the bus. The RIM has a mechanism to maintain ring continuity in the event of a bus

user failure. The RIU interfaces with the host CPU and performs the many protocol related tasks associated with the token passing protocol.

A maximum delay of six bit-times is permitted between the input and output of a RIU. All BIUs repeat the received messages, except the transmitting BIU which removes its own message from the ring and reissues the token.

The normal configuration for the HSRB is a dual ring configuration where one ring is active and the other is inactive. Each message is simultaneously transmitted on both rings, but the message on the inactive ring is normally ignored.

There are three types of frames defined for the HSRB: Token, Message, and Beacon. Token frames are used for access control, Message frames pass information among bus users, and Beacon frames transmit control information during start-up or reconfiguration.

5.1.2.4.2.1 Token Frame

The protocol uses a token which is continually passed from one BIU to the next around the physical ring. The token is a Token frame which consists of a Token Starting Delimiter Field (TSDF), a Control field, and a Token Frame Ending Delimiter Field (TFEDF). The TSDF and TFEDF establish the start and end of the Token frame. The Control field consists of five sub-fields relating to the network operation. This token, with no message attached, is called a free token. Only the BIU that receives a free token can access the bus. Refer to AS4074.2 (1988) for more detail.

5.1.2.4.2.2 Message Frame

A Message frame is the vehicle used to transfer information from one bus user to another. It consists of a Claimed Token sub-frame, which is the free token with the Token Ending Delimiter stripped off, followed by a Preamble field, various address and message control fields, an Information field, and a Frame Status field (Aerospace Information Report [AIR] 4289, 1990).

With the exception of the Information field, all other fields in the Message frame are considered overhead. They are used to ensure error free message delivery to the destination and correct protocol operation.

If a bus user wishes to send a message, it may wait for a free token and then claim it, as long as it has higher priority messages to send than may be reserved. Otherwise, the user can make a reservation in the Claimed Token sub-frame of a message already circulating around the ring.

Reserving a free token is done by setting the appropriate priority bits in the Control field of the Token frame according to the priority of the message the bus user wishes to send (AS4074.2, 1988). If the field is already set to a lower priority, the bus user replaces the previous reservation with its higher priority reservation. If the field is already set to a higher priority, then the bus user wishing to place a reservation for a lower priority message must

wait until the priority field of a Claimed Token is lower than its message priority level.

After the message is passed completely around the ring, the sender removes the message from the ring and issues a free token with the priority field set to the priority of the Claimed Token which it removed (AIR 4289, 1990). The first user that has a message of that priority or higher may claim the reserved free token.

The bus standard allows for an option where it does not require the last issued Claimed Token to be received by the sending bus user before it issues a new free token. Operation in this manner allows multiple short messages on the ring simultaneously. This mode of operation does not guarantee that the highest priority message will be serviced first. Therefore, the bus standard limits the number of consecutive short messages to 16.

5.1.2.4.2.3 Beacon Frame

The Beacon frame is used during ring reconfiguration to transmit control information to all BIUs. Reconfiguration occurs after the application of power, or during error recovery, and establishes the master station. The master station issues the first free token after reconfiguration. The master station is normally the highest addressed bus user. There is only one master station, the rest are slaves. All bus users, however, must have master capability. The reconfiguration also determines the number of participating bus users.

Since the ring allows reconfiguration at power-up, new bus users can be added simply by attaching them to the ring and applying power. This assumes that the required loading analysis has already been performed, the addresses of receiving or transmitting BIUs have been implemented in the new BIU, and any other required changes have been made.

A bus user may be a bus bridge, as in any network. A bridge functions as a receiving BIU on the ring which originates the message and as a transmitting BIU on the ring which is to receive the message. The bridge completely receives the message, verifies its correctness, and acknowledges receipt, before the message is passed to the receiving BIU. The HSRB standard includes examples of bridge implementations and guidelines for the designer, along with guidelines on handling the protocol acknowledgement through the bridge.

5.1.3 Data Integrity Concerns

The integrity of data in an integrated digital avionic system is a key concern of the user. Hence, it needs to be a key concern of the designer and systems integrator. Problems arise in the use of data buses when they are pushed beyond their designed limits, causing a bus overload condition. Another cause of concern is due to bus faults induced by internal or external sources. An internal source may be a faulty bus user, while an external source may be radiated noise. Some issues relating to data integrity are examined in the following sections. Applications to avionic data buses are made.

### 5.1.3.1 Bus Capacity

Bus capacity deals with the ability of a data bus to handle its load. A data bus is used to deliver information in a safe and timely manner. If data are not available for a computation when they are needed, the system requiring the data will yield results that are less than desireable. Since avionic systems are becoming increasingly complex and more integrated, there is a growing need to pass more variables between systems on a particular data bus. This need is driven by various factors such as cost savings, performance improvements, and pilot workload reduction.

When an avionic system is designed, the designers and system integrators ensure that there is ample free time on the data bus to handle all of the bus traffic during the worst case condition. Draft 1 of ARINC Specification 629, Part 2, section 4.1 (1989), states the following:

> "... bus capacity is a finite resource and should be utilized in a conservative manner. Therefore, it is recommended that the system designer exercise diligence in the design process."

When new LRUs are added to an existing system at a later date, is the designed worst case loading known? If it is, is it accounted for in the modified system? Does the new integrated system meet the original design specification and update rate for all variables?

In practice, the theoretical maximum bus capacity is limited by several factors:

- Bit rate
- Message format
- Protocol
- Architecture

The bit rate, or clock rate, of a data bus is only one factor relating to bus capacity. It quantifies the number of bits per second that are transferred across a data bus. If the clock rate of a data bus is 1 megabit per second, and the word size is 20 bits, then the theoretical throughput is 50,000 words per second. An increase in the bit rate will yield an increase in throughput. There are physical limitations, however, that dictate the maximum bit rates for a given configuration. The maximum bit rate depends on factors such as the following:

- Bus medium used (wire/fiber optic)
- Physical characteristics of the medium
- Bus interface logic device speed

The message format has a pronounced affect on bus capacity. Error detection and correction bits add overhead to the basic data word. In addition, if more data

bits are defined than are necessary for a particular variable, or if fields in a wordstring are defined but not used, then overhead is increased, which reduces the data bus capacity.

The particular protocol used also has an effect on the bus capacity. When a protocol which requires an acknowledgement is used, the response time of the receiver and the transmission time of the acknowledgement must be accounted for. If the response time of the receiver varies, the worst case response should be specified and used in the calculation of throughput.

In an acknowledgement type protocol, consideration should be given to the additional load created by error correction. Upon detection of an error, retransmission may be requested. Retransmissions may force the bus to become aperiodic. Hence, it is necessary to plan for a certain amount of retransmissions in an acknowledgement-based protocol.

A protocol that bases bus access on a delay time also influences throughput. In effect, a certain delay time becomes the unique user address. The higher the number of users on the bus, the lower the overall bus capacity.

Bus architecture has an influence on bus capacity. If a system contains gateways or bridges, the resulting delays need to be accounted for. These delays may be in the form of error checking, protocol conversion, data format conversion, or other operations on the data performed in a gateway or bridge.

Not considered in this section, but significant to overall system performance, are buffer availability in the receiver; the processing capability of the receiver; the ability of the sender to maintain the required update rate; and other areas not directly related to bus throughput, such as hardware-software interaction. It should be recognized that due to problems in these areas bus performance may be degraded, but that the cause is not the data bus.

5.1.3.1.1 ARINC 429 Bus Capacity

The ARINC 429 bus uses a word length of 32 bits. There are two transmission rates: low-speed, which is defined as being in the range of 12.0 to 14.5 kilobits per second; and high-speed, which is 100 kilobits per second.

There are two modes of operation in the ARINC 429 bus protocol: character-oriented mode and bit-oriented mode. In the character-oriented mode, periodic updates of each variable are maintained on the data bus. These periodic rates are specified in Attachment 2 of the specification, along with the message labels, equipment identifiers (IDs), and other essential information. Knowing the bit rate and the essential update information from Attachment 2, a determination of bus capacity can be made.

For the bit-oriented mode, the determination of throughput is complex and is based on numerous protocol related variables. The bit rate specified is the same as for the character-oriented protocol and can be either low- or high-speed. Bus capacity is difficult to determine when the bit-oriented mode is used, and no guidelines are given in the specification for making this determination.

Bus utilization remains constant during operation of the character-oriented protocol. The system designer defines the load based on the LRU messages and update rates required for all LRUs and selects the appropriate bus speed to support the update rate required. No guidance is given in the specification for overhead allowance.

Since the ARINC 429 bus is a broadcast bus, no access protocol is used by transmitters on the bus. Bus availability is not a problem for a bus with a single transmitter. There is, therefore, no access protocol overhead limiting bus capacity. There is a message format overhead, but it is minimal.

Out of the 32-bit word length used, a typical usage of the bits would be as follows:

- Eight bits for the label
- Two bits for the Source/Destination Identifier
- Twenty-one data bits
- One parity bit

Thus, the information bit-rate for the ARINC 429 bus is typically a factor of twenty-one thirty-seconds of the clocked bit-rate. An information bit-rate of 65,625 bits per second is the maximum obtainable rate with the given overhead.

5.1.3.1.2 Commercial Standard Digital Bus Capacity

The CSDB is similar to the ARINC 429 data bus in that it is an asynchronous broadcast bus and operates as a character-oriented protocol. Two bus speeds are defined in the CSDB specification. A low-speed bus operates at 12,500 bits per second and a high-speed bus operates at 50,000 bits per second.

Data are sent as frames consisting of a synchronization block followed by a number of message blocks. A particular frame is defined from the start of one synchronization block to the start of the next synchronization block. A message block contains an address byte, a status byte, and a variable number of data bytes. The typical byte consists of one start bit, eight data bits, a parity bit, and a stop bit.

The theoretical bus data rate for the CSDB operating at 50,000 bits per second, with an 11-bit data byte, is 4,545 bytes per second. The update rate is reduced by the address byte and synchronization block overhead required by the standard.

The CSDB Interblock and Interbyte times also reduce the throughput of the bus. According to the specification, there are no restrictions on these idle times for the data bus. These values, however, are restrained by the defined update rate chosen by the designer. If the update rate needs to be faster, the Interblock time and the Interbyte time can be reduced as required.

5.1.3.1.3 ARINC 629 Bus Capacity

The current draft of ARINC Specification 629, Part 2, section 4, is entitled "Bus Performance Analysis." The draft treats "Bus Loading" and will also include sections on "Response Times" and "Data Latency."

Bus loading is discussed for the CP protocol. There are three levels of bus data traffic defined in the CP protocol. The first level consists of all periodic transmissions. Loading due to periodic traffic is evaluated before level two and level three traffic. The specification states the following:

> "Normally a worst case estimate can be obtained by simply summing the maximum bus loads, after balancing has been attempted, of all terminals [bus users] attached to the bus." (ARINC Specification 629, Part 2, section 4, 1989).

The standard recommends that bus traffic be balanced before computing the bus loading. This means that the designer should attempt to even out the traffic load to minimize the worst case load. After this, a simple summation of the maximum loads presented by each LRU will give the level one bus loading.

In contrast to level one loading, level two and level three loading is much more difficult to ascertain. Aperiodic traffic depends on the flight phase or the mode in which the aircraft is operating. According to the specification, the designer will need to make several evaluations:

- The average load presented by the identified traffic.
- A worst case assessment, if transmission of level two messages within one TI is to be guaranteed.
- A less severe case, where transactions are triggered by some event.
- A statistical evaluation of bus loading.

Level three traffic is aperiodic and lowest in priority. For this level, the specification requires that the designer make the following assessments:

- The average load presented by the identified traffic.
- A worst case assessment, taking into account that some transfers may result in closely spaced bus accesses and may use a file transfer protocol.

Since the ARINC 629 bus protocol is based on bus quiet times for operation, it suffers throughput degradation when high periodic update rates are required with a large number of users. Three factors which contribute to this degradation are the Interstring Gap (IG), the TG, and the SG.

IGs are required time intervals inserted between contiguous messages. A TG is a unique time interval for each bus user which must be satisfied for a user to access the bus. The SG is a time interval greater than the largest TG that must be satisfied for each user before a user can access the bus.

Another factor contributing to throughput degradation is a small number of data words per message. The larger the message, the more efficient the data transfer becomes. However, to allow periodicity for all users means that some trade-offs must be made between the frame rate for all users and the number of words per label. A large number of users and high periodic update rate also detract from the protocol performance.

On the other hand, greater performance can be realized if the following guidelines are followed:

- Use as many words per label as is practical.
- Choose reasonable values for the periodic update rate and the number of users.
- Keep the TG values sequential and choose the smallest set possible.
- Place the most used data words in a message closest to the label to enhance receiver performance.

Concerning the initially designed bus capacity, section 4.4.6 of ARINC Specification 629, Part 1 (1990), states the following:

> "During initial development, bus loading should not exceed 50% of its capacity in order to allow for growth during the system's operational life."

The designer is cautioned that capacity is a finite resource and should be used conservatively.

5.1.3.1.4 Avionics Standard Communications Bus Capacity

Data are sent on the ASCB as a series of eight frames, each with a duration of 25 milliseconds. There are no retransmissions or complicating protocol factors. The computation of bus capacity is straightforward. The messages transmitted in each frame are predetermined for a particular application, and there is no deviation once the operation is established.

The ASCB standard gives the following information for computing the bus capacity:

- Bus clock rate = 2/3 MHz = 0.0016 ms/bit
- Zero insertion factor = 6/5 x 0.0016 = 0.0018 ms/bit (The HDLC component automatically inserts a "zero" to prevent six consecutive "ones." The receiving HDLC component automatically removes the inserted "zeros.")
- 8 bits/byte x 0.0018 ms/bit = 0.0144 ms/byte, or 69,444 bytes/second

Bus utilization remains constant for ASCB during operation. The system designer defines the throughput based on the LRUs and update rates required for all

systems, and based on the byte rate defined above. Overhead is also added to allow for future expansion. For a typical application, the ASCB utilizes approximately 80 percent of the available frame time (Jennings 1986).

5.1.3.1.5 MIL-STD-1553 Bus Capacity

For the MIL-STD-1553 bus, messages are passed between a BC and remote terminal (RT), which is a bus user, one RT and another RT, or one BC and another BC. Calculating bus capacity is viewed as a fairly simple task. According to the "MIL-STD-1553 Designer's Guide" (1982), the following are required for the computation:

- A hand-held calculator
- System data
- Decisions on the implementation of MIL-STD-1553

The "MIL-STD-1553 Designer's Guide" (1982) suggests the following values be used when computing bus loading:

- 20N + 68 - value for each BC to RT message
- 20N + 116 - value for each RT to RT message
- 20N + 40 - value for each BC to RT broadcast message
- 20N + 88 - value for each RT to RT broadcast message
- 68 - value for each mode code (MC) message without data word
- 88 - value for each MC message with data word
- 40 - value for each MC broadcast message without data word
- 60 - value for each MC broadcast message with data word

The value "N" represents the number of words in the message and the values calculated are in milliseconds. The average bus loading is given by the following:

Bus Loading = ( S / F ) x 100 percent

where S is the sum of the message type values and F is the frequency, 1.0 megahertz.

The "MIL-STD-1553 Designer's Guide" (1982), section I, paragraph 3.8, also makes the following recommendation concerning bus capacity:

> "A system should not exceed 40% bus loading at initial design and 60% at fielding, in order to provide time for error recovery/automatic retry and to allow growth during the system's life."

#### 5.1.3.1.6 Linear Token Passing Bus Capacity

Before a determination of the bus capacity can be made, it is necessary to calculate the token rotation time and categorize the traffic into message types and priorities. Clear and ample direction for these determinations is given in the LTPB user's handbook (AIR 4288, 1991). In addition, the handbook gives examples to aid the system designer or integrator in this task.

The token rotation time is calculated as follows (AS4074.1, 1988):

$$T = N\,(\text{Bus Length} / \text{Propagation Speed} + \text{Token Receiving Time} + \text{Token Transmitting Time})$$

where

T is token rotation time in seconds
N is the number of BIUs in the configuration
Bus Length is the distance from the transmitting BIU to the receiving BIU

The Token Receiving Time and Token Transmitting Time are equal since they both contain the same number of bits and have the same clock rate. This time is given by the following (AS4074.1, 1988):

$$(\text{Preamble Size} + \text{Token Length}) * \text{Bit Time}$$

The preamble is a bit pattern created by the transmitting BIU. It is used by the receiver to synchronize its receive clock to the clock of the transmitting BIU. The system designer has the liberty to set this value according to the requirements of the receiving hardware. It must be accounted for in computing the bus capacity.

There are four important characteristics of the bus traffic to quantify: types of messages, data message size, peak frequency, and latency. The message type describes a unique combination of message size and frequency. The data message size is the number of 16-bit words associated with the message type. The peak frequency defines the update rate for a message type. Latency is derived from the peak frequency and is used as a basis for a message's priority. Table 5.1-3 gives an example of how messages may be characterized.

The messages need to be assigned priority. The LTPB user's handbook suggests, for simplification, that the larger latencies be multiples of the smallest latency. Using this criterion, table 5.1-4 gives the priority breakdown, using the data from table 5.1-3.

TABLE 5.1-3. LTPB MESSAGE CHARACTERISTICS
(AIR 4288, 1991)

| Message Type | Data Message Size (Words) | Peak Frequency (Hz) | Latency (ms) |
|---|---|---|---|
| A | 20 | 100 | 10 |
| B | 50 | 75 | 10 |
| C | 50 | 50 | 20 |
| D | 150 | 50 | 20 |
| E | 20 | 25 | 40 |
| F | 225 | 25 | 40 |
| G | 1025 | 15 | 66 |
| H | 1000 | 12.5 | 80 |
| I | 150 | 12.5 | 80 |
| J | 2000 | 10 | 100 |

TABLE 5.1-4. LTPB MESSAGE PRIORITIES

| Message Type | Priority | Latency |
|---|---|---|
| A | 0 | 10 |
| B | 0 | 10 |
| C | 1 | 20 |
| D | 1 | 20 |
| E | 2 | 40 |
| F | 2 | 40 |
| G | 2 | 40 |
| H | 3 | 80 |
| I | 3 | 80 |
| J | 3 | 80 |

The time for a single transmission of all messages for each priority category is calculated from the following equation:

$d_{ij}$ = (Number of priority i words * 16 bits/word
+ Message overhead bits * Number of priority i messages)
* Bit time

In table 5.1-3, priority 0 messages have 70 words. If an overhead of 27 bits exists, and the bit rate is .02 microseconds per bit, then the transmission time for priority 0 messages is as follows:

(70 * 16 + 27 * 2 ) * .02 = 23.48 microseconds

The calculated values for priority 0 through priority 3 messages are 23.48 microseconds, 65.08 microseconds, 408.02 microseconds, and 1009.62 microseconds, respectively. Since there are 10 bus users, the total transmission time for each priority group becomes 234.8 microseconds, 650.8 microseconds, 4080.2 microseconds, and 10096.2 microseconds, respectively.

Next, the number of token rotations necessary to service the messages for each priority level of bus traffic is used to determine the expected bus loading. It is assumed that in one token rotation, all priority 0 traffic will be passed; in two, all priority 1 traffic will be passed; in three, all priority 2 traffic will be passed; and in four, all priority 3 traffic will be passed. The expected bus loading is then calculated by dividing the total transmission time for priority "i" messages by the number of token rotations necessary to pass priority "i" messages. Using the same data, the values of 234.8 microseconds, 325.4 microseconds, 1360.07 microseconds, and 2524.05 microseconds are obtained (AIR 4288, 1991).

If the total bus loading for priority 0 through priority 3 traffic, plus the token rotation time, is less than the required priority 0 latency, then sufficient bandwidth exists to support the network operation. In this example, the time for 10 bus users to pass their expected traffic is 4444.32 microseconds (234.8 + 325.4 + 1360.07 + 2524.05). If the token rotation time (45.5 microseconds for this example) is added, then the total becomes 4489.82 microseconds. Since the priority 0 latency requirement is 10 milliseconds, or 10,000 microseconds, this bus configuration is adequate and allows ample bandwidth for growth.

It is possible to enhance bus performance by requiring that successive bus users be located adjacent to each other. This will reduce the token passing time and the time for detection of the successor. Since the performance increase is proportional to the bus medium length, the effect is more dramatic in larger configurations (AIR 4288, 1991).

5.1.3.1.7 High Speed Ring Bus Capacity

The HSRB handbook includes a section on "Performance Calculation" for the HSRB. To proceed with this analysis, it is first necessary to compute the Ring Rotation Time (RRT). This is given by the following (AIR 4289, 1990):

RRT = Media Transmission Delay
+ Bus User Bit Delay
+ Bus User Modulation and Demodulation Delay

where

Media Transmission Delay = $L / V$
Bus User Bit Delay = $[ ( N - 1 ) * S_b + M_b ] / C_k$
Bus User Modulation and Demodulation Delay = $N * ( T_m + T_d )$

and

$L$ = Length of medium
$V$ = Transmission velocity of medium
$N$ = Number of bus users on ring
$S_b$ = Number of delay bits in a slave station BIU
$M_b$ = Number of delay bits in a master station BIU
$C_k$ = Clock rate
$T_m$ = Modulation delay in a BIU
$T_d$ = Demodulation delay in a BIU

The HSRB standard specifies $S_b$ and $M_b$ to be 6 bits and 40 bits, respectively, and $C_k$ as 50 Megabits per second when using wire media. Using a value of 150 meters/microsecond for V, 0.05 microseconds for $T_m + T_d$, 64 for N, and 300 meters for L, then the RRT can be computed as follows:

$$RRT = 300 / 150 + [ (64 - 1) * 6 + 40 ] / 50 + 64 (0.05)$$
$$= 13.56 \text{ microseconds}$$

The Message Length (ML) is also necessary for the performance computation. This is computed in the following manner (AIR 4289, 1990):

$$ML = \text{Overhead bits} + \text{Information bits}$$
$$= 170 + 20 * L_n + 40 * INT(I_n/256) + 20 * I_n$$

where

$L_n$ = Number of logical address words (equal to one if physical addressing is used)

$I_n$ = Number of Information words

If we use a value of $L_n = 1$, then ML may be calculated over the range of information words as shown in table 5.1-5.

TABLE 5.1-5. HSRB MESSAGE LENGTH VERSUS INFORMATION WORDS

| Information Words | Overhead Bits | Information Bits | Message Length |
|---|---|---|---|
| 1 | 190 | 20 | 210 |
| 1024 | 350 | 20480 | 20830 |
| 2048 | 510 | 40960 | 41470 |
| 3072 | 670 | 61440 | 62110 |
| 4096 | 830 | 81920 | 82750 |

Once the ML is known the Message Time (MT) can be calculated as follows:

$$MT = ML / C_k$$

The corresponding MTs for 1, 1024, 2048, 3072, and 4096 Information words are 4.2, 416.6, 829.4, 1242.2, and 1655 microseconds.

For the case of a single bus user transmitting a message, the worst case transmission time, which occurs when the BIU has just missed claiming the token, is given by the following:

Worst case transmission time for single transmitter = RRT + MT

The percent efficiency of the ring is now calculated as follows:

Efficiency = 100 * Information Time / Total Time

where the Information Time is computed as follows:

Information Time = Number of Information Bits ($I_b$) / $C_k$

Table 5.1-6 shows the relationship between the number of Information words and the efficiency of the ring. Note the dramatic loss of efficiency with a small number of Information words.

TABLE 5.1-6. HSRB EFFICIENCY VERSUS INFORMATION WORDS

| $I_n$ (words) | $I_b$ (bits) | MT ($\mu s$) | RRT ($\mu s$) | Efficiency (%) |
|---|---|---|---|---|
| 1 | 20 | 4.2 | 13.56 | 2.25 |
| 1024 | 20480 | 416.6 | 13.56 | 95.20 |
| 2048 | 40960 | 829.4 | 13.56 | 97.20 |
| 3072 | 61440 | 1242.2 | 13.56 | 97.90 |
| 4096 | 81920 | 1655.0 | 13.56 | 98.20 |

5.1.3.2 Data Bus Fault Tolerance

Fault tolerance deals with the ability of a system to operate in the presence of errors. Of primary importance here is that errors are detected. Once an error is detected, it may be dealt with in numerous ways. In this section, methods of error detection and correction, bus monitoring, and bus reconfiguration are examined along with specific examples from the data bus standards. Fault tolerant techniques relating to the hardware-software interface are discussed in section 5.2.

5.1.3.2.1 Bit Error Detection and Correction

Errors on any transmission medium are a fact of life. They are caused by events beyond our control or that are too expensive to control. In either case, the designer is faced with how to handle errors induced by sources outside the

system as well as internal sources, such as equipment and transmission medium failure. If the designer has some knowledge of the nature of the errors that will likely occur, then the task of implementing error detection becomes precise and efficient.

There are four common methods for detecting bit errors in data: a parity check, CRC, Checksum, and Hamming code. The Hamming code not only detects errors but can be used to correct errors. For further details on these methods, see Elwell et al. (1992).

Whatever method or methods are used by a data bus for error detection or correction, it is important that the bus standard be precisely observed. Designers and integrators should ensure that all transmitters and receivers agree on the format of error detection (odd/even parity, CRC polynomial generator, etc.) and that the specified checking is implemented. Error checking implemented in data bus hardware is only the first step in ensuring data integrity. The application software in the host CPU must respond to the detected error or else the hardware checking is useless.

As mentioned in section 5.1, address errors are a major concern for systems using data buses. An address field error caused by data bus induced noise, for example, can have a more profound affect on system operation than an error in the data field. A corrupt address field could cause an LRU to receive a message or command that was intended for another LRU. Although it is possible to "smooth over" errors in a data field by filtering, such an operation does nothing to protect the address field. The address field should also be protected by a high integrity check.

Additionally, since standards are very specific in defining each bit for the data word, all defined fields of a word should be checked to ensure that they are both valid and reasonable. Spare bits should be defined and fixed as either ones or zeros and checked by the receiver. A data bit field that has additional constraints placed on it by the standard should be checked by the receiver for compliance. For instance, when a field of eight bits may have only one bit in the logic "one" state at a time, all the rest should be tested to see if they are "zero."

5.1.3.2.1.1 ARINC 429 Bus Error Detection and Correction Methods

ARINC Specification 429-12, section 1.3.1 (1990), states the following:

> "A parity bit is transmitted as part of each data word to permit simple error checks to be performed by the sinks"

This data bus relies on parity bit error detection with each data word. By itself, this amount of protection does not seem adequate. In a data word of 32 bits, bit errors may occur in any even number of bits up to 32 without being detected by the parity technique. However, experience has shown that the high integrity of the twisted and shielded wire transmission medium and the slow signaling rate have ensured reliable bus transmissions.

An additional technique referred to in the standard is the "data reasonableness" check. This means that the host computer at the data destination must have information about the data it expects to receive. If no errors are indicated by the data bus hardware, the CPU tests the data to ensure that it is within anticipated reasonable bounds. This type of checking can be performed at the host CPU as an additional integrity check on data that is passed over the data bus.

Data filtering is also used to reduce the effects of data that may be within reasonable bounds but is incorrect. The incorrect value is smoothed out by averaging it with the preceding and following values.

ARINC Specification 429-12 also defines a bit-oriented protocol used for file data transfer. This protocol uses "handshaking" between communicating users, and also defines a CRC to be used for the file transfer. The use of the CRC ensures that a certain amount of errors occurring in the data will be detected. Since the file data are not refreshed regularly or continuously as are other data, reasonableness checks and filtering are not possible. Thus, the CRC was added to ensure additional data integrity. The generator polynomial used is as follows:

$$G(x) = x^{16} + x^{12} + x^5 + 1$$

5.1.3.2.1.2 CSDB Error Detection and Correction Methods

Two methods of error detection are referenced in the standard. They are the use of parity and checksums.

A parity bit is appended after each byte of data in a CSDB transmission. In section 2.1.4 of the standard, three types of transmission are defined:

- Continuous repetition
- Noncontinuous repetition
- "Burst" transmissions

The "burst" transmission makes use of the checksum for error detection, as the specification states:

> "It is expected that the receiving unit will accept as a valid message the first message block which contains a verifiable checksum." (GAMA CSDB, 1986).

5.1.3.2.1.3 ARINC 629 Bus Error Detection and Correction Methods

ARINC Specification 629, Part 1 (1990), recommends multiple levels of error detection. At the lowest level, parity is defined as part of the 20-bit data word definition. Section 4.4.2 of the specification states the following:

> "The last bit of each label and data word should be encoded such that word parity is rendered odd."

Section 6.5.1 of the specification allows two other options for error detection. They are the use of the checksum and CRC. The checksum is a 16-bit word formed by adding, without carry, the 16 least significant bits of all the data words prior to the check word.

The other option is to use a CRC. The same generator polynomial used by the ARINC 429 bus is recommended.

Although the checksum and CRC may be calculated by the host CPU, the time used to perform these calculations differs. The CRC is more complex than the checksum and the standard suggests that "dedicated hardware" may be required for this calculation. Calculation of the checksum, however, is relatively simple and is usually performed by software.

5.1.3.2.1.4 ASCB Error Detection and Correction Methods

Transmissions on the ASCB are initiated by a central controller. The ASCB uses both the CRC and the checksum. Each transmission of the ASCB has the CRC code appended to it by the HDLC hardware. This CRC is the same one used by the ARINC 429 bus. A checksum computed by the host CPU is added to every transmission and is sent as part of the user's data transmission (Jennings 1986).

5.1.3.2.1.5 MIL-STD-1553 Bus Error Detection and Correction Methods

This bus makes use of various error detection schemes. The command, data, and status words are checked using a parity bit in position 20. In addition, the "MIL-STD-1553 Designer's Guide" (1982) states the following:

> "... the traditional methods of computer data protection can be applied. These include checksums and cycle redundancy checks."

If further error protection is required, the "Multiplex Applications Handbook" (MIL-HDBK-1553A, 1988) recommends the use of a Hamming code protection method. The recommended method allows for data correction of up to three bits per word.

5.1.3.2.1.6 Linear Token Passing Bus Error Detection and Correction Methods

A Frame Check Sequence (FCS) is used on the LTPB for error detection. It is a CRC applied to Token frames and Message frames. The Token Frame Check Sequence (TFCS) is applied to the token and ensures that a bus user will not accept a token that has been corrupted. The Message Frame Check Sequence (MFCS) is applied to a message and ensures that a bus user will not accept a corrupt message. When a message with a CRC error is received, the receive buffers are cleared and the host is not notified of the message or the error (AIR 4288, 1991).

The LTPB uses a CRC generator polynomial of $x^8 + x^4 + x^2 + x + 1$ for the TFCS and a CRC generator polynomial of $x^{16} + x^{12} + x^5 + 1$ for the MFCS.

#### 5.1.3.2.1.7 High Speed Ring Bus Error Detection and Correction Methods

An FCS is used on the HSRB for error detection; it is a CRC applied to Beacon frames and Message frames. The Token frame does not have bit error detection applied to it as it does for the LTPB. However, when a host claims a token, the Token frame is modified and becomes part of the header of the Message frame. The Message frame has a field called the Message Control Frame Check Sequence (MCFCS), which is applied to the entire header of the Message frame. The Information field of the Message frame has its own FCS, the Information Frame Check Sequence (IFCS), applied to it to ensure that a bus user will not accept corrupt data. The Beacon Frame Check Sequence (BFCS) field is used to provide bit error detection for the Beacon Control field (AIR 4289, 1990).

The MCFCS, IFCS, and BFCS use the same CRC generator polynomial, $x^{16} + x^{12} + x^5 + 1$.

### 5.1.3.2.2 Bus Monitoring

Bus monitoring is a necessary function. The requirement for monitoring increases with bus complexity. Old methods of locating faults in analog systems will not work for digital data transmission. Placing an oscilloscope on a transmission line will indicate only that there is activity on that line. It will not indicate the origin of the activity or if the activity is correct.

One of the motivating factors behind bus monitoring is data integrity. When the designers are finished, the simulations are all run, and the firmware programmed and running in the target system, how can the functional system be validated against the requirements? If the system is working correctly under the present configuration, will it work the same under a slightly different configuration? Under normal use, how can the state of the system be determined?

There are two types of bus monitoring to consider. One type is performed by the data bus users to ensure bus communications. The other is usually performed by a dedicated bus monitor for the purpose of collecting maintenance data.

#### 5.1.3.2.2.1 Bus User Monitoring

User monitoring is performed by the bus interface of each user and should make the following checks:

- Protocol checking
- Received data monitoring
- Transmit data monitoring
- Host system interaction monitoring
- BIU hardware checking

The amount of bus monitoring used for integrity purposes by avionic buses varies greatly. With some data buses only a minimal implementation is made. Bus

standards should clearly define integrity issues and what parameters are to be monitored to ensure integrity. For instance, if a standard does not specify that buffer overrun errors shall be detected by all users, IC manufacturers or system designers might implement this checking due to cost factors. Monitoring should be planned at the beginning of the design, not added as an afterthought. It will exist only if it is intentionally planned and designed.

For data buses used in essential or critical systems, the designers should implement monitoring of any integrity related parameter. With current technology this is not a burdensome task. The following list contains some of the parameters which should be monitored by a bus user:

- Physical layer signal monitoring (Manchester modulation, parity, synchronization patterns, voltage margins, frequency of bad data due to collisions, HERF noise, or other interference)
- Local protocol monitoring (acknowledgements, timeouts, access denial, etc.)
- Self-monitoring for transmission validation (correct address, correct message format, "babbling" transmitter)
- Received data transmissions
- Transfer of data to and from host CPU

In addition to monitoring these parameters, provisions need to be made for reporting any errors to the host CPU. Without the capability to report an error, the ability of the user to detect it is useless.

5.1.3.2.2.2 ARINC 429 Bus User Monitoring

Since the ARINC 429 bus is older and unidirectional, the amount of bus monitoring by the user hardware is significantly less than the newer, bidirectional data buses. Additionally, being a unidirectional bus there are fewer parameters for the bus interface to monitor.

For periodic messages the specification requires simple parity checks, not so much for integrity as for compatibility with the hardware requirements (ARINC Specification 429-12, 1990).

> "...the parity bit was added to the BCD word for reasons related to BCD/BNR transmitter hardware commonality, not because a need for it existed for error detection."

The bit-oriented protocol requires that a CRC check be made on transfers. Other higher level protocol parameters should be monitored by the host CPU. Further checks may be performed by the host CPU only if the bus interface hardware includes the functional capability to do so. For example, if buffer overrun detection is not implemented in the hardware, the host CPU cannot detect this error.

5.1.3.2.2.3 Commercial Standard Digital Bus User Monitoring

Although many parameters are defined in the CSDB specification, there is no suggestion that they be monitored by receivers. The bus frame, consisting of the synchronization block and message block, may be checked for proper format and content. A typical byte, consisting of start, stop, data, and parity bits, may be checked for proper format.

The bus hardware should include the functional capability to monitor these parameters. Parity, frame errors, and buffer overrun errors are typically monitored in the byte format of the character-oriented protocols. The message format can be checked and verified by the CPU if the hardware does not perform these checks.

5.1.3.2.2.4 ARINC 629 Bus User Monitoring

Since this data bus is an autonomous access bus, self-monitoring becomes an even more important function for bus users. There are three distinct areas of monitoring defined for a user: protocol monitoring, received data monitoring, and transmission monitoring.

For received data, the user monitors the data for three conditions: a valid synchronization pattern, valid Manchester II modulation, and proper parity.

Transmission monitoring consists of monitoring the same parameters as for the received data when the user is sending. The synchronization pattern, Manchester II modulation, and parity are checked by the sending user on every transmission. An error causes the transmission to terminate. Other parameters which are monitored by the user are excessive message or wordstring length, undefined labels, and babbling conditions.

Protocol monitoring is performed by the user hardware also. This involves checking a number of protocol related timing parameters, such as the TG and SG. The protocol is implemented by dual hardware circuits. Each pair of protocol parameters is checked for differences. Excessive deviation will cause the transmitter to cease operation.

An error register is provided for the host CPU. Errors that are detected are indicated by particular bits being set in this register. The host CPU should monitor this register to ensure that any data bus errors receive appropriate action.

It is also possible to monitor "handshaking" between the user hardware and the memory. The ARINC 629 bus is designed to directly write and read the host memory without the intervention of the host CPU. Since this is the case, the CPU should check if these transfers were successful. The user hardware will set a bit in the error register if the correct handshake sequence does not occur.

5.1.3.2.2.5 Avionics Standard Communications Bus User Monitoring

The HDLC protocol used by the ASCB defines message delimiters and a CRC which the users monitor to determine message validity. In addition, other checks are performed by the hardware.

A Driver Enable Timer (DET) is implemented in the BCs and users to prevent babbling. If an LRU attempts to send a message longer than its preallocated time slot, the bus line driver is disabled by the DET. A checksum is added to each message. In addition, a data counter, which indicates data "freshness," is included. The host CPU must check these parameters to determine the status of each message.

The standby BC looks for invalid messages or a lack of messages from the operating controller and also monitors itself for correct operation. The active BC monitors itself for correct timing and transmissions. Upon detection of an error in one of the controllers, the controller will reconfigure to maintain a functional controller. Controllers do not, however, monitor user transmissions (Jennings 1986).

An HDLC protocol IC provides numerous parameters relating to bus operation that the host processor can monitor. It provides CRC, overrun error, transmit underrun, and other parameters in its receiver status register.

5.1.3.2.2.6 MIL-STD-1553 Bus User Monitoring

This bidirectional data bus relies on several checks of data integrity. At the physical interface, each message is checked by the bus user for a valid synchronization pattern and correct parity, and each bit of the word is checked for valid Manchester II modulation.

Other items are monitored by bus users for detecting errors on a data bus. Message formats are checked and undefined formats are rejected. For example, users reject noncontiguous messages, which have a gap between the command and data words.

Users also implement hardware timers to prevent babbling from its transmitter. If a user attempts to transmit for more than 800 microseconds, a hardware timer circuit will disable the transmission.

Upon detection of one of these errors in a data word, the user will set the error bit of the status word to a logic one. Also, normal sending of the status word is suppressed by the user. The BC will be alerted to a problem when the user response is not detected within the period of time it has to respond.

5.1.3.2.2.7 Linear Token Passing Bus User Monitoring

Monitoring is a requirement for all LTPB bus users. An LTPB BIU monitors its own transmissions and checks for various types of errors. Upon detection of an error, the host CPU is notified and action is required.

Specific transmission activities that are monitored for failure are Token claim activity, Token frame transmission, Message frame transmission, and a transmitter's detection of its own bus activity. Any detected error causes the transmitter to isolate itself from the bus and notify the host CPU of the error condition.

In each BIU there are many monitoring functions that support bus activity whose failure can impact the bus as a whole. These take the form of registers, generators, timers, a bus activity detector, and other hardware functions. The combination of all these functions is called the Self Monitor Function (SMF). A fault detected with any SMF requires corrective action and notification of the host CPU (AIR 4288, 1991).

Other activities that do not affect the bus operation but do affect the BIU and host system are not included in the SMF. These include tests such as Power-On Self-Test or Periodic Self-Test (AIR 4288, 1991).

5.1.3.2.2.8 High Speed Ring Bus User Monitoring

Monitoring is a requirement for all HSRB bus users. The master station monitors its own transmissions, checking for various types of errors. Upon detection of an error the host CPU is notified and action is required.

Monitoring is performed to detect Information field errors, message control errors, Token status errors, starting delimiter errors, Token format errors, and Token priority errors. Occurrence of these errors requires that the host be notified and, possibly, that it take corrective action.

Other errors may occur that require no explicit recovery action. In these cases it is not necessary to notify the host. These errors are reservation bits set too high, reservation bits set too low, and short message count errors.

There are also timers and counters implemented at each BIU on the ring. These timers ensure the correct operation of the protocol and guard against token loss and uncontrolled transmitters. Another timer ensures that a Beacon frame is received within a specified time (AS4074.2, 1988).

5.1.3.2.2.9 Maintenance Monitoring

Bus monitoring for maintenance purposes is a long-term data integrity issue. Monitoring is performed by a bus user that is specifically designed for this purpose. Data are gathered and stored so that analysis can be done at a later time. Tasks performed by the monitor should include the following areas:

- Check for faulty LRUs
- Check for faulty transmissions
- Check global protocol
- Record and report any error during flight

- Check general bus performance

Defective LRUs may be detected by a bus monitor that has sufficient information concerning the data bus implementation. If addresses of all users are known, then the monitor will detect a particular LRU which fails to respond when addressed. For access protocols based on TDMA, faulty transmissions may be associated with a particular LRU based on a time slot allocation table.

In addition to monitoring the operation of the LRUs attached to the data bus, there is a need to check the operation of the global protocol. Individual bus users may only verify that their own bus accesses obey the rules of the protocol. This does not guarantee that the overall protocol is functioning correctly. Data buses that implement higher level protocols, such as the bit-oriented protocols of the ARINC 429 and the ARINC 629 buses, need to be monitored for protocol violations at a level higher than a user would check. Unless the higher layers of the protocol are implemented in the user hardware, the host CPU or a dedicated communication processor must perform this monitoring function.

Monitored parameters can be used for both short-term and long-term performance evaluation. Short-term monitoring will yield information on bus quality, LRU failures, and the success of repairs. In the long-term, failure trends, mean time between failure (MTBF), mean time to repair (MTTR), performance trends, and cost of ownership can be ascertained.

Monitoring can be used to record serious errors that occur during flight and landing. While it is important that the pilot is not bothered by messages that are of little consequence, the pilot must be made aware of data bus failures that may affect flight safety. Failures of this nature should be detected by a bus monitor and reported to the cockpit so that appropriate action may be taken.

Maintenance monitoring needs to be planned for from the start of a design. One of the design goals should be ease of use. This means that the designer should keep the user in mind. The human interface needs to be simple and the messages informative. Messages can be stored in complete sentences. Today, large amounts of information can be stored in nonvolatile memory. Some of the information which might be stored in this memory and used for maintenance purposes is as follows:

- Reports of all monitorable data bus parameters
- Explanations of corrective measures to be taken for any given failure
- Diagnostic information, such as BIT status for all systems
- System diagrams
- System specifications

#### 5.1.3.2.3 Reconfiguration

Reconfiguration is a fault tolerance technique that is used in some data bus implementations. The ASCB and MIL-STD-1553 bus define it in the data bus specifications. In a centrally controlled data bus the integrity of the bus is based on the ability to not only detect a malfunctioning controller, but also remove such a controller from operation and resume operation with a standby controller. The MIL-STD-1553 bus implements this function by making use of the following ("MIL-STD-1553 Designer's Guide," 1982):

- External wiring between controllers
- Internal self tests by the controllers
- Status and health messages between controllers
- Data bus synchronization using clocks or mode codes

Another type of reconfiguration is to remove a defective user from the bus. In a centrally controlled data bus, the controller can monitor the response of any user and determine whether or not it is operating correctly. If the user does not respond within a specified time, or if it responds incorrectly, the controller can then proceed with a predefined error handling routine which may involve the removal of the user from the polling sequence.

The ASCB uses a redundant bus architecture with dual buses. Bus users transmit on only one of the buses and listen to both, while controllers can transmit on either bus. If one of the buses becomes unusable, the users have the ability to switch receivers to the other bus until valid transmissions from the BC are again received on the failed bus.

When a data bus operates under autonomous control, there is not a single source designated to monitor all users and take corrective action, as in the centrally controlled bus. It is necessary, therefore, for each user to monitor itself. Upon detection of an error, the user should execute an error handling routine which may involve the user isolating itself from the data bus. The ARINC 629 data bus is one in which a user will remove itself when an unbroken sequence of seven transmit errors is detected by the user's bus monitoring hardware.

### 5.2 Bus Hardware-Software Interaction

Constant breakthroughs in microelectronics make it difficult for a CE to address the hardware-software interaction between a digital data bus and an avionic system. Very Large Scale Integration (VLSI) ICs and multiversion software, which make up digital systems, often contribute to the CE's dilemma. With these advancements come new failure modes which need to be evaluated before a system can be considered airworthy. Section 5.2 helps the CE understand the failure modes at the hardware-software interface of a digital data bus and avionic system.

First, the hardware-software interface is identified. Next, data integrity problems that may arise when the bus and avionic system interact through

hardware and software are identified. Finally, analyses of the error detection and recovery schemes for the data integrity problems are presented.

This section reviews the interaction of avionic systems with the ARINC 429 bus, ARINC 629 bus, ASCB, and MIL-STD-1553 bus. Although the MIL-STD-1553 bus is used for military applications, problems due to hardware-software interaction resemble those of bidirectional data buses used in civilian aircraft.

### 5.2.1 Bus Interface Units and Central Processing Units

Figure 5.2-1 illustrates how avionic systems are connected to a data bus through a BIU, and shows the point of hardware-software interaction. Although the ARINC 429 bus, ARINC 629 bus, ASCB, and MIL-STD-1553 bus are different, their point of hardware-software interaction remains the same. Each data bus uses a BIU to communicate data between a bus medium and a CPU within the avionic system.

The main part of each LRU is the avionic system. It exchanges data with other avionic systems over the bus medium. An avionic system may be a flight control computer (FCC), a display computer (DC), an autopilot, or any other system which processes digital data during a flight. Avionic systems are usually constructed with complex hardware, but always contain a CPU under software control to carry out the system's repetitive functions. The CPU software is the software of interest in this section.

The CPU software allows the CPU to perform the system's application specific tasks. The CPU software may also be responsible for establishing communication between the CPU and BIU. For example, for the ARINC 429 bus, the ASCB, and the MIL-STD-1553 data bus, the CPU software receives data from, and sends data to, its BIU. For ARINC 629 bus operations, a CPU's software merely tells the CPU how to respond to signals initiated by the ARINC 629 BIU.

Two entities are needed to transfer digital data between avionic systems: the bus medium and the BIU. The bus medium connects the BIUs and carries digital data. The media are typically bundled in groups and attach systems, like those in figure 5.2-1, throughout an aircraft.

The BIU connects the avionic system to the bus medium. This unit performs all bus related tasks (e.g., bus timing, conversions, transmissions, receptions) under control of its own software, or the CPU's software. For example, data coding is accomplished by a circuit within the BIU, while transmission and reception could be controlled by software executed in the CPU. The actual functions are usually implemented in hardware and will vary, depending on the type of data bus and application.

A BIU interfaces to a CPU through the BIU's internal registers and the CPU's Random Access Memory (RAM). The registers are memory locations in the BIU that a CPU can directly access. Status registers within a BIU notify the CPU of conditions within the BIU, while control registers set up hardware operations of the data bus. Again, registers and their uses will vary depending on the design and application of the system.

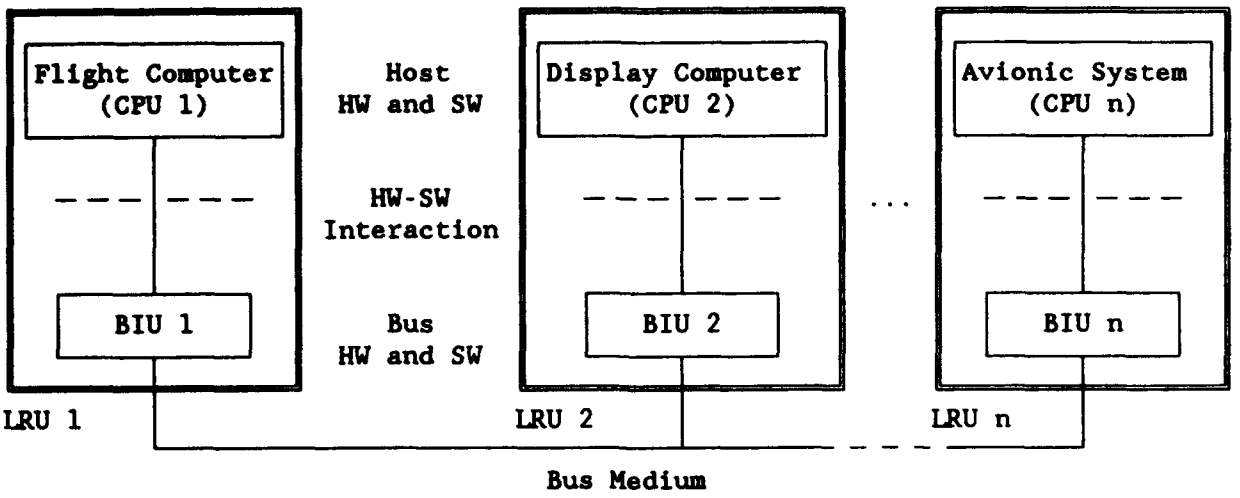



FIGURE 5.2-1. DATA BUS HARDWARE-SOFTWARE INTERFACE

The CPU's memory stores data pertaining to operations of the aircraft. For example, before altitude data can be passed from one LRU to another (i.e., from an FCC to a DC), the first LRU's CPU must send the data to the BIU. This transaction is accomplished as follows:

- An LRU receives altitude data from its sensors, and the CPU processes the data according to its software.
- The CPU then stores the processed data in memory for the BIU.
- At this point, the BIU is instructed to access the data and codes it into a format which is usable by other BIUs.
- The coded data are sent to other BIUs via the bus medium.

Once data are received by other BIUs, the procedure is reversed so that the receiving LRU can use the data for its dedicated purposes. All data transfers between a BIU and CPU are accomplished using address, data, and control lines.

The hardware-software interaction between the BIU and the avionic system's CPU should be an area of concern for the CE since failures at this interface can impact the entire system. The type of data bus, as well as the system manufacturer, determine how a BIU and CPU perform this interaction. For example, an ARINC 429 BIU may be either totally or partially controlled by the system's CPU, as previously described. In ARINC 629 bus applications, each BIU uses personality PROMs to regulate the hardware-software interaction.

The ARINC 429 bus, ARINC 629 bus, ASCB, and MIL-STD-1553 bus employ ICs to realize various proportions of the BIU. In most cases, the IC can implement all of the operating modes for a specific data bus (e.g., a MIL-STD-1553 BIU IC can be configured as a BC, an RT, or a bus monitor). However, interaction with a CPU is of the same form, regardless of mode. This section looks at one BIU

IC for each bus (table 5.2-1) and examines how improper hardware-software interaction between the BIU IC and the avionic system's CPU can inhibit data integrity.

A BIU IC does not perform all of a standard BIU's functions. It is beyond the scope of section 5.2 to discuss the hardware-software interactions of the non-integrated portions of BIU circuitry. Only interaction between the BIU IC and the CPU software are discussed.

5.2.1.1 Data Transfer Techniques

When either a BIU or CPU is requested to send data to an external location, it must use certain techniques to ensure that the data are successfully received. Since all units perform this task, the techniques must be flexible enough to adapt to many environments. For data buses, memory mapping and Direct Memory Addressing (DMA) are used to move data between a BIU and CPU.

Memory mapping may involve putting BIU registers at specific CPU memory addresses. The CPU could then access the register as a memory location rather than as an input/output (I/O) device. For example, the CPU's software could execute a memory instruction, rather than an I/O instruction, to write data to the BIU's register. MOV is a typical memory instruction, and IN and OUT are typical I/O instructions that a CPU uses to transfer data.

TABLE 5.2-1. BUS INTERFACE UNIT INTEGRATED CIRCUITS

| Data Bus | BIU IC |
|---|---|
| ARINC 429 | Harris Semiconductor's HS-3282, ARINC Bus Interface Circuit |
| ARINC 629 | National Semiconductor Corporation's XD15U9AIC, ARINC 629 IC |
| ASCB | Intel Corporation's Intel 8274, Multi-Protocol Serial Controller (MPSC) |
| MIL-STD-1553 | Digital Device Corporation's BUS-61553, MIL-STD-1553 Advanced Integrated MUX (AIM) Hybrid |

DMA is used by systems for high-speed block or packet data transfer between two memories. In a standard DMA configuration, the memory address and control lines are directly controlled by the sending device, rather than the CPU. The sending device uses a DMA controller.

The DMA controller must be initialized by a CPU's software. This is accomplished by writing data to registers in the controller. A DMA controller's registers are similar to the registers in a BIU in that they tell the controller how to operate.

The major difference between DMA and memory mapped I/O is that a CPU does not control the transfer of the data during a DMA operation.

Memory mapped I/O and DMA processes can both be accomplished through Shared Interface RAM (SIR), also called dual-port memory. With data buses, such as the ASCB and MIL-STD-1553, this is a common technique. SIR means that both the CPU and BIU share the same memory. An illustration is provided in figure 5.2-2.

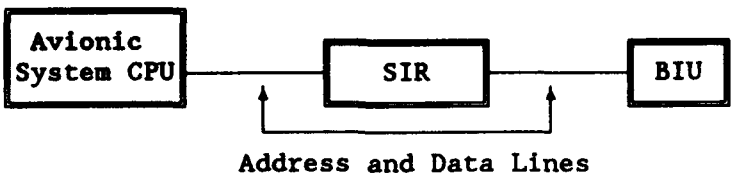



FIGURE 5.2-2. SHARED INTERFACE RAM

With this configuration, both the CPU's and BIU IC's address and data lines are directly connected to shared RAM. Access to the SIR by the two units is controlled by an arbitrator circuit. This type of shared memory provides the benefit of isolating the BIU from the CPU (i.e., no synchronization is required). Furthermore, the data transfer rate is increased since neither device has to wait for the other.

Although the ARINC 429 bus, ARINC 629 bus, ASCB, and MIL-STD-1553 bus are different, each uses registers and memory during their operations. Registers hold data pertaining to operations of the BIU IC and can be either written or read by the CPU. Memory other than registers is used to hold data during communication between a BIU and CPU. Through these registers and memory, hardware-software interaction takes place.

### 5.2.2 Hardware-Software Interaction Faults

Two types of data are passed between the BIU and CPU: bus configuration data and flight data. Bus configuration data are only sent to BIU IC registers, while flight data (e.g., altitude, heading) is shared with other avionic systems. If either type of data were to become corrupt, an error could result. Since the CPU (controlled by software) interacts with the BIU IC's registers and memory, the CPU's software has the capability to disrupt both types of data and affect hardware operations.

If the host CPU writes faulty bus configuration data to the BIU IC registers, the BIU could be set up for an improper mode, reset, or shut down. On the other hand, if the BIU puts faulty flight data in the CPU's memory, the CPU would propagate an error. Also, if external noise or an adverse environmental

condition causes data in either location to become corrupt (i.e., an inverted bit), the entire system could be affected. These situations will vary depending on the type of system used and the conditions under which the system is operating.

Typical errors that affect the hardware-software interaction of a BIU and CPU are presented in table 5.2-2. Column (a) of table 5.2-2 represents errors common to all data buses; column (b) lists errors unique to certain buses. These errors can be present in bus configuration data or flight data. It is beyond the scope of this chapter to discuss every failure mode which could cause these errors. Therefore, a generic description of the errors is provided in the following sections and, where possible, their trigger events defined.

TABLE 5.2-2. DATA BUS HARDWARE-SOFTWARE INTERACTION PROBLEMS

| Errors Common to all Data Buses (a) | Bus Specific Errors (b) |
|---|---|
| Parity Errors | Timing Errors |
| Overrun Errors | Interrupt Handling Errors |
| Synchronization Errors | |

5.2.2.1 Parity and Overrun Faults

Parity and overrun errors are common to all buses and can occur in all cases of data transfer (e.g., CPU to BIU or BIU to CPU). Parity errors may occur when digital data are either transmitted or received with an incorrect number of binary "1"s.

Depending on the system, parity errors can be triggered in many ways. For example, lightning or another environmental condition can cause data to become corrupt while it is passing through the bus medium. As a result, a unit receiving the data may detect a parity error. Section 5.1 provides a more detailed discussion of parity errors.

Overrun errors can occur at many levels of the data bus, as with parity errors. An overrun error means that current data was not used before new data was put in the same memory location or register. This error results in the loss of the old data. Overrun errors which affect the hardware-software interaction can occur in memory shared between the BIU and CPU and in the BIU during reception of data from the bus medium. This type of error can be caused by a babbling BIU that improperly transmits data on the bus or a timing flaw in a CPU's software that causes it to write data to a memory location at the wrong time.

A MIL-STD-1553 bus using a BUS-61553 IC is subject to overrun errors because it shares memory with a CPU. Although sharing memory offloads some of the CPU's

tasks and allows for DMA operations, overrun errors can easily occur because both the BUS-61553 IC and CPU are able to access the same memory. For example, when data are placed in the shared memory by a BIU (or CPU), it must notify the CPU (or BIU) that the data are available. The CPU (or BIU) then reads the appropriate location in memory and retrieves the data. An overrun problem arises when one unit updates data in a location of shared memory before the other unit reads the original contents of the memory. If this situation occurs, the updated data are written over the original data.

The HS-3282 IC is susceptible to overrun errors during reception of data from the bus medium. Data from the line receiver is placed into the HS-3282 IC's shift register. When the data are valid, a signal is generated by the HS-3282 IC telling the CPU that data are available in the register. If the data are not read by the CPU at this time, new data words being received by the HS-3282 IC will overwrite the data in the register.

Similarly, the MPSC is vulnerable to overrun errors. The MPSC stores flight data from the bus medium in receive registers. If the CPU neglects the data, the next data word coming into the receive registers will overwrite the previous data word.

5.2.2.2 Synchronization Faults

Synchronization is used between LRUs to correlate serial data transmissions and receptions. When one LRU has data to send to another LRU, its BIU may first send a synchronization pattern to the receiving LRU. This allows the receiving LRU to recognize the first bit of the message. Synchronization patterns may also be sent to announce the end of the data. The ASCB uses both of these patterns in LRU to LRU messages (Jennings 1986).

A framing error is a form of synchronization error that can occur during a write or read instruction by an LRU. A framing error means that an appropriate number of framing, or synchronization, bits around the data word were not detected by the receiving unit.

Figure 5.2-3 shows an eight-bit serial data word that could be sent by an LRU through its BIU. As defined by the system's protocol, the receiving LRU knows what type of synchronization pattern to expect. If the data word shown in figure 5.2-3 is supposed to be surrounded by synchronization patterns made up of all digital "1"s, but digital "0"s show up in these patterns, a framing error occurs. These errors could be the result of line noise entering the bus medium during data transfer between two LRUs. Regardless of the cause, if data possessing framing errors are passed on to the CPU, the system could be affected.

The MPSC employs framing when it is used in ASCB applications. Before information from the CPU is sent by the transmitting BIU, the information is framed as shown in figure 5.2-3. These framing bits allow the receiving system to temporarily synchronize with the transmitting system and eliminate timing skews between the two systems.

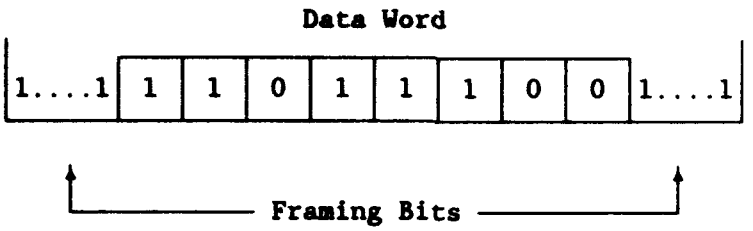



FIGURE 5.2-3. DATA FRAMING

Besides using synchronization between two BIUs, a CPU and its BIU may also need to be synchronized. This would be required if the CPU was responsible for initiating data transfer between LRUs. In this case, if the CPU does not know when the BIU is ready, it cannot properly instruct the BIU to send or receive data.

When an ARINC 429 BIU that uses the HS-3282 IC is executing the Bit-Oriented Communications Protocol (BOCP), this synchronization can occur. In the BOCP, the transmitting LRU broadcasts a Request To Send (RTS) message to the receiving system prior to transmission of flight data on the ARINC 429 bus medium. Immediately after the receiving system gets the RTS message, it must respond within a predetermined amount of time with a Clear To Send (CTS) message, a Not Clear To Send (NCTS) message, or a Destination Busy (BUSY) message (ARINC Specification 429-12, 1990). Since the HS-3282 IC does not have the logic to produce these messages, it is the CPU's responsibility to undertake the task. Improper use of these messages by a CPU could cause the HS-3282 IC to miss transmission or reception of data.

The types of errors presented above are common to all data buses. Parity, as well as overrun errors, can happen at the bus medium to BIU interface, as well as the BIU to CPU interface. Synchronization and framing errors only occur at the bus medium to BIU interface. However, they can be triggered by a condition within the BIU or CPU.

### 5.2.2.3 Timing Faults

A timing problem can arise when a CPU does not complete its data transfer to the bus before its access time to the bus expires. Timing errors of this nature are common in a time-multiplexed environment. A typical timing problem between a CPU and an ARINC 629 BIU can occur while the CPU is sending data to the bus medium.

The problem arises from the fact that BCAC's Integrated Avionic Computer System (IACS) integrates many avionic systems on a number of central CPUs and uses autonomous ARINC 629 BIUs. The CPUs perform several functions which share the CPU memory. The functions are partitioned to prevent hardware and software failures in one function from affecting another partition's functions. All of the CPUs within the IACS are controlled by a software algorithm known as the Real-Time Executive (RTE).

The RTE controls transmit and receive timing between the LRUs. To ensure that all transmissions and receptions are coordinated, the RTE gives each LRU a specific amount of time in which to complete its message transmission. To accomplish a transmission, an ARINC 629 BIU must obtain processed data from the CPU and completely transmit the data on the bus medium in a predetermined amount of time. Since the RTE controls when the ARINC 629 BIU obtains data, the RTE could instruct an ARINC 629 BIU to interrupt a CPU before all of the CPU's data are ready to be transmitted. The ARINC 629 BIU accesses the data without first checking if the CPU is done with its process. When this occurs, the ARINC 629 BIU could transmit partially updated or otherwise erroneous data to other LRUs.

A similar timing problem arises with the MIL-STD-1553 data bus. If periodic data are to be processed by a CPU's software, the MIL-STD-1553 BIU must notify the CPU that data are available and ready to be processed. In a MIL-STD-1553 bus application, a specific signal is used to annunciate this condition.

Once the signal is generated, the CPU has a certain amount of time to acknowledge the signal and process the data. (Recall that the time is designated by a BC, and the BUS-61553 IC is capable of performing BC operations.) If the CPU takes more time to process data than the BC allows, the BC must either terminate the CPU's access to the bus, or wait for the CPU to complete its task. If the BC elects to terminate the CPU's access to the bus, an error similar to the ARINC 629 bus timing problem could result. The transmitting BIU could get erroneous or old data and send it to other LRUs. On the other hand, if a CPU is constantly allowed to overshoot its allotted time, the entire network will "jitter in its periodicity" ("MIL-STD-1553 Designer's Guide," 1982). These two descriptions show how both a distributed control and a centrally controlled bus can be exposed to similar timing errors.

#### 5.2.2.4 Interrupt Handling Faults

Interrupts are a standard method of initiating data transfer between a BIU and CPU. For example, when a BIU places data in SIR the BIU must send a signal on an interrupt line to the CPU to announce that the data are available. This signal is called an interrupt.

If a BIU generates an interrupt to the CPU, the CPU may respond with an acknowledge signal and, either suspend what it is doing in the main part of the program and read the data, or continue to process until some later time.

Interrupt handling problems can arise when more than one interrupt is generated at one time. For example, if a CPU is already servicing one interrupt and its BIU initiates another interrupt, which one should get priority and how will throughput of the bus be affected? Avionic system manufacturers must deal with these conditions.

The MPSC uses interrupts to notify the CPU when one of 14 conditions occur. If all of these conditions happen within a short period of time, they could cause the CPU to be so tied up with interrupts that it cannot maintain the required application processing. Furthermore, the CPU may not be able to promptly service all of the interrupts. This would also affect the operation of the LRU.

### 5.2.3 Fault Detection

If not detected, all of the errors discussed in section 5.2.2 have the potential to cause a bus failure. To recognize these types of errors, BIUs and CPUs can employ bit-level detection schemes during data transmission and reception. Using these schemes, the BIU or CPU can spot faulty data before it leaves or enters the unit's boundaries. For both BIUs and CPUs, bit-level detection schemes include parity checks, CRCs, checksums, and Hamming codes. Each check is valid for detecting certain types of bit errors. The checks, and which buses use which checks, are detailed in section 5.1.

When a BIU is responsible for error detection, it should be able to annunciate results of the data checks so that corrective action can be taken when necessary. In most of today's avionic BIUs, this notification is performed by setting or resetting a specific bit in the BIU's status register. Once a bit has been appropriately set, either the BIU can interrupt the CPU and report the error, or the CPU's software can periodically access the BIU's register and read the error.

If the BIU is incapable of performing any checks, the CPU may be responsible for error detection. In this case, the CPU checks bus configuration or flight data which enters or leaves its bounds and flight data entering the BIU from the bus medium. Error detection routines within the CPU include ones previously mentioned and may be implemented as a routine in the CPU's software. A CPU's detection responsibilities will vary depending on each application and must be defined by the system's designer.

Monitoring and voting are other methods that can be used to ensure that failures at the BIU to CPU interface do not go undetected. These are also discussed.

#### 5.2.3.1 Bus Interface Units and Fault Detection

The MPSC contains 21 registers which a CPU can access. These registers are split between two redundant channels: A and B. Of the 21 registers, 10 belong to channel A, and 11 belong to channel B. Channel A registers include Write Registers (WRs) zero though seven (WR0-WR7) and Read Registers (RRs) zero and one (RR0 and RR1). Channel B registers are the same, except that channel B includes an extra register used to service interrupts: RR2. This register either contains the interrupt vector programmed into WR2 or holds the vector of the highest pending interrupt within the MPSC.

When the MPSC receives a data word from the bus, the MPSC checks the data for integrity. If a parity, framing, or CRC error is detected by the receiving circuitry, the MPSC sets a specific bit in the appropriate read register ("Microcommunications," 1990). The system's CPU can check for errors by polling the MPSC, or by an MPSC interrupt.

The BUS-61553 IC uses similar methods to inform its CPU of parity, overrun, and synchronization errors. Within the BUS-61553 IC are three internal registers that the CPU can access: the Configuration Register, the Interrupt Mask Register (IMR), and the Start/Reset Register ("MIL-STD-1553 Designer's Guide," 1982). Each has different applications.

The IMR can be read or written by the CPU. Upon reception of a data word from the bus medium, the BUS-61553 IC checks the data to ensure that it does not violate the MIL-STD-1553 bus formats. If a parity, overrun, or synchronization error is detected, the "message error bit" within the IMR will be set ("MIL-STD-1553 Designer's Guide," 1982). The CPU can then read the IMR and take appropriate action. Other errors that the BUS-61553 IC can detect include loop test failures, coding errors, and time-out errors.

A Hamming code is another detection scheme that the BUS-61553 IC can use. The IC uses this code to detect and correct up to three erroneous bits in a flight data word. Detection is accomplished by sending a protection word immediately after each 16-bit data word. If blocks of data are to be checked, the protection word would follow each consecutive 16-bit data word in the block. Section 80 of the "Multiplex Applications Handbook" (MIL-HDBK-1553A, 1988) discusses this error detection scheme. In addition, a general description of Hamming codes is provided in section 5.1 of this chapter.

The ARINC 629 IC transfers bit-level errors, as well as diagnostic information, to a CPU via an error register. The error register is 16 bits wide. Each bit represents a different error condition. Twelve of the 16 bits are latched ("ARINC 629 Communication Integrated Circuit," 1990). When an error occurs, a corresponding bit is set in the error register. The other four bits in the error register reflect the ARINC 629 IC's current status.

The HS-3282 IC does not hold error information for its CPU. Instead, the HS-3282 IC passes error detection responsibility directly to its CPU or another external device. The HS-3282 IC does, however, use a configuration register to distribute internal control signals. One of these signals directs the HS-3282 IC to check its transmission for proper parity.

From the above discussion, it is apparent that BIUs are capable of annunciating many types of errors to a CPU through internal registers. The errors include ones previously discussed, like parity and overrun, but may also include others like coding and loop-test failures. Each BIU and data bus manufacturer must develop their own method to inform the avionic system of hardware-software interaction errors while keeping within the bounds of the data bus's standard. Even though this is a job for the BIU and data bus manufacturer, the CPU programmer and system integrator must design the system to utilize all information provided by the BIUs.

#### 5.2.3.2 Monitoring

Monitoring can be performed at many levels in a data bus. However, this monitoring discussion details only processes that apply to errors at the hardware-software interface.

Besides a BIU annunciating faults to the CPU using interrupts, most BIUs can be monitored by a CPU's software. As with the other forms of error detection, this allows the CPU to take appropriate corrective action in the event of an error. Software monitoring by a CPU may mean periodically polling a register or memory location in an LRU, or may require a dedicated algorithm in the CPU's software

to oversee the operation of the entire BIU. As with detection methods in BIUs, monitoring techniques will vary from one application to another.

The MPSC provides a good example of how software monitoring can be employed. When the MPSC is configured for the polled mode of operation, the CPU can monitor conditions by reading bits in the MPSC's RR0 and RR1. Data available, status, and error information are apparent in RR0 and RR1 for both channels of the MPSC.

An example of an algorithm that a CPU can use to monitor the MPSC is discussed in "Microcommunications" (1990) and is called MPSC$POLL$RCV$CHARACTER. This algorithm tells the MPSC to get data from the bus medium and wait until the "character available" flag in RR0 is set. After this flag is set, the CPU checks RR1 for parity, synchronization, and overrun errors. If errors are detected, the receive buffer must be read and another algorithm, RECEIVE$ERROR, must be called. This algorithm processes errors received by the previous algorithm. However, the RECEIVE$ERROR procedure is application dependant. The RECEIVE$ERROR algorithm requires the address of the affected MPSC channel and the contents of RR1 to operate. Both algorithms are shown in Application Note Number 134 ("Microcommunications," 1990).

If a CPU is incapable of monitoring the BIU at this level, or the software overhead required for the task is not permitted, monitoring can also be done by a dedicated LRU. For example, the MIL-STD-1553 bus employs bus monitors and the ASCB implements a similar, special purpose monitor called a Listen Only User. These monitors are separate LRUs. They are attached to the bus medium as shown in figure 5.2-1.

The MIL-STD-1553 bus monitor listens to all data on the bus and "extracts selected information to be used at a later time" ("MIL-STD-1553 Designer's Guide," 1982). A typical bus monitor performs no transfers on the bus, but bus monitors usually have the capability to become a MIL-STD-1553 bus RT under request from the BC. Applications of the bus monitor include data collection and monitoring the overall system for status information.

In some cases, a bus monitor can be configured as a back-up BC. When this occurs, the bus monitor collects data, watches transmissions, and performs the same jobs as the current BC, with the exception of issuing commands on the bus. This way, the bus monitor is continuously aware of the operation of the overall system and subsystems, and is available to serve as a back-up BC if an error between the hardware and software takes down the original BC ("MIL-STD-1553 Designer's Guide," 1982).

The ASCB BC is also capable of being used as a self-monitor, as stated in the ASCB Specification:

> "In the active control mode, the bus controller shall self-monitor its own bus control operation. If bus control performance, as described in this specification, is not being performed properly, the bus controller shall remove itself from bus control operations and assume the standby mode. Monitoring techniques shall provide coverage for both hardware faults and software errors. In addition, the monitor

shall verify proper content and timing of all control sequences being transmitted." (GAMA ASCB, 1987).

The monitors used in both the ASCB and MIL-STD-1553 buses must watch for single points of failure at the BC and associated BIUs. This environment helps ensure that hardware-software interaction errors will not cause the simultaneous failure of the BC and other BIUs on the bus.

5.2.3.3 Voting

Voting is another fault detection method that can be used by LRUs. Although this technique is usually applied at the system level, it can be utilized at the BIU to CPU interface. Voting is typically done at either the input or output of a system.

Voting requires at least three redundant units. Although in most applications a single CPU interacts with a single BIU, either of these units can be made redundant to incorporate voting. For example, an LRU may contain three CPUs which process data.

Input voting can be done on data from the bus medium before it reaches the CPU, while output voting can be done on data between redundant CPUs and the BIU. (The definition of input and output voting will vary depending on the reference point in the system.) In both cases, a circuit within an LRU compares the values from triply redundant CPUs or BIUs and passes on a refined value. Thus, erratic data from any of the redundant units will be detected. Figures 5.2-4 and 5.2-5 illustrate the concepts of input and output voting.

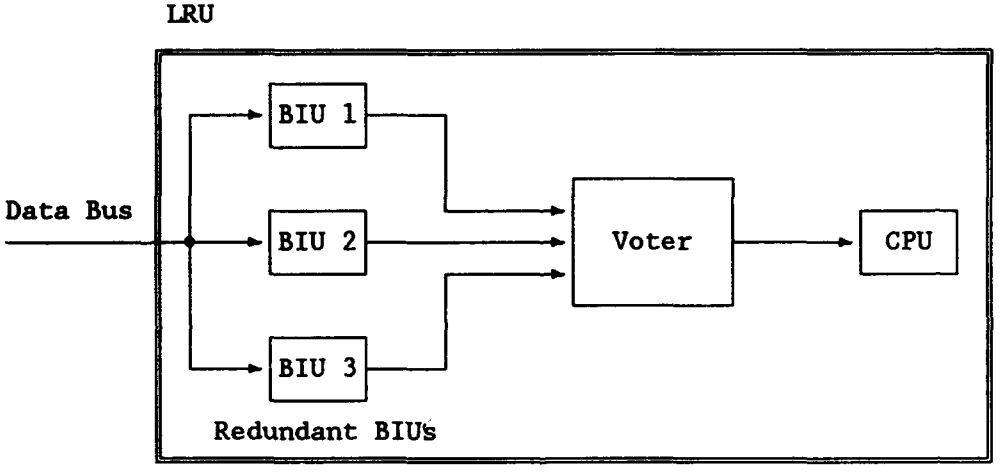



FIGURE 5.2-4. INPUT VOTING

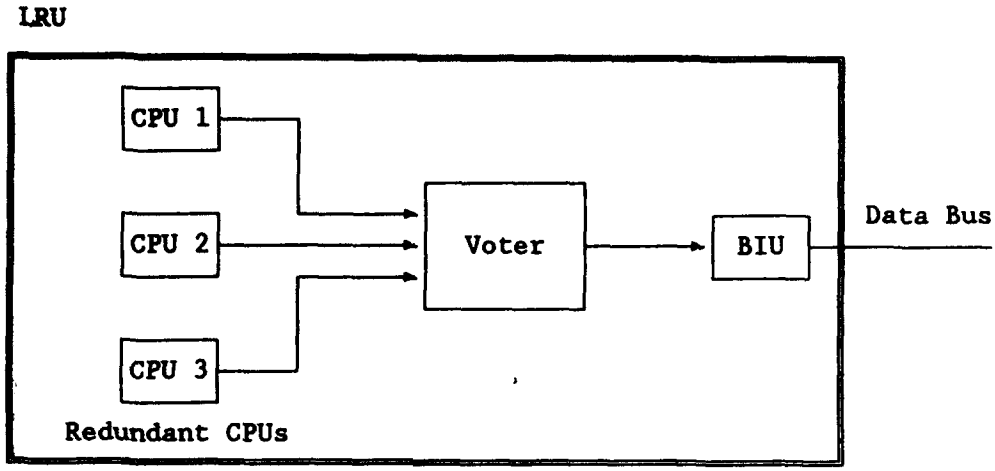



FIGURE 5.2-5. OUTPUT VOTING

The extent of the voting architecture depends on which component failures are to be compensated. Input and output voting can be used to create systems that have a high level of fault tolerance.

### 5.2.4 Fault Correction

Previous sections presented typical errors which occur during data transmission and reception between a BIU and CPU. Also discussed were different methods used to detect the errors. The errors addressed, however, are not the only ones that can occur, nor are the detection schemes the only ones that can be used. Nevertheless, it is the system designer's job to ensure that no hardware or software errors between an avionic system and its data bus cause a flight-critical or flight-essential system to fail.

The correction methods described in the following sections apply to the faults presented in section 5.2.2. Retransmission is a standard method of correction for errors that have already occurred. Multiple buffering is a method that prevents certain errors from occurring. Besides these methods, fault tolerant bus architectures that rely on redundancy for error correction are presented. Although these architectures are not usually incorporated by the buses discussed in this chapter, they are valid solutions to many hardware-software interaction problems.

#### 5.2.4.1 Retransmission and Default Data

Once a parity error has been detected by a BIU or CPU, retransmission and use of default data are correction methods that can be used. Retransmission simply means sending the same message again, and default data are values automatically used unless other values are specified. For correction purposes, default data could be used in place of data which has been verified to be unusable. These simple schemes can effectively correct parity errors that result from transient interferences.

Most data bus systems are capable of using software to request retransmission if a parity error occurs. Consider an ASCB using the MPSC. The CPU can monitor the MPSC's status by testing appropriate bits in the MPSC's RRs. If a parity error is detected within the MPSC, the data are discarded, and the CPU runs the MPSC$POLL$RCV$CHARACTER algorithm. Depending on the application, the algorithm could be set up to request retransmission from the sending unit.

The ARINC 429 data bus, under the BOCP, is also capable of using retransmission in the event of a parity error. Prior to flight data transmission on the ARINC 429's bus medium, the transmitting system sends an RTS message to a receiving system. If the RTS message is accepted and the transmitting system is allowed to transmit its data, it sends a Start of Transmission message, followed by the data, to the receiving system. Immediately after the receiving system gets the data, its CPU can test it for parity errors. If a parity error is detected by the receiving system, it sends a Not Acknowledge message back to the transmitting system. When this message is received, the transmitting system could be configured to retransmit its data.

Retransmission is useful for correcting synchronization errors. As pointed out in section 5.2.2, framing bits can be used to synchronize data during receptions. If the framing bits become inverted due to an error, the BIU may not be able to recognize when a reception is completed. In this case, the BIU or CPU could request a retransmission from its source.

A system that uses the ARINC 429 data bus under the BOCP and uses an HS-3282 IC, employs retransmission in the event of a synchronization error. If a transmitting system's RTS message is ignored, or if the receiving system sends a message which prevents the transmitting system from broadcasting its data (NCTS or BUSY), the sending unit retransmits its RTS message within a time defined by the ARINC 429 bus specification. If the second RTS message is ignored, the transmitting unit should keep trying until five RTS messages have gone unacknowledged. If, however, the sending unit receives a BUSY message, it may repeat its RTS message up to 20 times. ARINC Specification 429-12 (1990) states:

> "The actual number of attempts a source should make before giving up, or taking some different course of action, when the limit is exceeded depends on the application."

Using default data is another way to recover from parity, overrun, and synchronization errors. For example, if a BIU receives data with bad parity from the bus medium, the CPU may elect to use default data for the next process. Even though this method keeps errors from tying up a system, the designer must ensure that using default data will not upset the operations of a flight-critical or flight-essential system.

#### 5.2.4.2 Interlocks

Interlocks are a method of preventing timing errors during data transmission on serial data buses. Interlocks, which are usually constructed with hardware, prevent BIUs from transmitting at inappropriate times.

An ASCB BIU is capable of using an interlock to prevent timing problems during transmission, as stated in the ASCB specification:

> "Each user which transmits on the bus, has an interlock to prevent erroneous transmissions longer than its allocated time on the bus. A separate, dedicated hardware timing circuit, is used to enable the transmitter, in each of the users, only when the specific request is received." (GAMA ASCB, 1987).

This interlock is provided by the DET which ensures that an ASCB user will not transmit out of its time frame.

5.2.4.3 Multiple Buffering

Although overrun errors are as common as parity and synchronization errors, they are more complicated since a receiving system may not be aware that an overrun error has occurred. One method for preventing overrun errors is a memory management scheme called multiple buffering. Besides keeping data from being overwritten, multiple buffering prevents partially updated data from being read by the CPU or sent to the BIU. Both the ARINC 629 bus and the MIL-STD-1553 bus use multiple buffering.

To employ the multiple buffering scheme, a BIU and CPU must share memory. The memory is segregated into several areas which are swapped by the CPU or BIU at appropriate times. The key to this scheme is that the CPU and BIU are only allowed to access one area of shared memory at a time.

MIL-STD-1553 applications using the BUS-61553 IC employ multiple buffering to prevent overrun errors by assigning two or more areas of memory for each address shared by the CPU and BIU. Each area is 32 bits wide. Control information, contained in another part of memory, specifies which area is to be used by the CPU and which area is to be used by the BUS-61553 IC.

When the BUS-61553 IC is to receive information, it writes data in one area, while the CPU reads previous data from the other area. Upon completion and validation of the received message, circuitry within the BUS-61553 IC toggles the two areas, making the newly received data available to the CPU. During transmit operations from the CPU to the BIU, the scheme is reversed. The CPU writes data to one area, while the BIU reads data from another area. When the CPU completes its write, the CPU swaps the two a eas of memory and allows the BIU to access the new data. All memory swaps occur totally between the reads or writes ("MIL-STD-1553 Designer's Guide," 1982).

Multiple buffering is a valid solution to the ARINC 629 bus timing problem. A partition within BCAC's IACS could be set up to write to a different buffer than the ARINC 629 IC reads. As described above, the read and write buffers could be swapped, preventing the ARINC 629 IC from reading a buffer that is currently being written.

### 5.2.4.4 Grace Periods

A correction method for the timing error presented in section 5.2.2 can be implemented in either hardware or software. A hardware solution utilizes a multiple buffering technique as described in section 5.2.4.3, while a grace period is a software correction method used for both the MIL-STD-1553 and ARINC 629 bus timing problems. A grace period can be implemented within the IACS's RTE, or the MIL-STD-1553 BC's software.

Recall that an IACS's RTE is capable of instructing each LRU when to obtain data. Therefore, if the RTE knew when an LRU's CPU was done with processing, the problem would be resolved.

To correct the timing problem, each of an ARINC 629 LRU's tasks are completed under a software subroutine within the RTE. In this subroutine, the RTE monitors whether an LRU has completed its process. If one LRU's process is not completed when the RTE wants to switch to another LRU, the RTE allows an LRU extra time (a grace period) in which it can finish its job. A MIL-STD-1553 application uses similar correction methods for the timing problem; the BC provides a grace period (equal to one minor frame) to the LRU.

Another method that the ARINC 629 bus could use to correct the timing problem requires the functions within the LRUs to transmit and receive data at the beginning of their time frame. Furthermore, each LRU's time frame must be longer than any of the transmissions or receptions could possibly take. Although this solution eliminates the timing problem, processing completed while an LRU is in a current time frame would not be made available until the next time frame (Bakken 1988). The advantage to this solution is that it requires less CPU overhead than the grace period solution.

The use of grace periods merely increases the time to complete a task. If transmissions exceed the grace period, an error would be announced and corrective steps would need to be taken as if the grace period was never implemented. It is the system designer's responsibility to decide what solution would be best for a situation.

### 5.2.4.5 Prioritizing

A BIU or CPU can employ prioritizing to eliminate incorrect handling of interrupts. The purpose of prioritizing is to decide which interrupt is more critical.

The MPSC uses priority in both a vectored and nonvectored mode to decide which interrupt deserves attention. In the vectored mode, the MPSC sends the location of the interrupt's service routine to the CPU along with the interrupt condition. In the nonvectored mode, the CPU is responsible for determining the location of the interrupt's service routine. In either mode of operation, the 14 interrupt conditions are categorized by the MPSC into three different interrupt requests for each channel. This means that there are six interrupt requests generated by the MPSC ("Microcommunications," 1990).

Correct handling of these six requests can be accomplished by a priority resolution circuit. In the vectored mode of operation, a circuit within the MPSC decides which interrupt deserves priority. In the nonvectored mode, a circuit contained in an external device, such as Intel's 8259A Programmable Interrupt Controller, may prioritize the interrupts.

A system that uses the BUS-61553 IC requires the CPU to determine which interrupt should have priority. The BUS-61553 IC contains an IMR which holds information about interrupt conditions for the CPU. The interrupt conditions may be the ones explained in the BUS-61553 IC's data sheet or others defined for a specific system. If any of these interrupt conditions occur, the BUS-61553 IC sends an interrupt request signal to the CPU. The CPU responds with an acknowledge message and reads the IMR to determine which interrupts have occurred. The CPU then selects the highest priority interrupt and runs the appropriate service routine.

The 8088 CPU uses an Interrupt Vector Table (IVT) when establishing the priority of interrupts. Interrupt vectors, which point to the beginning of the service routines for a BIU, are put in this table. The 8088 CPU uses the position of the interrupt vector in the IVT to decide which interrupt deserves priority. Other processors, like Zilog's Z80, can be set up in the same manner to service interrupts and eliminate interrupt handling errors.

5.2.4.6 Redundancy

Because so many errors are application dependant, having a back-up system is a good method of correction. Redundancy is the most widely use method for prevention and correction of all data bus errors resulting from hardware-software interaction. Most avionic systems implementing flight-essential and flight-critical applications use at least one form of redundancy to meet requirements for certification.

Redundancy employs either similar or dissimilar hardware and software to mimic operations of a primary system. These redundancy techniques can be applied at all levels of the system including CPUs, BIUs, and the bus medium. All of the BIU ICs employ a form of redundancy within their bounds. The HS-3282, ARINC 629, BUS-61553, and MPSC ICs all are capable of transmitting or receiving data on one of two channels. However, all of these BIU ICs have only one interface to the CPU.

When choosing a redundant technique at the hardware or software level, the designer must decide whether to employ similar or dissimilar redundancy. Similar redundancy makes the whole system easy to design and verify, but does not guard against generic errors. Dissimilar redundancy does protect the system from these errors, but takes more time to design, is more expensive, and is harder to evaluate during certification.

Redundant techniques that use both hardware and software include Honeywell's Self-Checking Pair (SCP) (Driscoll 1983), triplication and voting (Spitzer[2] 1986), and the Fault Tolerant Multi-Processor (FTMP) Architecture (Lala 1983). Although these techniques are not designed by the data bus manufacturers, they provide valuable techniques that can be used by data bus manufacturers when

designing the bus hardware-software interface. See chapter 5 of the "Handbook - Volume I" (Hitt 1983) for further discussion on this topic.

5.2.4.6.1 The Self-Checking Pair

Figure 5.2-6 shows a diagram of two SCPs. The SCP includes identical halves made up of application processors (APs) and BIUs. The transmitting and receiving LRUs are each an SCP. Notice that the only differences between this diagram and figure 5.2-1 are that external monitors watch each input and output, and each CPU and BIU has a back-up.

The monitors on the transmit (output) side and the receive (input) side of the SCP are the key to the system. Assuming that both transmit CPUs process the same data, the BIUs' outputs to the monitors (and bus medium) should be identical. If, for some reason, data to both output monitors is not consistent, the monitors are able to switch the faulty system offline. Input monitors function in the same way. If a faulty output monitor or bus error causes bad data to be passed to the receiving system, the input monitors should catch the error and prevent it from being passed to the receiving system.

The SCP is applicable to both unidirectional and bidirectional data bus networks. An SCP could be placed in one LRU of a bidirectional network, or transmit and receive SCPs could be placed at the ends of a unidirectional bus. To enhance the performance of these networks, the SCP CPUs could be programmed using dissimilar software.

5.2.4.6.2 Triplication and Voting

The previous section described how a dual redundant SCP was able to address the issue of fault correction in a digital system. It also mentioned that the CPUs in the SCP could be programmed with dissimilar software to enhance the operation of the SCP. In 1984, the Sperry Corporation developed a fault tolerant system which employed multiversion programming, voting, and monitoring for error detection and reconfiguration for error correction.

This particular architecture uses three redundant CPUs in two identical FCCs. Two of the CPUs within each FCC share memory and are programmed with identical software, while the other CPU is programmed with dissimilar software and has its own memory. The output of the paired CPUs, as well as the single CPU, go to separate data buses.

Each FCC uses one of the paired CPUs to perform both flight-critical and flight-essential functions, while the other two CPUs perform flight-critical functions only. The outputs of the paired and single CPUs are compared by two monitors. If a monitor detects a failure at any CPU's output, the system is gracefully reconfigured so that one FCC is always engaged. A diagram and discussion of how the system reconfigures itself in the event of an error is presented in Digital Avionics Systems (Spitzer 1987).

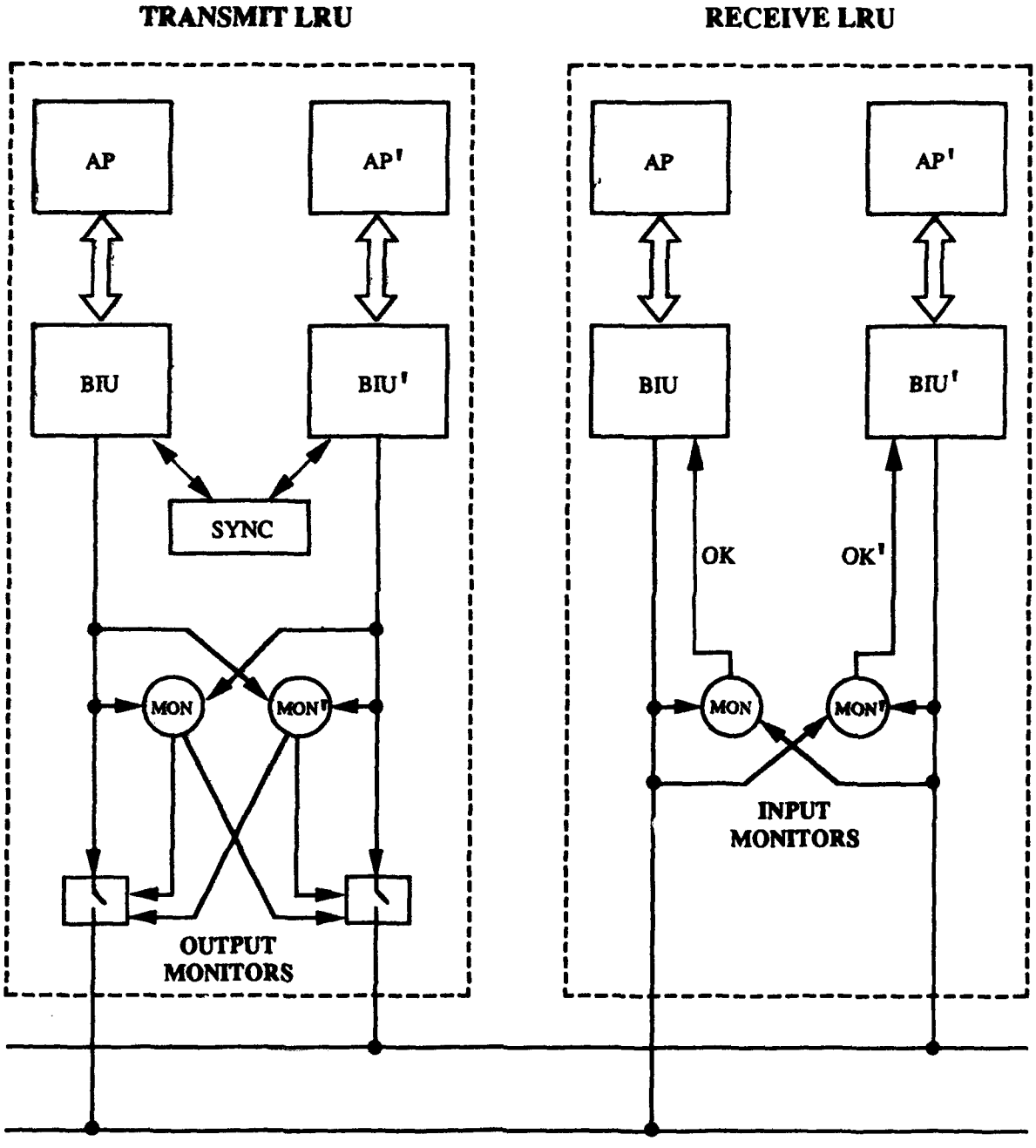



FIGURE 5.2-6. SELF-CHECKING PAIRS
(Driscoll 1983)

#### 5.2.4.6.3 Fault Tolerant Multi-Processors

In a fault tolerant environment, multiple CPUs are used to process similar data and monitor transactions taking place on the bus. These CPUs typically share a central memory and communicate over one or more redundant data buses. This configuration allows a back-up CPU to immediately take over the process of a failed CPU. One method of employing multiple CPUs in flight control applications is called FTMP.

The FTMP architecture uses both hardware and software to detect and correct errors in an avionic system. Hardware within an LRU is used to accomplish fault detection and error masking. Ten LRUs, each made up of CPUs, memory modules, and BIUs, are organized in triads to form three groups of three LRUs and a spare LRU. Any three CPUs, BIUs, or memory modules may be organized as a triad. Communication between LRUs is accomplished over four, triply redundant, bidirectional buses called the transmit, receive, polling, and clock buses. Each triply redundant bus is backed up by two spares, making the total number of bus connections 20. During a data transfer operation, the CPUs send data to the shared memory modules from which the BIUs can obtain the data and send it across the buses to other LRUs.

When a fault is detected at an LRU, a System Configuration Controller (SCC) ensures that all CPUs are aware of the fault and have the same information about the fault. The SCC is merely an algorithm run within an LRU triad that reads error information from all 10 LRUs.

Some faults can be immediately isolated and detected by the SCC. For faults not so easily identified, the FTMP executes a reconfiguration routine to isolate the source of the fault. This routine swaps LRU triads (depending on the nature of the fault) between the redundant data buses until the faulty LRU is identified (Lala 1983).

After a fault has been isolated, the FTMP implements techniques to recover from the condition. These techniques include using the spares of each unit. Recall that there are three triads and one spare of each CPU, memory module, and BIU. To reconfigure from a failure of a CPU, first the spare CPU would be brought online. If the spare CPU was already online and another fault occurred, the FTMP would remove the entire LRU triad, operate from the other triads, and use the remaining two CPUs in the failed triad as spares for the remaining LRUs. A similar recovery method is used for memory module or BIU failures.

Even though the FTMP was designed for use with MIL-STD-1553 data buses, it is acceptable for commercial aircraft. FTMP is capable of masking single faults in a system by reconfiguring each faulty node with redundant spares.

#### 5.2.4.7 N-Version Programming and Recovery Blocks

N-version programming and recovery blocks are software based methods usually employed in redundant systems containing three or more CPUs. These are valid means of dealing with certain hardware-software interaction problems, and are presented in chapter 9 (Hecht 1989) of this handbook.

### 5.2.5 Summary

All of the discussions in section 5.2 are meant to help the CE better understand the hardware-software interaction between a data bus and its avionic system. This interface is important because many situations that affect the integrity of a bus or an avionic system may arise at this point and can easily be overlooked during the certification process.

Because new technology constantly changes the way avionic systems communicate, it is hard for a CE to evaluate hardware-software interaction during a system's certification process. To help the CE with this problem, appendix C provides a hardware and software analysis checklist for failures in bus related hardware and software. The checklist is not specific to any particular failure mode. It is a general approach to evaluating bus related hardware and software failures which could impact the operation of flight-critical or flight-essential systems.

## 5.3 Bus Protocol Specification and Verification Methods

Development work in the area of data buses is progressing rapidly due to the requirement for higher throughput and reliability. Along with this development comes the need for new comprehensive methods of evaluation and testing. New data buses must be analyzed to ensure that they will function properly under all foreseeable conditions.

One area that requires careful attention from the designer is the communication protocol. In a system of distributed computers that are required to communicate with each other, rules must be developed and implemented to avoid chaos when messages are exchanged. The complete set of rules is referred to as the protocol. The protocol should ensure safe and timely delivery of data or control messages from one user of a data bus to another. The fact that the protocol may be implemented in a single high-density IC is all the more reason to subject the protocol to rigorous analysis.

Specification techniques are used to model and define protocols while verification techniques demonstrate that the protocol satisfies the specification. Protocols having different characteristics require different specification and verification techniques. No single method is suited to every existing protocol (Merlin 1979). The following sections describe some of the formal methods used to specify and verify communication protocols. Techniques such as state machine analysis and Petri nets are examined, along with examples and applications to current data buses.

### 5.3.1 A Protocol Specification Guideline

Recently, the ISO adopted ISO 7498 (1983), "Information Processing Systems - Open Systems Interconnection - Basic Reference Model." This standard was designed to facilitate the interconnection of systems from different network manufacturers. It is the IEEE standard model for the "Open Systems Interconnection" (OSI) architecture.

Organizations responsible for developing protocol standards increasingly make use of the Basic Reference Model. The ARINC 429 DITS has been modified to make use of the model, and the ARINC 629 bus totally reflects its philosophy. The Basic Reference Model aids the designer in developing a protocol without imposing unnecessary constraints upon its design. When a protocol is functionally layered, as the model requires, it is more easily understood by those who wish to study it. The use of the model also clarifies the purposes and capabilities of the protocol.

### 5.3.2 Protocol Specification Content

The use of formal techniques for specifying and validating protocols has increased due to the rise in protocol complexity and the need for reliable data transmission in distributed systems. A list of general guidelines used in specifying protocols is given in table 5.3-1.

TABLE 5.3-1. PROTOCOL SPECIFICATION GUIDELINES
(Bochmann and Sunshine 1980)

1. A general description of the purpose of the layer and the services that the layer provides.

2. A precise specification of the service to be provided by the layer.

3. A precise specification of the service provided by the layer below and required for the correct and efficient operation of the protocol.

4. The internal structure of the layer in terms of entities and their corresponding relations.

5. A description of the protocol(s) used between the entities, including:

   a. An informal description of the operation of the entities.

   b. A protocol specification which includes:

      (1) A list of the types and formats of messages exchanged between the entities.

      (2) A list of rules governing the reaction of each entity to user commands, messages from other entities, and internal events.

   c. Any additional details, not included above, such as considerations for improving the efficiency, suggestions for implementation choices, or a detailed description which may come close to an implementation.

Although item 1 in table 5.3-1 is important in understanding the protocol, it is not required. If any of the other elements of the specification are lacking, the specification is deemed incomplete.

### 5.3.3 Protocol Specification Methods

Protocol specifications must be both concise and easy to understand. In complex protocols these two goals are in conflict. A natural language description may appear to be easily understood, but leads to lengthy and informal specifications which often contain ambiguities and are difficult to check for completeness and correctness (Bochmann and Sunshine 1980).

Formal techniques and their variations are used for protocol specification. The major methods are the use of Petri nets, state diagrams, high-level computer languages, and various grammars designed for this particular application. State diagrams, Petri nets, and grammars are used to model the responses to data transfers at a layer interface or an internal timer. This type of modeling is event or transition driven. A particular drawback of this method is that protocols using sequence numbers become quite cumbersome to model. If an eight-bit sequence field is used, then a separate state would exist for every possible combination of the eight bits.

High-level programming languages are used to model protocols and have the advantage of being easily understood since they appear more like natural language. The problem of representing sequence numbers in the state diagram is easily handled by the use of a variable to represent all combinations of that number. This method differs little from an actual implementation of the protocol. However, certain unique characteristics of the programming language which may be nonessential to the protocol model could hinder the implementation.

#### 5.3.3.1 The Finite State Machine

The Finite State Machine (FSM) concept has been a key element in protocol specification. It can be used to model the global state of the protocol over an entire network, or one state machine may be used for each entity in a layer. At a given time, the state machine may be in only one of the defined states.

For complex protocols it is tedious and time consuming to generate a state diagram of all the possible states. When this is the case, one approach to simplify the protocol representation is to group together a large number of states. Since some states consume a relatively small amount of time in relation to other states, these states may be regarded as transient and grouped together as one state for purposes of analysis. Since states are defined to be cases where the FSM is waiting for the next event to occur, the number of states may be represented by $2^n$, where n is the number of bits needed to represent the variables which cause the transitions.

In a given state there are zero or more transitions to other states which happen when a designated event occurs. Typical events which cause transitions in the FSM are when an internal timer triggers, when a message is received, when a message is transmitted, or when an interrupt occurs. If the bus medium, or link, is modeled separately from the sending and receiving protocol, then the

transitions that may cause the link FSM to change states are a message entering the link, a message leaving the link, or loss of data in the link.

In figure 5.3-1, a sender-receiver topology is modeled in a simple fashion with an FSM model. There are four distinct global states and four distinct transitions between the states given in this FSM. The action of an entity sending data forces a state transition to the "Wait for Data" state. Upon receiving new data the "Process Data" state is entered. When "Process Data" is finished, the "ACK" status is sent and the "Wait for Acknowledge" state is entered. Finally, when the "ACK" is received the network returns to the "Idle" state, clearing the way for new data to be sent. The advantage of this model is that the global characteristics of the network can be directly checked. If the protocol is complex the FSM model will be complex and difficult to construct.

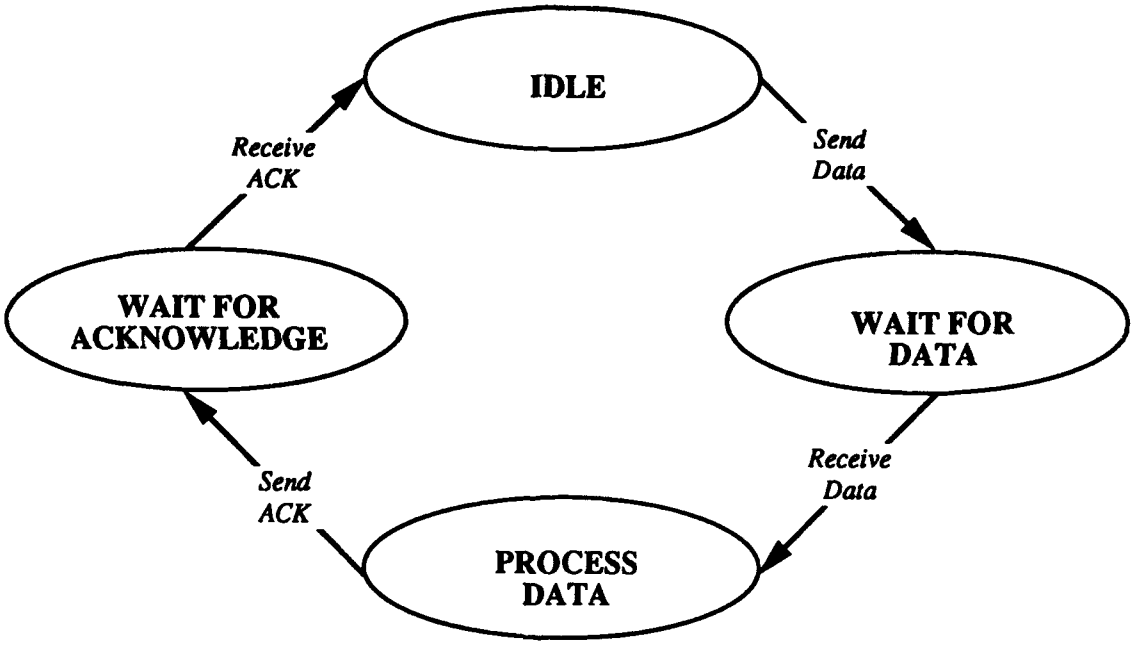



FIGURE 5.3-1. STATE MACHINE
(Merlin 1979)

Another method of representing a protocol is to use multiple FSMs. Figure 5.3-2 represents a simple protocol modeled by multiple, coupled FSMs. The receiver moves from the "Receiver Ready" to the "Receiver Busy" state on the transition caused by data reception. It moves back to "Receiver Ready" after processing is completed and the "ACK" is returned. The Link FSM shows the delivery of data from the source to the destination. It models the data transfer and the acknowledgement of the data. If the delay in the link is not significant, then the Link FSM may not be necessary and the model can eliminate this FSM. Like the receiver, the sender moves between the "Sender Enabled" and "Sender

Disabled" states based on data being sent and the corresponding acknowledgement being received. This model has the advantage of allowing implementation of each entity without the problem of having to decompose a single FSM description into the different entities. A complex FSM can be implemented more easily when it is divided into concise functional elements and modeled so that all corresponding interactions are apparent.

Transitions modeled by the FSM are considered to be instantaneous. The "Send ACK" event of the receiver occurs at the same moment as the "LINK RECEIVES ACK."

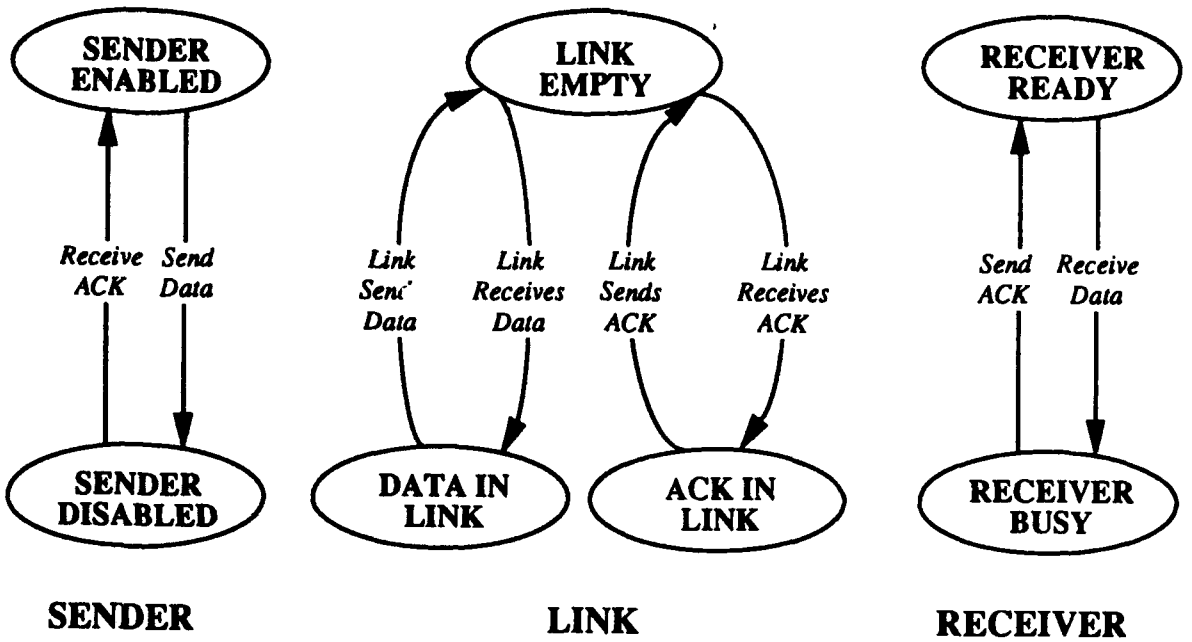



FIGURE 5.3-2. COUPLED STATE MACHINES
(Merlin 1979)

5.3.3.2 Petri Nets

Petri nets use four basic elements to represent a protocol: places, transition bars, arcs, and tokens. Places represent states in which the protocol may exist at any given moment. Directed arcs connect transitions to the places and the places to the transitions. The transition bars are transitions which may have zero or more input and output arcs. Input places of a transition are those which originate at a place and arrive at the transition. Output places of a transition are those which originate at a transition and arrive at the place. A token is indicated by a dot inside a place. (This token is not to be confused with the token in a token passing network architecture.) The following rules are given by Danthine (1977) for the operation of a transition:

- A transition is said to be enabled or fireable if each of its input places contains at least one token.

- The firing of an enabled transition consists of removing one token from each of its input places and adding one token to each of its output places.

- The firing of an enabled transition may not occur instantaneously. Firing may be considered as depending on an outside authority.

Representation of the Petri net is often done in an algebraic form resembling a grammar. Each transition contributes a rule to the grammar (Tanenbaum 1981). If a defined state of a Petri net consists of places A, C, and G, which contain tokens while the other places are empty, this state is represented as ACG. If a transition causes the tokens to move to new places such as A, D, and F, then CG → DF represents this action and is a rule for this Petri net. Since the place A is common to both states, it is eliminated from both sides of the rule.

A simple Petri net is shown in figure 5.3-3, with four places, four transitions, one token, and directed arcs between the places and transition bars. The token that initially resides at place A causes transition 1 to fire. When this happens the token is removed from A and put at B. This sequence continues through B, C, D, and finally back to A again. There is no starting point or terminating point in this model; it simply continues forever in a loop.

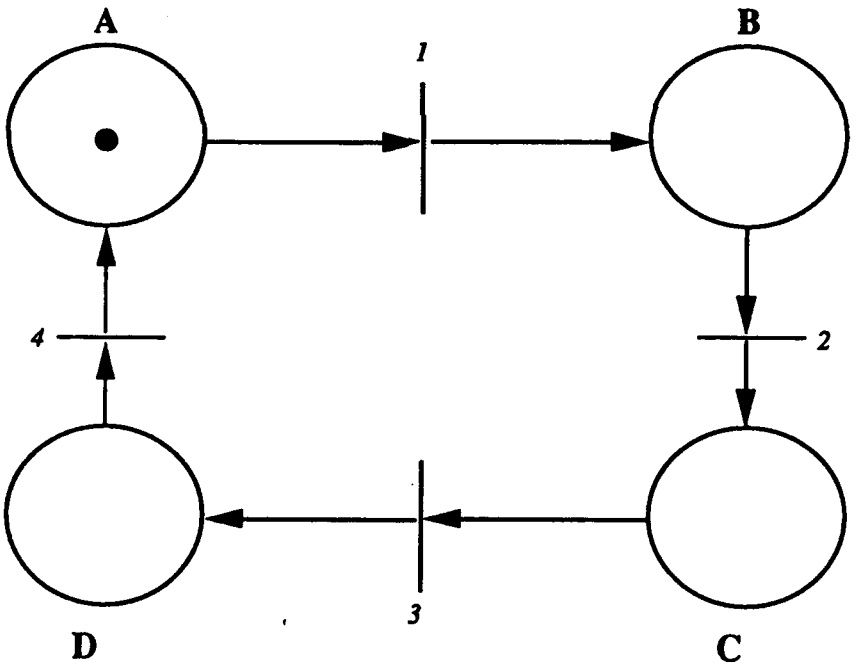



FIGURE 5.3-3. PETRI NET WITH FOUR STATES AND FOUR TRANSITION BARS

Petri nets may be used to model protocols in the same way that state machines are used. However, Petri nets have broader application in some cases. Certain resources, such as a receiver with multiple buffers, are better represented by

Petri nets than state machines. Each buffer allocation can be handled by a separate set of tokens being added to the net. Events which occur in an arbitrary order are also easily represented by the Petri net.

#### 5.3.3.3 Other Methods of Protocol Representation

FSMs and Petri nets are common methods for protocol representation, but not the only ones. High-level programming languages and grammars are used to model protocols and have the advantage of being easily understood since they appear more like natural language. Advantages of using a high-level language include ease of representing counters, data, and variables. Complex control structures, on the other hand, are difficult to represent and understand when represented by a high-level language.

The use of a high-level programming language to model a protocol by its very nature comes close to an actual implementation of the protocol. Also, the unique characteristics of the programming language, which may be nonessential to the protocol model, will be obvious in the implementation.

### 5.3.4 Protocol Verification Methods

With a shift from unidirectional to bidirectional data buses, the access protocol assumes an added degree of complexity. As complexity increases, so should the concerns that relate to protocol verification. Verification involves demonstrating that the interactions of distributed protocol modules satisfy the service specification of the protocol (Sunshine 1979). A protocol may logically meet all the requirements of the specification, but this does not guarantee that a particular implementation is correct.

Also a particular protocol of layer, n, may meet the requirements but its correct operation is based on the service provided to it by the n-1 layer. For example, if the Network Layer is not operating correctly, the cause may be in the physical or Data Link Layer which the Network Layer relies on.

Certain general properties of any protocol may be checked. Areas which should be checked are as follows (Merlin 1979):

- Deadlock Freeness
- Liveness
- Tempo-Blocking Freeness
- Starvation Freeness
- Recovery from Failures
- Self Synchronization
- Correct Execution of the Purpose of the Protocol

Deadlock Freeness means that the protocol will not terminate. There should exist no states in the protocol design or implementation which are terminal. Liveness shows that from a given reachable state, any other state can be reached. Tempo-Blocking Freeness simply checks that there is no infinite looping. Starvation Freeness means that no process will forever be prevented from acquiring an available resource. Recovery from Failures states that when a failure occurs, the protocol operation will return to the normal execution within a finite number of states. Self Synchronization means that from any abnormal state the protocol will recover within a finite number of states. Correct Execution of the Purpose of the Protocol means that a protocol is doing what it was designed to do.

#### 5.3.4.1 Global State Generation

These are the particular properties that are checked for a protocol, but the method used to apply these checks varies. In a protocol modeled by an FSM or a Petri net, one of the more common methods of verification is called global state generation. Global state generation is implemented by starting with a given initial state and identifying all possible transitions from that state to another state. Each of the new states is examined until no new transitions are identified. Some transitions may lead back to a state already encountered. When this is complete, all possible outcomes of the protocol are known and observations may be made concerning the properties listed above.

Global state generation has a limitation. It may only be used on protocols that can be represented with a finite number of states. An advantage of this technique is that it may be easily mechanized for automatic testing of certain properties.

#### 5.3.4.2 Assertion Proving

Assertion proving is another technique for verification. This is applied to the protocol and its description as though they were parallel programs. Assertions are made about certain variables based on the description. If the protocol and description compare at predetermined points, then the proof holds. The assertion proving method is commonly used with protocols that have many states. Assertion proving requires special considerations for implementation. Therefore, it does not lend itself to automation as the method of global state generation does.

#### 5.3.4.3 Other Verification Methods

Two other methods in use are "induction over the topology" and "adherence to sufficient conditions." In the first method, the holding of a property or occurrence of an event is proven by showing that certain conditions will propagate throughout the topology (Merlin 1979). If a certain property holds for a system with x entities, then it will also hold for a system of x+1 entities. The latter method uses constructive design rules that automatically result in correct protocols. For instance, for every send transition implemented by the designer, the design rules specify the corresponding receive transition of the peer entity (Bochmann and Sunshine 1980). At each step in the design, the protocol is checked to ensure that it satisfies the properties it specifies.

There is no one method that can be applied easily to all protocols. Depending on the complexity and the topology, one method may be preferred over another. In cases where the state explosion becomes a problem, such as for complex protocols, it is sometimes necessary to make simplifications of the model for the purpose of verification.

### 5.3.5 Application to Avionic Data Buses

Protocols may be implemented in hardware as well as in software. Most of the hardware used in avionic data buses, such as the ARINC 629 bus, ASCB, and MIL-STD-1553 bus, has been implemented in a single high-density IC or a combination of several high-density ICs. When this is done, the protocol is not accessible to examination and scrutiny as is a software-implemented protocol. As avionic systems using data buses increase in complexity, so do the protocols and the bus hardware used to implement the protocol.

A protocol may be implemented in any of the seven layers of the OSI Basic Reference Model, from the Physical Layer to the Application Layer. The complexity of a protocol is not the same from one level to the next. At the Data Link Layer a protocol may be straightforward, but at the Network Layer become highly complex and, therefore, difficult to model.

As seen in the ARINC 429 bus standard, a more complex bit-oriented protocol is added on top of the previously defined physical and Data Link Layers. This can be done with any data bus, whether it is unidirectional or bidirectional. Since protocols may be layered in this manner, some data bus standards, such as ARINC Specifications 429-12 and 629, have defined protocol transactions which can be used at the higher layers.

Although a data bus may be implemented strictly in hardware, it should not be treated any differently in the areas of specification and analysis than a software-implemented protocol. Hardware-implemented protocols should be subjected to rigorous analysis, like that specified in RTCA/DO-178 for avionic software.

#### 5.3.5.1 ARINC 429 Bus

The ARINC 429 DITS is a unidirectional broadcast type bus with only one transmitter. Access to the bus by the transmitter is not a matter of contention. Another factor contributing to the simplicity of this protocol is that it was originally designed to handle "open loop" data transmission. In this mode there is no required response from the receiver when it accepts a transmission from the sender. The system simply depends on the integrity of the shielded twisted pair of transmission lines, a data integrity test using parity, and data reasonableness checks by the host processor.

With an increasing need for more functions to be handled by the data bus, a new protocol was developed and has been incorporated into the standard. This protocol is bit-oriented, as described in section 2.5 of ARINC Specification 429-12, and is used along with the previously defined character-oriented protocol. It is intended to be used for the transfer of data files from one bus

member to another using techniques that are common to computer networks to ensure safe and orderly delivery of data files. Four layers of the OSI Basic Reference Model are described for use with the bit-oriented protocol: Physical, Data Link, Network, and Transport Layers. Labels, timing, and protocol transactions are described as well. The protocol transactions specify an orderly and controlled transfer making use of closed-loop control. Commands such as RTS and CTS are used along with timeouts, which are required on all transactions.

The use of a data bus in this manner can be modeled with a Petri net and tested for all the properties a protocol should have, such as Deadlock Freeness, Liveness, Recovery From Failures, etc. It is beyond the scope of this chapter to attempt to model this protocol, but such analysis should be done by a developer for verification purposes.

5.3.5.2 ARINC 629 Bus

The ARINC 629 bus is a bidirectional bus with multiple transmitters and receivers. Access to the bus by all transmitters must conform to a thoroughly tested integration standard.

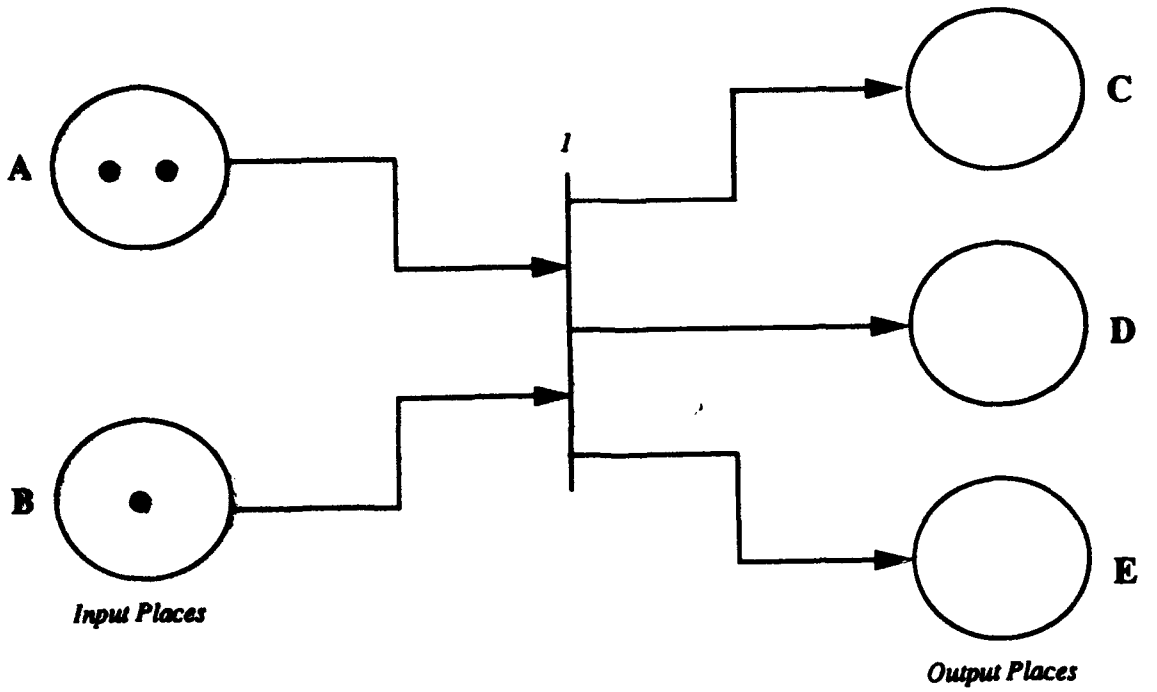
At the lower levels, the access protocol is implemented in hardware. The protocol at this level may be analyzed by an FSM or a Petri net method. Included in Attachment 7 of the ARINC 629 bus specification is a state diagram of the overall access protocol. The purpose is to give an overview of how a terminal accesses the bus for a particular operational mode. Some of the other functions of the hardware that could be modeled for verification are self monitoring, interaction with the host CPU, the data bus, various timers, and error checking and handling. If the complete actions of the hardware were modeled, the state diagram would be quite complex.

The state diagram of the CP is included for reference in figure 5.3-4. The diagram shows the general actions of the access protocol based on the three defined levels of access: L1, L2, and L3. The conditions for transition to the next state are also shown. In the ARINC 629 bus specification, each of the three levels of access are expanded to one complete page in Attachments 7c, 7d, and 7e, respectively.

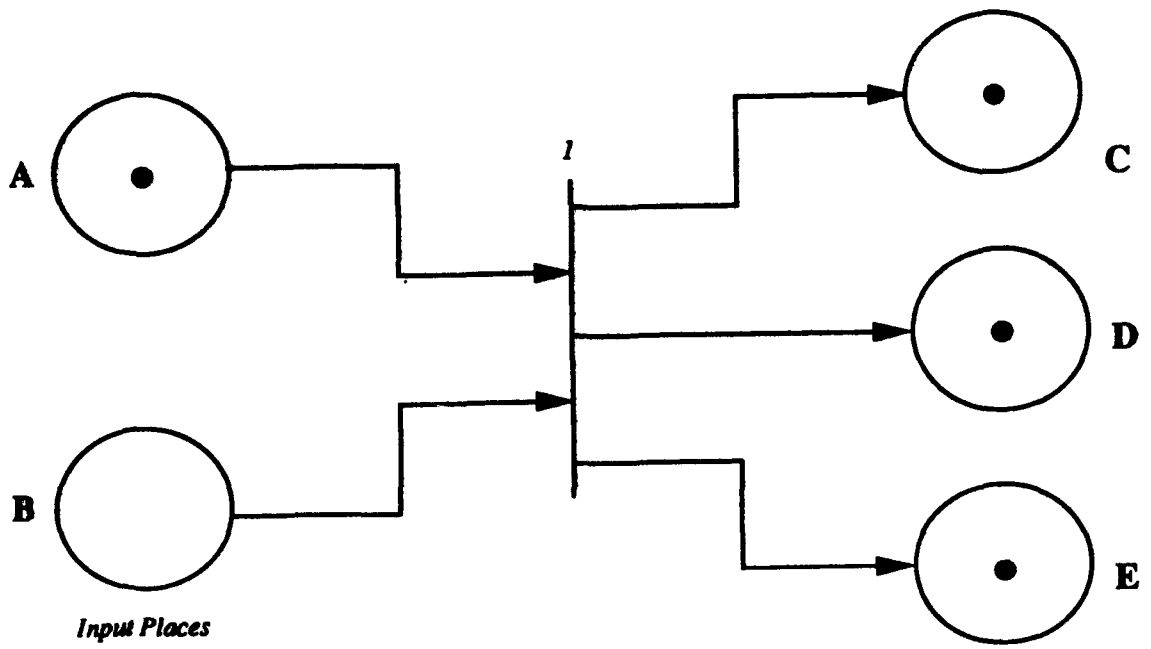
Figure 5.3-4 shows, in a general manner, how a terminal acquires the bus for a transmission when using the CP. The three levels of access and the various timers are explained in section 5.1 of this chapter.

As with the ARINC 429 DITS, this standard also defines the data bus in terms of the layers of the OSI Basic Reference Model. Not only is it designed for use as a broadcast bus, but it is also intended to be used as a closed-loop system, as stated in ARINC Specification 629, Part 1, section 6.3.1 (1990):

> "Directed messages may or may not be used to direct information between two systems so that handshaking protocols may be established for message checking capability."




**BEFORE FIRING**




**AFTER FIRING**

FIGURE 5.3-4. ACCESS PROTOCOL OVERVIEW FOR ARINC 629 BUS
(ARINC Specification 629, Part 1, Attachment 7b, 1990)

In this closed-loop mode it is necessary to define a complete protocol which utilizes the services provided by the lower protocol layers (the ARINC 629 data bus). This protocol should define parameters that may be used for directed data transfer such as an acknowledgement response, some form of flow control, the data transfer and data structures, and timeout conditions. When all of the necessary parameters are specified, along with the rules governing their interactions, it is possible to subject the protocol to analysis as previously defined in section 5.3.

5.3.5.3 ASCB and MIL-STD-1553 Bus

A data bus that uses a form of central control can be examined using the analysis techniques presented in this section. When a command is issued on the bus, a response is anticipated from the addressed terminal. Whether the response occurs or not, the timeout conditions, the number of retries, and the error handling can all be modeled for the interaction with the terminal. A global network model can be created and checked against the specification for correct operation. When redundant central controllers are used, there needs to be a clear definition of the interaction between them for detecting and handling controller error conditions. Modeling can provide this clarity.

The ASCB is implemented as an open-ended protocol where the response from the terminals is not checked by the BC. Therefore, no end-to-end interaction may be modeled for the ASCB specification.

The MIL-STD-1553 is a candidate for analysis by use of formal techniques. The interactions can be formally examined for any problems using the guidelines previously set forth in section 5.3.

5.3.6 Summary

A data bus specification should address integration problems by defining the hardware as completely as possible. A data bus specification addressing a software protocol should also be complete to avoid future integration problems. Questions need to be asked concerning these protocol specifications and implementations, such as the following:

- Is the protocol implementation correct according to the specification?
- Is the protocol specification complete?
- Do all systems have the same timeout values for every timeout condition?
- Do all systems have the same retry value?
- How can the protocol parameters be tested under every possible condition?
- Have all the properties of the protocol been checked?

The fundamental question that needs to be addressed on this topic is "Has the protocol been completely specified and verified by the use of formal methods?"

When formal methods are used, the implementor may have more confidence in the protocol.

## 5.4 Bus Integration Standards, Guidelines, and Techniques

A typical avionics system consists of several subsystem boxes that are connected in a unique arrangement to input sensors, output devices, and each other to produce the functionality required for a particular aircraft. The intercommunication is provided by one or more digital data buses, as well as some analog data buses and point-to-point wiring.

Each subsystem is typically designed independently of the others. They may even come from different manufacturers. This section describes the standards, guidelines, and techniques used to ensure that data buses reliably integrate the subsystems that they interconnect. This section also addresses how these integration aids relate to the certification of aircraft. For additional information, integration aids for buses that are used primarily in military applications are included. A list of some of these documents is given in table 5.4-1. For more detail, refer to the bibliography.

### 5.4.1 Levels of Integration

There are several levels at which reliable integration of subsystems must be ensured. The lowest level is the physical integration of the hardware. Physical integration includes mechanical and electrical aspects. For the subsystem hardware to be properly integrated, the pieces must be mechanically compatible and the bus interface of each subsystem must obey the bus standards for voltage levels, signal encoding, signal timing, and other electrical characteristics. These specifications must not be exceeded for any configuration that the system might take on, and for any environment in which the system might be placed. Integration at this level is essential for bus messages to be generated and received.

The logical integration of the hardware is the next level of integration. The hardware protocol defines the sequence of bits that constitutes the smallest unit of data that can be transferred on the bus as a legal message. Bus messages form the building blocks for all higher level transfers of information. The subsystems must obey the bus standard for the timing, sequence, and polarity of each bit in a bus message. This ensures that all messages are encoded and decoded into the proper sequence of synchronization bits, start-of-message bits, control bits, address bits, data bits, status bits, error detection bits, error correction bits, and/or end-of-message bits. The patterns of many of these groups of bits must obey certain rules. If there are exceptions, the hardware produces a bus message error signal. Integration at this level also is essential for bus messages to be generated and received.

TABLE 5.4-1. INTEGRATION STANDARDS AND GUIDELINES, BY BUS
(PART 1 OF 2)

| Document Name | Publisher |
|---|---|
| ARINC 429 Bus | |
| ARINC Specification 429-12 | ARINC |
| ARINC 429 Supplement | GAMA |
| ARINC 429 Receiver/Transmitter | Western Digital |
| ARINC 429 Bus Interface Circuit | Harris Semiconductor |
| ARINC 429 Bus Interface Line Driver Circuit | Harris Semiconductor |
| Application Note No. 400 | Harris Semiconductor |
| ARINC 629 Bus | |
| ARINC 629 Part 1, Technical Description | ARINC |
| ARINC 629 Part 1, Supplement 1 | ARINC Draft |
| ARINC 629 Part 1, Supplement 2 | ARINC Draft |
| ARINC 629 Part 2, Applications Guide | ARINC Draft |
| ARINC 629 Part 3, Data Standards | ARINC Draft |
| ARINC 629 Part 4, Test Plan | ARINC Draft |
| ARINC 629 User's Manual | BCAC |
| ARINC 629 Terminal Device | LSI Logic |
| ARINC 629 Communication IC | National Semiconductor |
| ARINC 629 Serial Interface Module | SCI Technology |
| ARINC 629 Current Mode Coupler | SCI Technology |
| ARINC 629 Serial Interface Module | AMP/Dallas Semiconductor |
| ARINC 629 Current Mode Coupler | AMP/Dallas Semiconductor |
| GAMA CSDB | |
| GAMA CSDB | GAMA |
| EIA RS-422-A | EIA |
| GAMA ASCB | |
| GAMA ASCB | GAMA |
| EIA RS-422-A | EIA |
| WD193X Synchronous Data Link Controller | Western Digital |
| ASCB Data Link Coupler | SCI Technology |
| MIL-STD-1553 Bus | |
| MIL-STD-1553-B Standard | Military Standard |
| MIL-HDBK-1553-A Handbook | Military Standard |
| SAE AE-12 Systems Integration Handbook | SAE |
| SAE AS4112 RT Production Test Plan | SAE |
| SAE AS4113 BC Validation Test Plan | SAE |
| SAE AS4114 BC Production Test Plan | SAE |
| SAE AS4115 System Test Plan | SAE |
| Multiplex Applications Handbook | AFSC |
| Multiplex Applications Handbook Addendum | AFSC |
| MIL-STD-1553 Designer's Guide | Data Devices Corporation |

TABLE 5.4-1. INTEGRATION STANDARDS AND GUIDELINES, BY BUS
(PART 2 OF 2)

| Document Name | Publisher |
|---|---|
| **SAE LTPB** | |
| AS4074.1 Standard | SAE |
| AIR 4288 Handbook | SAE Draft |
| AS4290 Test and Validation Plan | SAE Draft |
| **SAE HSRB** | |
| AS4074.2 Standard | SAE |
| AIR 4289 Handbook | SAE Draft |
| AIR 4291 Test and Validation Plan | SAE Draft |

The logical integration of the software is the next level of integration. Although the hardware protocol usually permits all possible permutations of control, address, data, and status bits, a particular system usually supports only a few of the possibilities. The software protocol determines the legal field formats and message sequences. The subsystems must obey the software protocol standards to be integrated into a reliable system. Otherwise, legal bus messages might not reach their proper destination or might not be properly interpreted.

The final level of integration occurs at the functional level. The function that each subsystem is to perform in response to a received message must be consistent with the intent of the subsystem generating the message. The application programs must all use the same data definitions. At this level, the content of the messages becomes important. The subsystems must obey the system standard for legal communications at this level to be integrated into a reliable system.

The first three levels of integration are clearly bus-dependent integrations. It would appear that functional integration is not a bus integration issue. It is primarily the concern of the system specification, rather than a bus standard. However, since every LRU which communicates on a bus must use the same data definitions, the job of standardizing the definitions has been relegated, in many cases, to the bus standard. In fact, not only do the bus standards define the acceptable data words, but many of the protocols accept no other data types. Many of the buses do not transparently transfer whatever data the LRUs wish to transmit.

### 5.4.2 The Ideal Bus Integration Standard

Certainly a bus integration standard requires that, at a minimum, the bus medium and the LRUs satisfy a bus standard which specifies items such as the following:

- Bus medium

- Bus connectors
- Electrical characteristics that all signals on the bus must satisfy
- Logical characteristics that elementary messages on the bus must satisfy
- Electrical characteristics that each LRU must satisfy
- Logical characteristics that each LRU must satisfy
- Environmental conditions under which the equipment must operate
- Electromagnetic requirements that all of the equipment must meet
- LRU test procedure

But this is not sufficient because problems that are unique to a particular system configuration are uncontrolled by such a bus standard. For instance, the bus standards do not specify the interactions of multiple LRUs in a system. That is left for the system specification. A bus integration standard that is designed to control the integration of LRUs must also specify the following integration specific items:

- Physical layout of the bus
- Control, address, and status words that are allowed on the bus
- Interpretation of the allowed control, address, and status words
- Data words that are allowed on the bus
- Interpretation of the allowed data words
- Integration test procedure

Where possible, control of these items should be accomplished by precise specification. Where greater flexibility is required, the standard should use formal guidelines. These must consist of precise definitions with formulas, tables, rules, or flowcharts that constrain the system designer to produce working configurations.

None of the avionic data bus standards qualify as bus integration standards by these criteria. Some of these integration-specific topics are either not addressed at all or are only discussed, as opposed to specified. Furthermore, no generic bus integration standard was discovered by these researchers for avionic buses.

### 5.4.3 Bus Integration Standards and Guidelines

The standards that address system integration by data buses consist of the data bus standards and the data bus test standards. These standards regulate the integration of subsystems to varying degrees. Generally, they do not address the integration-specific topics directly.

All of the data bus standards specify, to some extent, the physical makeup of the bus conductors. They generally do not specify the physical layout of the system. That is unique to each system. Nevertheless, some of the bus standards at least address the effects that the system layout has on the electrical characteristics of the bus. All of the bus standards specify the electrical and logical hardware requirements for each LRU attached to the bus. They do not all address the electrical and logical characteristics of multiple LRUs interacting on the bus. All of the bus standards also address the software protocol that each LRU must obey. However, they cannot specify the content of the control, address, data, and status words for a particular system. These are unique to each LRU and system.

Elwell et al. (1992) delineate the strengths and weaknesses of each set of bus standards as they apply to subsystem integration. Guidelines are also discussed, whether they are part of the standard or not.

### 5.4.4 Bus Integration Techniques

The complexity of the interactions among LRUs on bidirectional buses has motivated many data bus designers to use various design analysis techniques when designing and certificating systems that use data buses. Typically, conventional computer system design techniques are adapted to the unique requirements of data bus development. These techniques use mathematical or otherwise logical constructs to represent the system being designed. The representation is then exercised and tested to determine if the real system most likely has, or will have, the desired characteristics. Each technique emphasizes a particular characteristic. The goal of these techniques is usually to increase the confidence that the system will always satisfy the requirements for it. Using these techniques could give the developer confidence that an aircraft is worth building, or give a CE confidence that a built aircraft should be TCed.

Some of these techniques are described and their application to data bus design and analysis presented. Their use in certification will be addressed later. A list of the techniques documents is given in table 5.4-2.

Some of the techniques described in the following sections are recommended or required by the data bus documentation. Specifically, MIL-HDBK-1553 states, "It is essential that a proposed network be simulated before the design is finalized." (MIL-HDBK-1553A, 1988). The HSRB test plan requires that the station tester emulate the host and all other bus stations. Similarly, both the LTPB and the HSRB specify the use of analysis in their quality assurance plan. Some of the analysis techniques are recommended in the certification documentation. In AC 25.1309-1A, the Functional Hazards Assessment, FMEA, and FTA are all offered as acceptable means of showing compliance with RTCA/DO-178. Other techniques are commonly used by developers simply as good engineering practice.

TABLE 5.4-2. INTEGRATION TECHNIQUES DOCUMENTS

| Document Name | Reference |
|---|---|
| Computer Resources Handbook for Flight Critical Systems | Hecht and Hecht 1985 |
| Fault/Failure Analysis for Digital Systems and Equipment | ARP 1834 |
| Procedures for Performing a Failure Mode, Effects and Criticality Analysis | MIL-STD-1629A |
| Fault Tree Handbook | Vesely et al. 1981 |

5.4.4.1 Modeling

Modeling consists of creating a system of mathematical equations that formulates all the significant behavior of the system being modeled. The reliability of the system is a common behavior of interest.

In unidirectional broadcast buses, the bus is little more than a transmission medium, since all of the communications control is embedded in the LRU software. For these buses, modeling is used only to analyze the behavior of the electrical signals on the bus. The standards specify this behavior for an ideal bus. Modeling is necessary to confirm that a particular implementation, with multiple LRUs, specific bus lengths, and specific LRU separations, conforms to the ideal. This means that a particular layout of a bus must be sufficiently characterized so that the shape of the signal waveform can be calculated for any point on the bus at any time in the sequence of transmissions. This was done for the Mark 33 DITS, for various configurations, to confirm that distortions remain within the permissible limits of the waveform. The waveforms are presented in appendix 1 of ARINC Specification 429-12. The ARINC 629 bus standard provides for the use of this kind of analysis also. Although ARINC 629 bus operation has been established for lengths up to 100 meters, "A systems designer may extend the bus length if proper analysis demonstrates that there is no loss of bus integrity." (ARINC Specification 629, Part 1, 1990).

A model of the electrical characteristics of a bus network is usually used to aid the engineer when developing a design. A tentative layout for integrating multiple LRUs can be set up in the model, and the electrical behavior checked for unanticipated problems. As various layouts are checked, the iterative process guides the designer toward a trouble-free solution. This technique turns trial and error learning into a convergent engineering design process.

For bidirectional buses, bus communication is controlled by a computer system of its own. Bus transmissions are controlled by a state machine, implemented with hardware and software, that serves no function for the LRU except to control bus communication. This computer system can be quite complex, involving a protocol that controls numerous unique interactions in an environment that requires fail-safe operation. The reliability required of a bus used in critical avionics may be provided by a fault tolerance scheme that is distributed across hardware and software features and even across LRUs. The design of such a system is greatly dependent upon the use of modeling.

The Computer Resources Handbook for Flight Critical Systems (Hecht and Hecht 1985) presents a simple analytic model for assessing the reliability, availability, and fault tolerance of a system. An analytic model allows the designer to evaluate the likely outcomes of system design decisions and gives insight into the behavior of the design.

A thesis from the Naval Postgraduate School gives a good example of how modeling is applied to a complex data bus network (Nelson 1986). In that study, a computer architecture that uses advanced hardware-software reliability techniques is modeled for the purpose of determining a design that can meet FAA safety requirements for critical systems. Conventional reliability analysis is inadequate, since it is based on hardware reliability alone. In this case, the reliability was based on the reliability of the components of the system, and the capability of the system to identify correctly both the occurrence of a fault and its precise location within the system configuration. (This is exactly what is done in the bidirectional bus protocols as they try to ensure that no two transmitters attempt to operate simultaneously.) A Semi-Markov analysis computer program was used to create the model. This model was used to generate a configuration that met the safety requirements.

The Hybrid Automated Reliability Predictor (HARP) embodies yet another approach to the modeling of computer systems that use advanced reliability techniques (Bavuso et al. 1987). It addresses a weakness of structural decomposition methods. In order to do a structural decomposition, the fault tolerant behavior of a system must be able to be partitioned along with the mutually independent subsystems. This often is not the case. The HARP program uses behavioral decomposition instead. Bavuso et al. applied the method to two flight control systems as examples.

Some models are more general purpose. Parhami (1979) developed an approach to modeling bus redundancy. The model can be used to assess the tradeoff between increased redundancy and increased complexity for single and multiple bus systems.

#### 5.4.4.2 Simulation

Simulation is very similar to modeling. Simulation consists of creating a system of mathematical equations that formulates all significant contributions to the behavior of the system being modeled. Simulation, however, assumes that

the system exists. A simulation usually combines a computer program emulation of most of the functions of the system (before they are implemented) with some of the actual hardware. Simulations that rely heavily on emulation are sometimes called emulations.

Since real, rather than proposed, behavior is modeled by a simulation, the model can be, and should be, validated. The response of the simulation to a particular real-life scenario is compared against the response of the real system. Once the simulation is validated, it is used to do analyses which would be too costly in time, money, or risk to perform on a real system.

The ARINC 629 bus, MIL-STD-1553 bus, LTPB, and HSRB all rely on simulation for the validation of a particular bus network. The LTPB handbook includes a program listing that can be used to simulate the priority scheme of the protocol. This simulation aids the system designer in choosing protocol parameters while the bus design is still only on paper. The HSRB test procedure requires a simulator that can emulate a host and all other stations. Simulators are also used to test and evaluate ARINC 429 buses.

The United States Air Force Aeronautical Systems Division defined guidelines for the development of computer programs used in digital flight control systems. Sylvester and Hung (1982) present the concepts for V&V of these systems that require extreme reliability. They found that,

> "The key to the development approach leading to V&V is the consistent and integrated use of models and simulations. The verification of such simulations with ground and flight test information leads to validation of flight control system concepts and implementation." (Sylvester and Hung 1982).

They proceed to present a conceptual framework where the problem of design and test of highly reliable systems may be studied. The design process should start with a functional simulation, validated against experimental data and analysis. As it continues, the simulation should evolve into a simulation test facility which uses as much of the prototype hardware as possible. In the testing phase, flight tests should be instrumented to gather data to confirm that the earlier simulations were valid. Sylvester and Hung also describe an entire system of simulation plans and reports, and a cross-reference index for the integration of simulation into the design process.

The need for early validation of complex computer systems is also addressed by Karmarker and Clark (1982):

> "Few automated or semi-automated techniques, however, have been developed to address the verification of the very early development stages, namely system requirements and system design. Instead modern practice relies on formal and informal reviews, and analytical studies and trade-off analyses of various aspects of the system design."

They present a tool and a development methodology for using a system level emulation to perform this early validation. They have applied the technique to a flight control system.

The National Aeronautics and Space Administration has also investigated using emulation as a technique for validation, rather than relying only on analytical modeling. Becher writes, "ways must be found to reduce the risk caused by these new technologies" (Becher 1987). Becher developed an algorithm to emulate the hardware of complex integrated computer systems as logic gates, flip-flops, and tri-state devices. The emulations are used as general reliability analysis tools in the Avionics Integration Research Laboratory (AIRLAB). Such an emulation also lends itself to using fault injection to determine the response of the system to faults.

Petrichenko (1988) writes on some lessons learned from doing simulation. The article gives a good introduction to some of the basics of simulation techniques, particularly for hardware-in-the-loop simulation. He observed that an added benefit of creating a simulator is that it functions as an independent development of the same function as the system being simulated. As a result, when the logic of the two differ, often the system logic may be found to be faulty.

Hecht and Hecht (1985) address the simulation of reliability models. They point out that simulation allows complex models to be evaluated for the system failure modes. Furthermore, a simulation can be tailored to concentrate on the unlikely problem areas that are of particular interest in critical systems. They discuss some general-purpose simulation programs that can be used. Bannister et al. (1982) also address the evaluation of which design is best for a particular application. They state, "Simulation and analytical tools are the time-proven means for the precise evaluation of a given design." They then discuss some software tools that can be used for this purpose.

Simulation is taken one step further with the ARINC 429, CSDB, and MIL-STD-1553 buses. Manufacturers make black box testers that are used to simulate an LRU connection to the bus. They are made to generate and evaluate messages according to the electrical and logical standards for the bus. They consist of a general purpose computer connected to bus interface cards. The simplest ones may simulate a single LRU transmitting or receiving. The most complex ones may be able to simulate multiple LRUs simultaneously, as well as a BC, where applicable (McCartney and Phillips 1981).

These simulators are invaluable for system integration in highly integrated systems. In such systems, a single LRU cannot be tested without the entire system being present. Testing should not be held off that long. Furthermore, the correctness of the data bus itself must be checked before LRUs can be installed (Sawtell and Dawson 1988). These simulators provide a solution to both problems; they can be used to verify bus operation and to simulate the other LRUs in the system. Fitzgerald and Polivka (1982) also point out the usefulness of a system tester that can be used in data bus system development and integration testing.

Although simulation can be used to detect many integration problems, a simulation cannot be run for the many hours required to prove that a system will not have a critical failure more often than $10^{-9}$ per flight-hour. VanBaal (1985) states it well:

> "To put this figure into perspective, it should be realized that the total accumulated number of flight-hours on turbo-jet powered airplanes since their introduction in 1957 is estimated to be in (sic) the order of $3 \times 10^8$. It is thus easily seen that proof of such low probabilities by means of ... simulation is highly impractical."

As a result, more analytical techniques should be used to support simulation. Some of these techniques are discussed in the following sections.

5.4.4.3 Fault/Failure Analysis

F/FA is a general term for analysis techniques used to identify systematically and, possibly, quantify the effects of hardware failures on a system. Modern data bus interfaces are sufficiently complex to warrant such an analysis. The techniques usually implied are those of the formal F/FA process defined by ARP 926A. The application of these basic techniques to processor-based digital electronics is presented in ARP 1834.

ARP 1834 discusses, in detail, the purposes of F/FA, the types of F/FA, and the considerations to be made in choosing which techniques to use and how thorough an analysis to perform. The desired effect is to produce a credible statement of the possible faults and their effects in the most cost effective manner. The analysis techniques fall into two classes. Those that analyze the faults from the top-down and those that analyze from the bottom-up.

In the first case, the analyst postulates the undesirable system effects and deduces from them what subsystem faults could produce such an effect. The analyst asks the question, "How can this failure occur?" Each subsystem fault is then analyzed to determine what lower level fault would cause the subsystem fault. Each of these branches is expanded until they are terminated by faults considered to be sufficiently controllable or sufficiently unlikely. Top-down analysis has an advantage in that it can be performed on design models. A disadvantage is that it does not guarantee that every possible fault is identified. ARP 1834 covers the use of FTA for a top-down approach. An example is given in appendix 2 of the ARP.

In the bottom-up method, the system components and their relationships are known. The analyst identifies every possible failure mode of each component at the level of interest and then deduces the effect each failure would have on the next higher level. This procedure answers the question, "What failures are possible?" This method is exhaustive. It covers all the bases, but it can become unmanageable for complex systems. A bottom-up analysis of IC-based circuits must be initiated at some level higher than the component level to be feasible. The process defined in ARP 1834 covers FMEA for a bottom-up approach.

The two approaches tend to be complementary. The F/FA process provides for using both approaches. Since both approaches rely on the ability of the analyst to think of failure modes and their implications, it is essential that a well-coordinated team effort be used to conduct a correct and comprehensive survey of all system faults and their effects (Vesely et al. 1981). Failure Mode, Effects, and Criticality Analysis (FMECA) and fault insertion are also presented as available methods. Each of the techniques incorporated by ARP 1834 is defined apart from that document and can be used independently of it. For this reason, they are discussed individually in subsequent paragraphs. Additional discussion is presented in chapter 3 (Curd 1989) of this handbook.

5.4.4.4 Fault Tree Analysis

FTA is a method that helps ensure that decisions about a system are based on all known pertinent information on the system. In particular, a decision about the likelihood of a certain undesirable event occurring should take into account the implications of all credible ways in which the event can occur. FTA does this by providing a directed, disciplined process for identifying the failure producing faults. Furthermore, the analysis recognizes that a complex system is more than the sum of its parts. Component interactions determine much of the character of a system. Thus, FTA is particularly appropriate for analyzing bus-integrated avionics for problems that are peculiar to the system interactions. FTA is usually used for analyzing hardware faults, but application has been made to software and computer systems (Hecht and Hecht 1985).

FTA is recommended by ARP 1834 as a component in an F/FA. FTA of hardware is defined, explained, and demonstrated in the Fault Tree Handbook (Vesely et al. 1981). They explain,

> "Fault Tree Analysis is a deductive failure analysis which focuses on one particular undesired event. The undesired event constitutes the top event in a fault tree diagram. Careful choice of the top event is important to the success of the analysis. If it is too general, the analysis becomes unmanageable; if it is too specific, the analysis does not provide a sufficiently broad view of the system. Fault tree analysis can be an expensive and time-consuming exercise and its cost must be measured against the cost associated with the occurrence of the undesired event." (Vesely et al. 1981).

Because fault trees can easily become unmanageable, Hecht and Hecht (1985) suggest that FTA be used to identify the critical events at the subsystem level, then use FMECA to determine the potential causes of these events.

A typical fault tree is shown in figure 5.4-1.

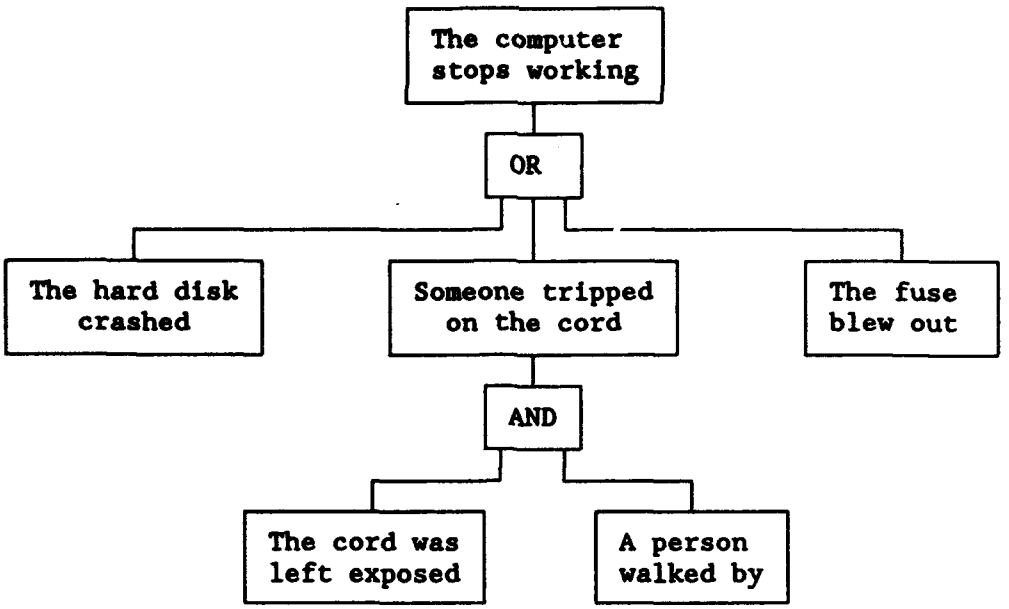



FIGURE 5.4-1. TYPICAL FAULT TREE

The fault tree is produced by a directed qualitative process. However, once the tree is produced, a quantitative probability of the undesirable event occurring can be calculated. The FTA can be used for either purpose: simply to identify the causes so that they can be controlled, or to calculate the probability given the set of causes.

A quantitative analysis is shown in figure 5.4-2. It is calculated as follows:

$$2.3 \times 10^{-4} \approx 5.7 \times 10^{-5} + (7 \times 10^{-4})(0.2) + 2.9 \times 10^{-5}$$

where it is assumed that every person that walks by will trip on the cord.

A particular fault tree only accounts for the effects of the most credible attributing faults, as thought of and assessed by the analyst. It is not a model of all possible system failures or all possible causes for system failure. To accomplish that, the analyst must identify every possible system failure and develop the fault trees for each of them.

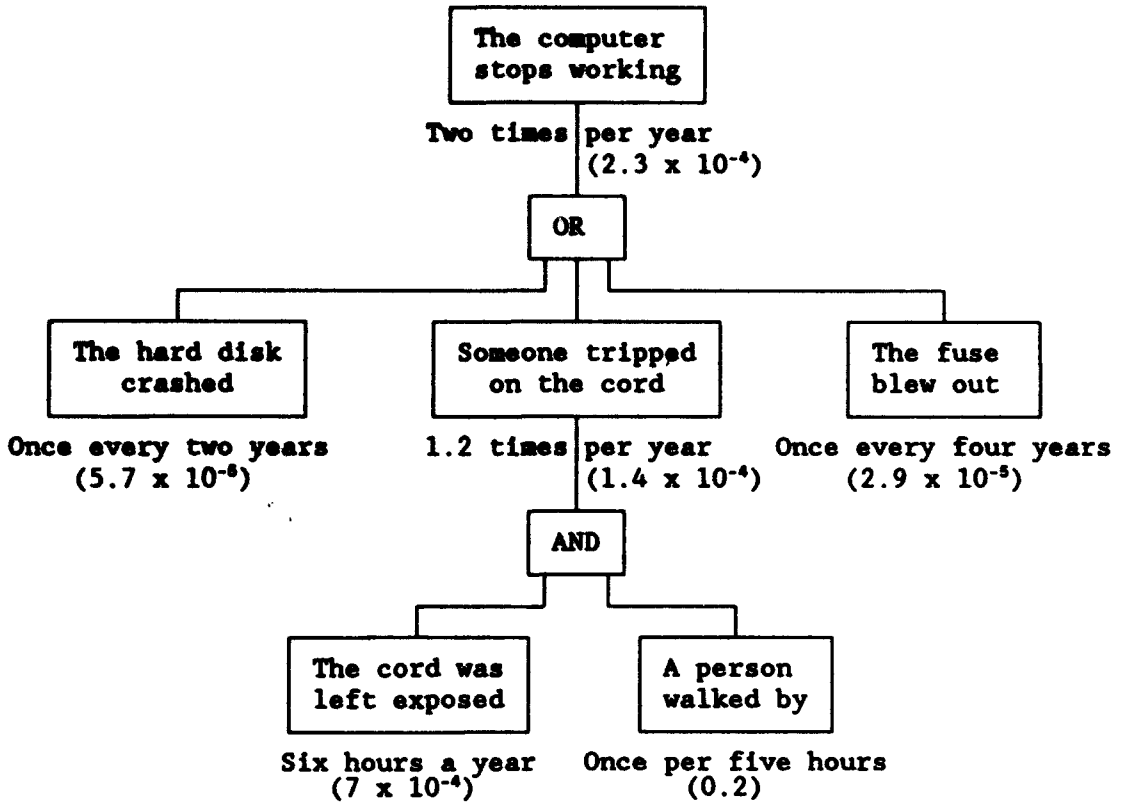



FIGURE 5.4-2. QUANTITATIVE FAULT TREE ANALYSIS

5.4.4.5 "Parts Count" Failure Analysis

If it is assumed that the failure of any single component in a system will cause a system failure, the probability of system failure is simply the sum of the individual failure probabilities. The analyst counts the number of each component, multiplies each of these by the probability that the component will fail, and then adds these together. This probability is the most conservative estimate, since all dependencies are covered. Thus, if the system or subsystem failure probability is sufficiently low using the parts count method, then it will be found to be sufficiently low by any more refined method. The additional detail of those methods would be unnecessary. Some of these more refined methods are discussed in the following paragraphs.

5.4.4.6 Failure Mode and Effects Analysis

FMEA is a systematic analysis of failures and their effects, that uses an inductive, bottom-up approach. It is one of the techniques recommended by ARP 1834 for an F/FA of digital systems.

In a purely qualitative analysis, the analyst identifies every significant failure imaginable at a certain subsystem level and then describes the effects that result as the impact of the failure ripples up to the system level. A

simple qualitative FMEA is shown in table 5.4-3. A more detailed worksheet is provided in MIL-STD-1629.

TABLE 5.4-3. FMEA QUALITATIVE ANALYSIS REPORT

| Component | Failure Mode | Failure Effects |
|---|---|---|
| Bus Line Driver | Open circuit | 1. LRU can no longer transmit.<br>2. Bus impedance is changed. |
| | Short Circuit | 1. Bus transmission disabled until driver timeout.<br>2. Bus impedance is changed. |

Once the effects are identified, a quantitative analysis can be performed to find the likelihood of system failure based on the combined contributions of the various unrelated failures. Table 5.4-4 shows a typical quantitative FMEA report. The probability of a critical system failure is $2 \times 10^{-4}$, which is the sum of the four critical effect probabilities.

TABLE 5.4-4. FMEA QUANTITATIVE ANALYSIS REPORT
(Vesely et al. 1981)

| Component | Failure Probability | Failure Mode | Percent Failure by Mode | Critical Effect Probability | Noncritical Effect |
|---|---|---|---|---|---|
| A | $1 \times 10^{-3}$ | Open | 90 | | x |
| | | Short | 5 | $5 \times 10^{-5}$ | |
| | | Other | 5 | $5 \times 10^{-5}$ | |
| B | $1 \times 10^{-3}$ | Open | 90 | | x |
| | | Short | 5 | $5 \times 10^{-5}$ | |
| | | Other | 5 | $5 \times 10^{-5}$ | |

FMEA is generally used to provide an analysis of hardware, but MIL-STD-1629 defines both a strict hardware approach and a functional approach in its procedures for performing FMEA on hardware. Extending FMEA further, VanBaal (1985) found that no special treatment is required when the software elements of a system are included in the analysis. He concluded, "an FMEA of a system containing software can be performed and yields useful results with regard to system safety" (VanBaal 1985). However, the quality of the software has to be ensured by following good software engineering practices. Thus, FMEA can be

used to address data bus integration issues associated with processor-based bus interfaces.

A quantitative FMEA requires that failure rate data be available for all the components. This is not usually available for software or for new and novel hardware components or subsystems. Warr (1984) describes a special method of FMEA that could be applied in these situations. A multifunctional team of design engineers defines the relative rankings and relative weights for each product and its failure modes. The relative risk associated with each failure mode can be calculated from these weights. This method might prove to be especially useful for certificating new and novel bus-integrated systems for which no earlier counterpart exists. Hecht (1986) also addresses some of the unique requirements for applying FMEA to digital avionics.

FMEA is called out as the recommended means of analysis by one of the bus standards. An ARINC 629 bus is to be made of a single, unspliced cable. If a splice is made, the standard recommends that an FMEA be completed for each splice.

5.4.4.7 Failure Mode, Effects, and Criticality Analysis

FMECA is recommended by ARP 1834 for use in F/FA of digital systems. The technique is defined by the U.S. Department of Defense in MIL-STD-1629. The purpose of an FMECA is the early identification of all critical failure possibilities so that they can be eliminated or minimized in the system design. The standard establishes the following procedures:

> "to systematically evaluate the potential impact of each functional or hardware failure on mission success... Each potential failure is ranked by the severity of its effect in order that appropriate corrective actions may be taken to eliminate or control the high risk items." (MIL-STD-1629A, 1984).

FMECA is very similar to an FMEA, but the criticality of the failure is analyzed in greater detail and controls are described for limiting the likelihood of each failure. An FMECA worksheet might look like that shown in table 5.4-5. MIL-STD-1629 contains a more detailed worksheet.

TABLE 5.4-5. FMECA ANALYSIS REPORT

| Failure Mode | Failure Effects | Control | Net Effects |
|---|---|---|---|
| Bus line driver open circuits during transmission | LRU cannot transmit | LRU switches to redundant bus | One transmission is lost |

The standard defines a two-step process, beginning with an FMEA and followed by a more detailed Criticality Analysis (CA). The purpose of the CA is to rank

each potential failure mode according to the combined influence of its severity and likelihood, i.e., according to risk.

The CA can be a qualitative categorization of the failures into five probability categories, or a quantitative calculation based on the hardware component failure rate data in MIL-HDBK-217. MIL-STD-1629 contains a detailed worksheet that is used for a quantitative analysis. Concerning the use of MIL-HDBK-217, Hecht and Hecht (1985) write,

> "While the overall failure rate of an LRU can be computed fairly readily from the part failure rate information, the failure probability in a specific mode pertinent to the aircraft level depends almost entirely on judgement. Thus, a considerable subjective component enters into this approach as well."

FMECA requires a team approach, since the analyst who understands the effects at the component level will probably not properly assess the criticality of the effects at the system level.

5.4.4.8 Fault Insertion

Fault Insertion is suggested by ARP 1834 as a special technique for F/FA of digital systems. Because the system response to a failure may be time, mode, or data dependent, the analytical prediction of the response to a specified failure may be nearly impossible via the basic methods previously described (ARP 1834, 1986). Important considerations for fault insertion, taken from ARP 1834, are summarized below.

Rather than trying to deduce the response of a system to each failure, fault insertion consists of purposely inserting faults to observe the effects. This is most effective when it can be done in the actual system. Rather than alter a standard part, usually an LRU or a BC is emulated in a software based tester in which faults are easily generated. For verifying designs, a computer can be used to simulate bus components in a tester that includes fault generation.

Fault insertion into the actual system is the most realistic, but has the disadvantage that only simple faults can be generated easily. Faults internal to an IC cannot be generated at all. Furthermore, the F/FA cannot be performed until the system is built. On the other hand, faults can be generated at any point in an emulation or a simulation; but they provide less realism. In addition, a simulation allows F/FA to be performed on a system design, long before any part of the system is fabricated. The main disadvantage of emulation and simulation is that they must be validated against the system they claim to reflect. For additional discussions on this topic, see chapters 3 (Curd 1989) and 5 (Cooley 1989) of this handbook.

5.4.4.9 System Safety Assessment

System Safety Assessment (SSA) is a systematic and analytical methodology for assessing the safety of software controlled digital avionic systems. It is formulated for meeting the analysis requirements for civil aircraft airworthiness regulations. The methodology is summarized in table 5.4-6.

TABLE 5.4-6. SYSTEM SAFETY ANALYSIS METHODOLOGY
(VanBaal 1985)

1. Prepare a safety plan:

   - Goal
   - Function safety plan in SSA
   - Limits of the system and the SSA
   - Techniques and analysis methods
   - Safety criteria
   - Time-schedule, organization

2. Prepare a system description:

   - System components (hardware and software)
   - Functions of the system
   - Architecture
   - Interfaces (other systems, crew, environment)
   - Requirements
   - Safety-related measures already foreseen

3. Perform a hazard analysis:

   - A qualitative, top-down analysis of deductive character

4. Perform a failure mode and effect analysis:

   - A bottom-up analysis of inductive character; initially, only of a qualitative nature

5. Perform other analyses, where necessary. Some options are as follows:

   - Zonal analysis
   - Fault tree analysis
   - Sneak circuit analysis
   - Common cause failure analysis
   - Change analysis

The analysis begins with a Hazard Analysis (HA), which identifies the functions whose failure could lead to dangerous situations. The emphasis is on the effects that system failure has on things other than the system, like the airplane or the crew. VanBaal shows the HA worksheet; it is reproduced here in figure 5.4-3. An FMEA is then performed on the parts of the system that contribute to the functions identified by the HA. Additional analyses can be conducted as needed to demonstrate airworthiness. The options are listed in table 5.4-6.

| HAZARD ANALYSIS | Primary System: | Aircraft: | Date: | Page: of |
|---|---|---|---|---|

| # | Function, Failure Condition (Effects on Aircraft and Flight) | Possible Causes | Most Critical Flight Phase | Hazard Class | Hazard Limited By | Hazard Increased By | Remarks |
|---|---|---|---|---|---|---|---|

FIGURE 5.4-3. HAZARD ANALYSIS WORKSHEET HEADER
(VanBaal 1985)

5.4.4.10 Preliminary Hazard Analysis

The Fault Tree Handbook (Vesely et al. 1981) describes a Preliminary Hazard Analysis that is very similar to the HA. The "preliminary" emphasizes that this analysis should be conducted as early in the development cycle as possible to identify system safety requirements. Whereas the other methods focus on the effect of failures on system operation, this procedure assesses the potential hazards posed to system users and bystanders. The process consists of identifying hazardous situations and the events that could place the system in that situation. The likelihood of the enabling events must be assessed so that the need for preventative measures can be determined. This establishes the system safety requirements.

5.4.4.11 Sneak Circuit Analysis

The Computer Resources Handbook for Flight Critical Systems (Hecht and Hecht 1985) describes the sneak circuit analysis referred to in the SSA. It is a systematic way of detecting unintentional behavior in a system. A logic tree is developed for the logic of the system, regardless of whether the system is implemented in hardware or software. A software tool analyzes this tree to find all the conditions that can cause a given output, all conditions that are necessary to prevent a given output, and all conditions that can cause a combination of outputs. An analyst then examines these lists to ensure that there are no violations of the system requirements.

5.4.4.12 Petri Net Safety Analysis

Petri nets are a special form of state diagram that can be used to model and analyze system behavior, both hardware and software. From a Petri net, an analyst can identify all possible states of the system and, particularly, the terminal states into which the logic may lock up. Leveson and Stolzy (1987) give several references for this type of conventional Petri net analysis.

Leveson and Stolzy, however, address the use of Time Petri net modeling and analysis techniques in the safety analysis of real-time computer systems, like those in aircraft. They developed a Petri net variation which includes a time element, noting that "basically correct software actions which are too early or too late can lead to unsafe conditions" (Leveson and Stolzy 1987). From the analysis, they can determine "the timing constraints of the final system necessary to avoid high-risk states and the watch-dog timers needed to detect

critical timing failures" (Leveson and Stolzy 1987). They also develop a procedure for analyzing how failures affect the timing and reachability of the system states. This technique is appropriate for analyzing the complex state transitions of the bus communications in integrated avionics systems. It can be applied early in the design stage.

5.4.4.13 Testing Techniques

After all the design analysis, modeling, prototyping, and simulation, the real product finally needs to be tested. For transport aircraft, manufacturers use a Systems Integration Laboratory to test the data bus integrated avionics, since testing time on the prototype aircraft is much too expensive. This laboratory is also called a "hot-bench." The hot-bench allows the integrated system to be tested, but with simulated aircraft inputs. Simple bus bench testers only simulate generic bus communications. For GA aircraft, this system integration is often done in the actual airplane.

As all the pieces of the aircraft are manufactured, testing may be done on an "iron-bird" configuration. In this test configuration, the avionics are connected to a cockpit mockup that can be "flown" on the ground by test pilots. This provides a realistic, real-time environment in which to test the bus integrated avionics.

The final testing technique used is the flight test. The prototype airplane is actually flown in increasingly more demanding flight patterns to verify proper operation. While no bus integration problems are explicitly tested during this phase, it is still a very important part of the bus integration techniques. The real-time data that is collected is used to validate all of the previous simulations that were performed. If the small set of documented real flight behavior matches that of the simulations of the same behavior, then the simulations of all other unverified activity can be relied upon. For additional discussion on this topic, refer to chapters 8 and 9 of the "Handbook - Volume I" (Hitt 1983).

5.4.5 FAA Certification and Bus Integration

FAR Part 21 defines the general process that avionics manufacturers must follow for the FAA to certify that avionic systems meet the airworthiness standards and are in safe condition for flight. In other words, FAR Part 21 defines the requirements for the process, and Parts 23, 25, 27, 29, and 33 define the requirements for the product. To what extent do the certification process and airworthiness standards ensure that complex bus-integrated avionics are correctly integrated? Consider the four certification processes presented in section 3.

5.4.5.1 Type Certification and Bus Integration

Type certification requires the most thorough demonstration of compliance to the standards. One of the strengths of type certification is it requires that systems meet a standard of what constitutes airworthiness. The manufacturer who develops systems under a TC must (Part 21, subpart B) perform the following steps:

- Submit a plan for the development, production, and verification of the product.

- "Make all inspections and tests necessary to determine compliance with the applicable airworthiness ... requirements" and to determine that the materials, parts, and processes conform to those specified in the type design.

- Perform flight tests, as required by the FAA, to determine the reliability and proper functioning of the system to be approved.

- Submit "the type design, test reports, and computations necessary to show that the product to be certificated meets the applicable airworthiness ... requirements."

- Allow FAA investigators to make any inspections, ground tests, and flight tests necessary for them to determine compliance with the requirements of the FARs.

- Submit a statement of conformity certifying that each product manufactured under the TC conforms to the type design.

A TCed product cannot escape this process.

Since the TC process assures that products are checked for compliance with the airworthiness standards, the strength of the type certification depends on the quality of the airworthiness standards. All four standards include the same general requirements for equipment in aircraft (Subpart F, section 1301):

> "Each item of installed equipment must -
> (a) Be of a kind and design appropriate to its intended function;
> (b) Be labeled as to its identification, function, or operating limitations, or any applicable combination of these factors;
> (c) Be installed according to limitations specified for that equipment; and
> (d) Function properly when installed."

This general requirement is comprehensive in requiring proper functioning. From design and installation to operation, it requires that a bus-integrated system be properly integrated. However, it makes no attempt to define proper functioning and specifies no form of assurance or demonstration that a product meets this requirement.

What constitutes proper functioning is more specifically defined in section 1309 of each airworthiness standard. To varying degrees, proper functioning means that the equipment "perform its intended function under any foreseeable operating condition" and, generally, hazards that could result from probable malfunction or failure must be minimized or prevented. Parts 23, 25, and 29 specifically state that this latter requirement take into account the relation of systems to one another. This makes these regulations strong on integration. But Part 27, for normal category rotorcraft, does not state this. However, the

primary weakness of Part 27 lies in the requirements for demonstration and analysis.

Part 27, the airworthiness standard for normal category rotorcraft, does not require any specific use of analysis, inspections, or tests beyond that required to show compliance with the requirements. This leaves a lot of room for interpretation when establishing whether a bus-integrated avionic system meets the general requirement that it function properly when installed. It certainly does not point the developer in the direction of using integration techniques or following bus standards and guidelines. A stronger section 1309 is in order for more critical systems (Swihart 1984). However, even if section 1309 referred directly to the bus standards, these standards are weak on integration issues. Thus, successful integration of bus communications is not assured for normal category rotorcraft by the current type certification and bus integration documents.

Section 1309 of Parts 23, 25, and 29 specifies analysis and testing for the purpose of demonstrating compliance with the requirements for the probability of failure and for environmental conditions. They also specifically list that an analysis must consider possible failure modes, multiple failures, failure effects, and fault detection. These requirements do much to ensure that integration concerns will be addressed and that integration techniques will be employed. For transport category aircraft, these activities are assured, since Parts 25 and 29 both require such analysis. For normal category airplanes, the analysis is only suggested as an acceptable means of showing compliance. This is a weaker regulation; but in practice, developers usually follow the suggested means for showing compliance. Although these airworthiness standards are more specific about the use of analysis and testing than Part 27, the integration of bus communications still is not ensured, since the bus standards give insufficient guidance on the integration issues.

5.4.5.2 Supplemental Type Certification and Bus Integration

Supplemental type certification involves the second most thorough regulation of the manufacture and installation of avionic systems. The basic tenet of this certification process is that the redesigned system must satisfy all the same requirements as the TCed product into which it will be installed. Thus, similar to the TC process, the manufacturer of a system that is to receive an STC must perform the following steps (Part 21, subpart E):

- "Make all inspections and tests necessary to determine compliance with the applicable airworthiness ... requirements" and to determine that the materials, parts, and processes conform to those specified in the type design.

- Allow FAA investigators to make any inspections, ground tests, and flight tests necessary for them to determine compliance with the requirements of the FARs.

- Submit a statement of conformity certifying that each product manufactured under the STC conforms to the type design.

Although an STCed product cannot escape this process, the STC process does not go as far in checking compliance to the airworthiness standards as did the TC checking. In particular, the manufacturer is not required to repeat the flight tests. Since an applicant for an STC is not the manufacturer who holds the TC for the original product, relaxing the requirements is unjustified. One manufacturer who is changing another's design must be sure to understand the design at least as well as the original design team. Changing a design after the fact is a more demanding process than working out a new design. Despite this, the process tends to move the focus from requiring the applicant to substantiate the entire design to substantiating the compliance of just the redesigned system, where it is assumed that the rest of the product is not affected.

Like the TC process, the STC process ensures that products are checked for compliance with the airworthiness standards. The strength of the supplemental type certification also depends on the quality of the airworthiness standards, as previously discussed. But the fundamental weakness of the STC process for bus-integrated avionics is that it seems to underestimate the ramifications of a second party altering a TCed product.

#### 5.4.5.3 Parts Manufacturer Approval and Bus Integration

A PMA gives a manufacturer approval to produce aircraft parts for sale or installation on the basis that they conform to another manufacturer's TC or STC. This is a reasonable regulation, however, it allows an owner or operator of an aircraft to manufacture parts for use on their aircraft without obligation to this regulation of aircraft part manufacture. In this case, installation approval (FAA Form 337) is all that is required. Although an installation approval requires that the part being installed is the one required by design, no investigation is required to verify the design. This seems to be a major weakness in the process of ensuring safe aircraft. An operating airline may manufacture a bus LRU according to the type design and then be given permission to install it, without any requirement that anyone perform inspections and tests. This is not an acceptable regulation of the production of complex digital bus-integrated avionics. Detailed accountability is a necessity.

When the parts are to be sold, the manufacturer's ability to build reliable parts is carefully examined. The regulations require that the manufacturer who desires to sell a part must perform the following steps (Part 21, subpart K):

- Identify the "product on which the part is to be installed."
- Submit the information necessary to show the design of the part.
- Submit the "test reports and computations necessary to show that the design of the part meets the airworthiness requirements" or that the design is identical to the original TCed part.
- "Establish and maintain a fabrication inspection system."
- Allow FAA investigators to make any inspections or tests necessary for them to determine compliance with the FARs.

- "Make all inspections and tests necessary to determine compliance with the applicable airworthiness requirements" and to determine that the materials, parts, and processes conform to those specified in the type design.

- "Determine that each completed part conforms to the design data and is safe for installation."

These requirements are commensurate with those required to ensure that bus-integrated avionics are properly produced. They are almost as strict as those for type certification. The first point avoids the problem that LRUs in digital systems are not 100 percent interchangeable, since the single intended point of installation must be specified. In general, these requirements also do not assume that the new design produces the same result as the type design. Thus, the manufacturer's design is reviewed and compared to the airworthiness standards, independent of the fact that the new design was to meet the type design.

The PMA is weak because no flight tests are required, even though the part is a new design; no specific inspections and tests are suggested; and the airworthiness requirements can be avoided by showing the new part is identical in design to the original. If the airworthiness requirements are used as the standard, the weaknesses of the PMA are limited primarily to those associated with the airworthiness standards, as previously discussed (except for the lack of a required flight test). But, if the manufacturer chooses to only show that the design of the part is identical to the type design, a more serious problem is allowed to occur for bus-integrated avionics. An identical design would likely be limited to the requirements set forth by the bus standard. Since the bus standards generally do not sufficiently cover bus integration, too much leeway is allowed in determining whether a design is equivalent.

5.4.5.4 Technical Standard Order Authorization and Bus Integration

A TSO Authorization gives a manufacturer approval to produce an aircraft part for sale or installation on the basis that it conforms to the minimum performance standard for the part, as specified by a TSO. A manufacturer who desires to produce parts under a TSO Authorization must perform the following steps (Part 21, subpart O):

- Issue a statement of conformance to the FAR and the TSO.

- Submit the technical data required by the TSO.

- Submit a description of the quality control system.

- "Conduct all required tests and inspections and establish and maintain a quality control system."

- Allow FAA investigators to witness tests and inspect parts, processes, facilities, and files.

This process is the weakest of the four. The TSO concept, that a part can be specified apart from its application, is contradictory to the design of parts that compose bus-integrated avionics. Furthermore, the bus standards do not provide a sufficient forum for producing a reliable TSO. In general, they do not provide an integration standard. Even apart from the integration issues, the bus standards leave too many requirements unspecified. A TSO that incorporates something like the Interface Control Documents published for a MIL-STD-1553 bus system could possibly provide the necessary integration criteria.

The TSO Authorization is weak on checking that the manufacturer's design truly satisfies the minimum performance standard of the TSO. All that is required is a statement of conformance. The FAA investigators may request analysis, inspections, and tests, but none are specified as a matter of course. It is unreasonable to expect the new design to fulfill the requirements without a systematic engineering approach. As pointed out earlier, most of the bus standards are open to interpretation. There is no guarantee that two designs for the same bus-integrated avionics would produce the same result.

The airworthiness issue is also inadequately addressed. It is left unaddressed by the TSO Authorization. The manufacturer need not be concerned with the airworthiness issue or the flight tests needed to substantiate it. The parts need only conform to a TSO. It is left to the installation process (FAA Form 337) to determine whether a part with a TSO number may be installed. Yet,

> "when a system is installed in an aircraft it's often the first time TSO approved components can be tested for proper integration. This point in the certification process is often too late for the type of testing which would be required to demonstrate all combinations of system operation." (Williams 1989).

Nevertheless, as pointed out before, the installation process does not investigate the design of the part and certainly not its impact on airworthiness.

The FAA has responded to this problem by issuing ACs on some of the advanced integrated systems, like autopilot, TCAS, and multisensor navigation systems. The ACs require an additional approval, the Preliminary Installation Approval, under certain circumstances. This approval is achieved through an STC program for the integration of the system into a specific aircraft. The manufacturer must obtain the STC before producing the system for sale. The AC further specifies the conditions under which the purchaser must either undergo an STC evaluation or simply apply for an installation approval.

Another problem with the TSO Authorization is that manufacturers with a TSO Authorization are given substantial freedom for changing the design. The manufacturer may make minor changes to the design being used without any further approval from the FAA. This could be reasonable, depending on what is considered a minor change. However, it is left to the manufacturer to make the determination. The manufacturer can of course solicit an unofficial opinion from the ACO. If too much latitude is taken, the FAA will discover this in the next audit of the manufacturer's activity. In this case, the manufacturer risks the possibility of being fined by the FAA.

The manufacturer can also request a deviation from the requirements of the TSO. To be granted a deviation, the manufacturer must show that the deviation is compensated for by factors or design features that provide an equivalent level of safety. The means of showing compliance to this level of safety is unspecified.

### 5.4.6 Summary

Until bus standards are standardized in addressing the complete development process, it is not sufficient for a developer to claim that a developed bus satisfies the bus standard. Even when bus standards and guidelines are followed, the extent to which reliable communication has been achieved depends on the particular bus documentation. Some standards barely address the integration problems.

Furthermore, bus standards will never be able to ensure that any particular design is proper. The standards must, necessarily, leave room for application specific variations. Thus, CEs should expect that bus communications be developed and validated through a methodology which includes most of the following techniques (Ashmore 1982, Bannister et al. 1982, Carter 1986, Earhart 1982, Hitt and Eldredge 1983, Shimmin 1989, Spradlin 1983, VanBaal 1985, and Verdi 1980):

- Requirements Capture - Use a system that ensures that each requirement is captured by the design. A cross-reference matrix to identify where each system requirement is satisfied in the system specification should result.
- Configuration Control - Use a system that tracks exactly what revision of which components constitute the latest approved configuration.
- Design Modeling - Model the design for the purpose of choosing the best one for the system specification.
- Hazards Analysis - Determine the effect of system failures.
- F/FA - Determine the probability of each failure occurrence.
- Hot Bench Simulation - Perform laboratory testing of LRUs based on simulation of the environment.
- Iron Bird Simulation - Perform testing of real-time operation on a simulation that uses as much as possible of the actual avionics and airframe.
- Flight Testing - Perform testing during actual flight of the aircraft.

If levels of performance are set for specific techniques, such a systematic development can ensure proper operation for any application. Certification procedures and airworthiness standards need to go further to ensure that bus integration is accomplished according to a systematic engineering process involving analytical techniques.

## 6. CONCLUSIONS

The data communication on serial data buses in aircraft has been analyzed from several different angles. The certification procedures have been reviewed to determine the certification activity that is performed on data buses. The regulations have been analyzed to identify the stipulations that avionic data buses must satisfy. The technology has been addressed from several points of view to determine the areas that require careful attention, regulation, and/or certification. What are the main areas of concern for using data buses in aircraft? To what extent are these areas addressed by the current regulations and certification procedures? What improvements are needed? The answers to these questions are summarized in this section.

### 6.1 Certification Procedures for Bus-Integrated Systems

The conclusion of the technical report research on certification procedures is repeated here (Elwell et al. 1992).

The TC and STC processes are sufficiently robust to accommodate the complexities of current and emerging serial data bus technologies. However, they do not currently address these technologies as specifically as necessary. The procedures leave the manufacturer and ACO free to mutually determine the specific analysis, tests, and documentation required to substantiate the safety of the aircraft being designed. They are even free to categorically accept data buses as safe, treating them as "simply a piece of wire." Data buses, however, are more than just wire and have failure modes that cannot be exercised by system level tests. Communications on bidirectional data buses are sufficiently complex that the methods of demonstration should be carefully thought out, documented, and standardized by data bus experts. A standard that functions like RTCA/DO-178 does for software, is needed for Type Certification of bus-integrated systems.

The Production Certificate situation is similar. The procedure appropriately requires the manufacturer to establish an approved production inspection and test system to ensure that each manufactured part meets the type design. However, the procedure is not specific about what inspections and tests to use. This is inadequate, since manufacturers implement various amounts of testing. For example, Earhart (1991) explains that, although MIL-STD-1553 buses have been designed and implemented for nearly 20 years, testing of LRUs is often insufficient because of fundamental misconceptions, such as the following:

- "Validation testing is not necessary if validated components are used to build the RT."
- "Because the [bus] interface board was validated in one LRU, validation testing isn't necessary on subsequent LRUs."

- "Validation testing is not necessary because the LRU has been operating in the system."

A formal bus testing standard should be adopted by the industry for each avionic bus to ensure that tested systems truly are reliable. Furthermore, the reliability of integrated systems is not ensured by tests of system components. Installation approvals need to include integration testing.

The procedure specified for a TSO Authorization inadequately addresses approval of bus-integrated systems, since it allows approval of a component independent of the system into which it will be installed. The FAA is making interim provisions through special ACs. For the long term, either the procedure should be formally made more robust, or integrated systems should not be eligible for this approval.

The current certification procedures have successfully supported the certification of nonessential systems using data buses. They have also been used to support the certification of bus-based essential systems backed up by conventional means. Most of this experience is with unidirectional buses, but some of it involves bidirectional buses used to a limited extent. To date, no civilian aircraft accidents are known to be due to data bus failure.

As bidirectional data buses are used for essential and critical systems and relied upon in fly-by-wire aircraft, the procedures need to specifically identify the steps necessary for ensuring reliable data bus operation.

## 6.2 Related Regulations and Standards

There is no specific approach for certificating systems containing digital data buses and integrated avionic equipment. As a result, the CE must consult many sources for information concerning any affects a data bus might have on a flight-critical or flight-essential system. To further complicate the problem, the sources must be related to broad federal regulations.

The AIAA, IEEE, and other organizations produce publications which address technical requirements for avionic systems integrated with data buses. Since current certification methods do not consider systems at the bus's level, these publications could be useful for establishing new certification procedures. Test procedures within the publications may ensure that particular interactions between a system and a data bus are not overlooked during a system's certification.

RTCA and SAE committees work towards making an avionics system easier to certify. Standards produced by these organizations may be used as part of the manufacturer's design process, or as informal guidelines to meet specific FARs, ACs, or SCs.

ARINC and GAMA also work with manufacturers to produce standards for avionic equipment and data buses. The standards include specifications of data bus topologies and protocols, as well as tests which data bus manufacturers can use during the development process. Because the standards contain specific

technical information about data buses, they are also useful for system certification.

Section 4 related current FARs to procedures defined by the above organizations. It is also useful for developing a certification process applicable to integrated avionic equipment and data buses. The new procedure for certification should consider information from a variety of sources and treat every integrated system separately. Whether a new certification process is developed, or current methods refined, a successful procedure should perform a thorough V&V on all aspects of flight-critical and flight-essential systems.

## 6.3 Bus-Integrated Systems Technology

### 6.3.1 System Integration Concerns

There is no one factor that can satisfy all of the requirements for data buses used in flight-critical systems. The accurate and timely delivery of data from the source to the destination demands that the data bus exhibit a mixture of attributes. The architecture needs to control data latency. Physical redundancy needs to be carefully considered and implemented. Protocols must ensure periodic, deterministic, simple, error-free, and efficient communication. Other attributes, such as maintenance and monitoring, are implemented at the discretion of the designers and system integrators. This implementation requires careful and detailed consideration in the design phase and should not be treated as an afterthought.

The system designer or system integrator is tasked with many integration decisions. Using standards that are not completely specified, or are unclear in certain areas, creates problems which might escape detection. Seeking to resolve any ambiguity at an early stage will ensure a more successful integration period.

The system integrator should not merely be concerned with having an operational system, but a system that operates correctly under all conditions. Exhaustive monitoring, recording, and reporting of data bus activity is the only way to ensure that data bus integrity is maintained.

### 6.3.2 Bus Hardware-Software Interaction

Hardware-software interaction between a BIU and an avionic system can be easily overlooked during a system's certification process. Integration has taken data buses to a new level, sometimes concealing what functions are implemented with hardware and what functions are implemented with software. Section 5.2 discussed hardware-software interaction between digital data buses and avionic systems. Special attention was given to the BIU IC and the host system CPU's interface.

To analyze hardware-software interaction at this level, section 5.2 discussed data integrity problems that arise when a bus and avionic system pass data. Common errors which occur during bus hardware-software interaction include parity, framing, and overrun errors. Other errors more specific to certain buses, are timing and interrupt handling errors. All of these errors could

result from dynamic conditions within the BIU or CPU, or from the system's design.

Regardless of the type, any undetected error can have a catastrophic effect on its system. For this reason, section 5.2 presented methods of error detection and correction. These methods included monitoring, voting, retransmission for after the fact correction, prevention, and redundancy. Any of the methods can be applied at the hardware-software interface.

Practical solutions for hardware-software interaction problems were presented in the final part of section 5.2. Although some of the solutions were designed by manufacturers of military aircraft, they are applicable to civilian aircraft as well. Detection and correction methods are a key part of the solutions. Section 5.2 demonstrated how some of the solutions are implemented.

### 6.3.3 Bus Protocol Specification and Analysis

One particular area that requires careful attention from the designer is the communication protocol. Since protocol specifications must be both concise and easy to understand, formal techniques are used for protocol specification. Formal techniques should be used to model and define protocols and to analyze the correctness and proper operation of the protocol.

With a shift from unidirectional to bidirectional data buses, the access protocol assumes an added degree of complexity. Since protocols may be implemented in hardware rather than software, the protocol should be subjected to rigorous scrutiny before it becomes "buried" in the hardware. The protocol specification and analysis should be performed as carefully as it is for a software-implemented protocol.

### 6.3.4 Bus Integration Standards, Guidelines, and Techniques

The integration of LRUs from various sources to form a system implies that there exists a central specification to which each manufacturer can design. The digital data buses provide this specification. The buses used to integrate the various LRUs on the market are designed according to a published industry standard for each bus. However, these standards primarily specify the operation of a single bus interface, rather than an entire integrated system. The system integration is mostly left in the hands of the system designer. As a result, when these systems are certificated, there is no standard by which to judge the applicant's claim that the system meets the airworthiness standards.

The airworthiness standards require analysis and testing, but bus integration standards need to be formally published by the industry to provide direction on these topics. These standards should specify the values and tolerances of all constant parameters.

Bus integration guidelines should be published to control the flexible aspects of a bus. These guidelines should provide formulas or formal procedures for deriving reliable values for variable parameters. In addition, these standards need to specify component and system tests designed to exercise the full scope

of significant failure modes. Some of the bus standards specify component tests, but none address system tests.

The standards, guidelines, and test procedures that have been developed for the MIL-STD-1553 bus come close to providing a model for bus standards and integration guidelines. The standard for individual LRUs is flexible, yet specific. Numerous integration guidelines are provided. The documents for civilian avionic buses would be greatly improved if their specifications and procedures were patterned after those of the military bus; and would be complete if integration standards were added.

Techniques for systematic, analytical design and analysis also need to be formally incorporated into the certification requirements. Techniques exist which can improve the reliability of bus-based digital systems, but they are not presently part of the bus standards or the certification procedures. These should be formally recommended in these documents for the development of bus-based systems.

### 6.3.5 FAA Certification and Bus Integration

The buses associated with integrated avionics currently in use have been certificated implicitly as part of the system that uses them. Thus, they are naturally certificated in an integrated environment. This has been sufficient for unidirectional and bidirectional buses used in nonessential systems. As bidirectional buses are fully used in essential and critical systems, the bus integration issues will need to be explicitly analyzed and tested.

The airworthiness standard for normal category rotorcraft, as specified in FAR Part 27, is particularly weak on bus integration issues since it does not require any specific analysis and testing, as do the other airworthiness standards. Although the other standards (Parts 23, 25, and 29) do require specific analysis, the industry provides very few guidelines on how to perform the bus analysis and testing required. Thus, even though the certification requirements are fairly strong for these categories of aircraft, there is no guarantee that the implementation is meaningful. The TC and STC procedures, which rely on these airworthiness standards, need to specify standards and guidelines that may be followed as an acceptable means for showing compliance to the airworthiness standards. This may be done either by reference or inclusion.

The STC procedure is weakened even further on integration concerns, since it allows a manufacturer to make a change to another's design without resubstantiating the entire design. Only the changed aspects need to be shown to meet the airworthiness standard. This would be a reasonable allowance to make for the manufacturer who designed it, but not for another. Another manufacturer is more likely to overlook the full ramifications of a change than would the original designer. This situation needs to be closely examined to see if the development of a bus-integrated system is adequately examined under an STC approval.

The PMA covers the production of bus-integrated systems fairly well, but it relies heavily on the completeness of design specifications for ensuring that

manufactured parts are reliable. Currently, the bus standards are too ambiguous for that to be a safe approach. Integration analysis, ground tests, and/or flight tests should be considered for every installation of a bus-integrated system. Alternatively, bus integration standards could be developed that are sufficiently specific that successful integration of a manufactured part could be more easily ensured.

The TSO Authorization regulations currently do not address integration at all. The regulations assume that systems developed under a TSO Authorization are interchangeable. This is not realistic for complex bus architectures and protocols. The FAA is taking steps to rectify this situation.

To sufficiently address the integration of systems with data buses, the industry needs to develop a comprehensive suite of documentation on each data bus. This documentation needs to include a thorough standard that includes test procedures and criteria for single LRUs and an integration standard that includes system specifications, tests, and guidelines. In addition, the FAA regulations need to adopt these standards as specifying an acceptable means for showing compliance with the airworthiness requirements for safe operation.

## 6.4 Summary

Improvements needed to support the certification of flight-essential and flight-critical systems that use data buses have been identified, as follows:

- The certification procedures of FAR Part 21 need to consistently require that integrated systems, rather than components, be certificated.
- The airworthiness standards of FAR Parts 23, 25, 27, 29, and 33 need to consistently require analysis and testing of equipment, systems, and installations, particularly in an integrated configuration.
- ACs should be published to establish formal guidelines for the specification, design, analysis, and testing of bus protocols and hardware.
- Bus standards need to be adopted by ACs as informal guidelines, specifying an acceptable means for showing compliance to the FARs.
- Bus standards need to specify a complete system engineering methodology for the specification, design, analysis, and testing of bus protocols and hardware, from the component to the system level.
- Specific analysis techniques need to be adopted by ACs as informal guidelines for bus analysis.

## APPENDIX A - DYNAMIC TIME SLOT ALLOCATION PROTOCOL

The operation of the Dynamic Time Slot Allocation (DTSA) protocol is based on time allocations being preassigned to all bus users. Each user is guaranteed that during its time slot, under error-free conditions, it will have an opportunity to access the bus. This access method lends itself to a high bus efficiency, even under heavy loading conditions. A simple state diagram for the DTSA protocol is given in figure A-1.

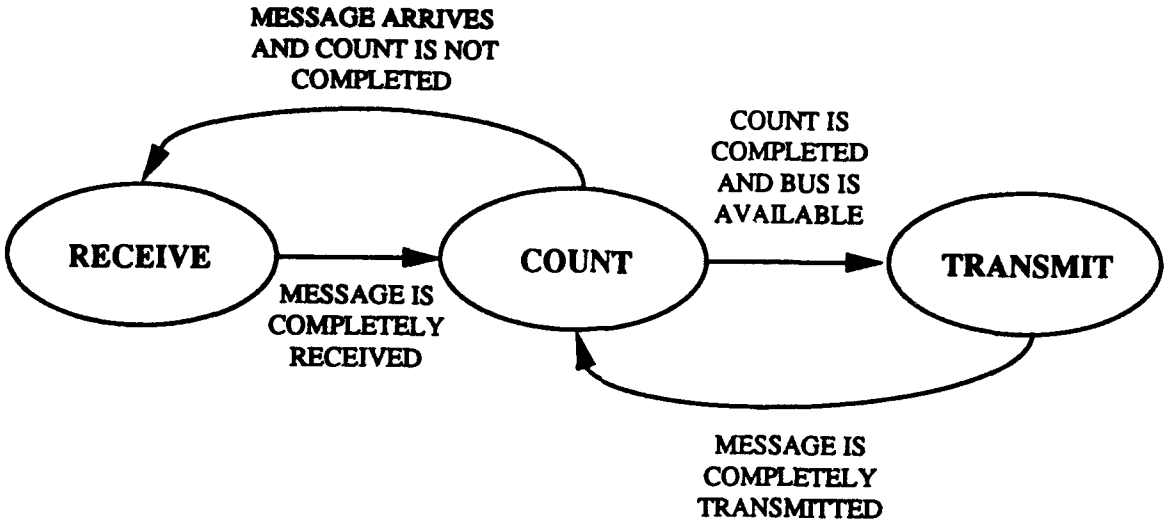



FIGURE A-1. DTSA ACCESS PROTOCOL STATE DIAGRAM

Under normal operation, one bus user will be in the transmit state and all other bus users will be in the receive state. After the transmitting user is finished, it and all the receiving users go into a count state to determine which can access the bus next. The amount of time each user must wait to send a message is determined by the following relationship (Porter, Couch, and Schelin 1983):

$$T_m = \begin{cases} T_c(n-m) & \text{if } n>m \\ T_c(N+n-m) & \text{if } n\leq m \end{cases}$$

where

$T_m$ - wait time for user m
$T_c$ - count duration (based on maximum propagation delay)
n - address of user performing computation
m - original address of last transmission
N - maximum number of users

As seen in the state diagram, each user fluctuates between the "count" state and "receive" state, or the "count" state and the "transmit" state. When a transmission is received from the bus, the user switches from the "count" to the "receive" state and then back to the "count" state when reception is complete. When a new transmission is received, each user decrements its counter. If user number four is in the count mode and then receives a transmission from user number two, user number four computes the time until it can originate a transmission, $T_c = n-m$, or two units of time. The sequence for four users is given in table A-1.

TABLE A-1. DTSA USER ACCESS SEQUENCE
(Porter, Couch, and Schelin 1983)

| Address of Terminal Performing Calculation (n) | Address of Last Terminal in Transmit Mode (m) | Number of $T_c$ Times Terminal Must Wait in Count Mode Before Transmitting<br>$n - m$ if $n > m$<br>$N + n - m$ if $n \leq m$ |
|---|---|---|
| 1 | 1 | 4 |
| 2 | 1 | 1 |
| 3 | 1 | 2 |
| 4 | 1 | 3 |
| 1 | 2 | 3 |
| 2 | 2 | 4 |
| 3 | 2 | 1 |
| 4 | 2 | 2 |
| 1 | 3 | 2 |
| 2 | 3 | 3 |
| 3 | 3 | 4 |
| 4 | 3 | 1 |
| 1 | 4 | 1 |
| 2 | 4 | 2 |
| 3 | 4 | 3 |
| 4 | 4 | 4 |
| SEQUENCE REPEATS | | |

If a user does not have data to send when its count duration decrements to zero, the bus interface simply sends a status message without involving the host central processing unit. This maintains a constant timing and access sequence for the protocol.

The timing diagram for different numbers of active users is given in figure A-2. Note that the frame time, $T_F$, which is defined as the time from the start of user number one time slot to the next start of user number one time slot, varies based on the number of active users and the length of messages being sent on the bus.

As seen by examination of the DTSA, time slot protocols exhibit greater throughput and shorter wait times during periods of heavy loading than does a Carrier-Sense, Multiple Access protocol. Also, for cases where the average access times of both protocols are approximately the same, DTSA provides the shortest maximum wait time (Porter, Couch, and Schelin 1983).

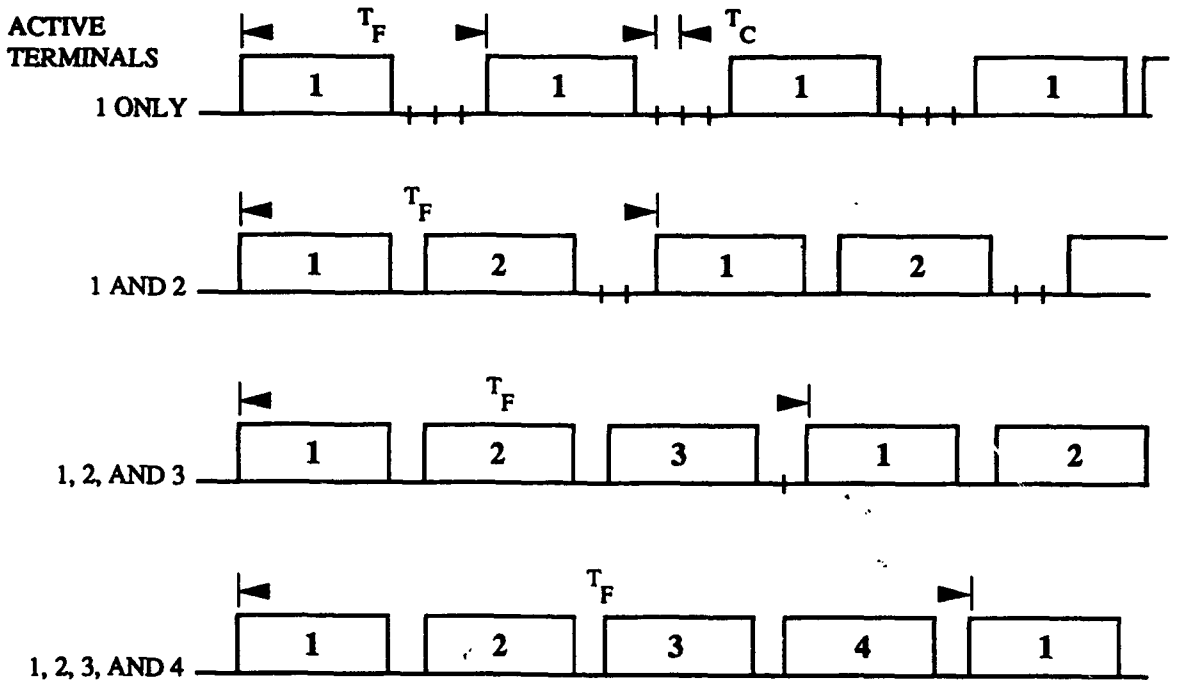



FIGURE A-2. DTSA ACCESS PROTOCOL USER TIMING

## APPENDIX B - HIGH-LEVEL DATA LINK CONTROL PROTOCOL

The High-Level Data Link Control (HDLC) protocol was defined by the International Standards Organization for the purpose of replacing character-oriented protocols. It is a bit-oriented protocol which may be broken into three main categories for better understanding:

- The bit stream
- Transmission format
- Station-to-Station cooperation

### B.1 The Bit Stream

Since HDLC is a bit-oriented protocol, data at the physical layer of the Open Systems Interconnect Basic Reference Model appears simply as part of the bit stream. This bit stream includes information which may be added by a higher layer (e.g., network or transport layer). It is then necessary to define the beginning and end of the data bit stream. This is done by using a flag at the beginning and end of the sequence. The entire bit stream is referred to as a frame.

The bit sequence that defines an HDLC frame is 0 1 1 1 1 1 1 0. This unique bit sequence appears only at the beginning and end of the frame. When data are sent using the HDLC protocol, the transmitter will test for the occurrence of consecutive ones. When five ones in a row are found, the transmitter will automatically insert a zero for the next bit. At the receiver, if there is a bit stream of five ones detected, the sixth bit is dropped.

### B.2 Transmission Format

Any information sent using the HDLC protocol uses the format shown in figure 5.1-6 of this chapter. Normally, the address and control fields are each eight bits in length. The address field may contain the address of the sender or receiver, depending on the particular configuration. A broadcast mode is implemented by using all ones in this field. Groups of users, or stations, may be assigned a particular address to which they are to respond, called group addresses. Extended addressing may be used by setting the last bit in this field to a zero. In this case, the address field can be extended by multiples of eight bits.

There are three kinds of frames defined for HDLC: Information frames (I frames), Supervisory frames (S frames), and Unnumbered frames (U frames). The formats of these frames are given in table B-1. I frames are used in data transfer to maintain a sequential flow of related information; S frames are used for data control, to acknowledge or reject messages from the sender; and U

frames are used for control purposes. They are used to implement initialization, disconnection, polling, and other functions (Tanenbaum 1981).

TABLE B-1. HDLC CONTROL FRAMES
(Meijer and Peeters 1982)

| Frame Type | MSB Bit Significance LSB | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
| I | Receive Count N(R) | | | P/F | Send Count N(S) | | | 0 |
| S | Receive Count N(R) | | | P/F | Supervisory Type S | | 0 | 1 |
| U | Modifier M1 | | | P/F | Modifier M2 | | 1 | 1 |

In table B-1, N(R) and N(S) refer to the number of I frames which are received or sent, respectively. All stations maintain counters for these variables. They are used to keep the frames in proper sequence. The sender increases the N(S) bit field by one for each I frame it sends, and the receiver increases the N(R) field by one for each I frame it acknowledges. These three-bit fields allow for only seven unacknowledged frames (Meijer and Peeters 1982).

When a sender polls another bus user, the sender sets this Poll bit. The receiver replies with a response frame and sets the F, or Final, bit. The S bit field indicates different types of supervisory frames. M1 and M2 are modifier bits for the Unnumbered frames. These bits are used to define the various control commands used by the HDLC protocol (Tanenbaum 1981).

The Frame Check Sequence field in figure 5.1-6 of this chapter is a method for checking the validity of the received frame. It is actually a Cyclic Redundancy Check (CRC) inserted in the message by the sender based on the generator polynomial $X^{16} + X^{12} + X^5 + 1$. If a CRC error is detected by the receiver, the entire frame is discarded and some form of error recovery should be exercised.

Combined stations can send both command frames and response frames. The difference between the two types of frames will be in the address bit field. If the address is the station's own, then the frame is a response frame; otherwise it is a command frame. In unbalanced operation, a frame sent by a primary station is always a command frame, and that sent by a secondary station is always a response frame.

## APPENDIX C - CHECKLIST FOR ANALYSIS OF DATA BUS HARDWARE AND SOFTWARE

Avionic manufacturers who wish to evaluate their systems can use the checklist provided in table C-1. The checklist should not be the only means used to evaluate these systems. It is merely a starting point for ensuring that single failures are adequately addressed. The checklist could be used in conjunction with a Failure Mode and Effects Analysis or other method described in section 5.4 of this chapter.

TABLE C-1. CHECKLIST FOR ANALYSIS OF BUS HARDWARE AND SOFTWARE (Bunce 1980)

| SYSTEM _____ | FAILURE MODE _____ |
|---|---|
| Is the failure detected by the system, LRU, CPU, or BIU? | YES [ ] NO [ ] |
| Does the CPU's software detect this failure? | YES [ ] NO [ ] |
| Does the BIU's hardware annunciate this failure to the CPU's software? | YES [ ] NO [ ] |
| Does the CPU's softwar provide effective methods for dealing with this failure? | YES [ ] NO [ ] |
| If the CPU's software cannot correct the error, will other hardware within the BIU or LRU? | YES [ ] NO [ ] |
| Will the failure cause either HW or SW to overload the other? | YES [ ] NO [ ] |
| If the failure mode is introduced into other SW logic, will other functions be affected? | YES [ ] NO [ ] |
| Is the system able to handle more than one of these failures at a time? | YES [ ] NO [ ] |
| Is reconfiguration of the system, by either the system itself or the crew, necessary? | YES [ ] NO [ ] |
| Explanations/Comments: | |
| Necessary Changes: | |

## BIBLIOGRAPHY


"Addendum to the MIL-STD-1553 Multiplex Application Handbook," Air Force Systems Command, Wright-Patterson Air Force Base, OH, March 1983.

Advisory Circular No. 21-16C, "Radio Technical Commission for Aeronautics Document No. DO-160C," U.S. Department of Transportation, Federal Aviation Administration, February 14, 1990.

Advisory Circular No. 21-303.1A, "Certification Procedures for Products and Parts," U.S. Department of Transportation, Federal Aviation Administration, August 10, 1972.

Advisory Circular No. 23-1309.1, "Equipment, Systems and Installations in Part 23 Airplanes," U.S. Department of Transportation, Federal Aviation Administration, September 19, 1989.

Advisory Circular No. 25-1309.1A, "System Design and Analysis," U.S. Department of Transportation, Federal Aviation Administration, June 21, 1988.

Advisory Circular No. XX-XX, "Certification of Aircraft Electrical/Electronic Systems for Operation in the High Intensity Radiated Fields (HIRF) Environment," U.S. Department of Transportation, Federal Aviation Administration, Draft 14, December 10, 1991.

AE-12, MIL-STD-1553 Databus Systems Integration Handbook, Society of Automotive Engineers, Warrendale, PA, 1991.

AGARD Conference Proceedings, "Design for Tactical Avionics Maintainability," AGARD-CP-361, Advisory Group for Aerospace Research and Development, Neuilly Sur Seine, France, October 1984.

AGARD Conference Proceedings, "Fault Tolerant Design Concepts for Highly Integrated Flight Critical Guidance and Control Systems," AGARD-CP-456, Advisory Group for Aerospace Research and Development, Neuilly Sur Seine, France, April 1990.

AGARD Conference Proceedings, "Tactical Airborne Distributed Computing and Networks," AGARD-CP-303, Advisory Group for Aerospace Research and Development, Neuilly Sur Seine, France, October 1981.

AGARD Lecture Series, "Computing Systems Configuration for Highly Integrated Guidance and Control Systems," AGARD-LS-158, Advisory Group for Aerospace Research and Development, Neuilly Sur Seine, France, June 1988.

AGARD Lecture Series, "Systems Engineering," AGARD-LS-164, Advisory Group for Aerospace Research and Development, Neuilly Sur Seine, France, May 1989.

AIR 1189, Airborne Internal Interface Standards for Moderate Bit Rate Digital Time Division-Multiplex Systems, Society of Automotive Engineers, Warrendale, PA, March 1972.

AIR 1207, A Primer of Aircraft Multiplexing, Society of Automotive Engineers, Warrendale, PA, January 1972.

AIR 4271, Handbook of System Data Communications, Society of Automotive Engineers, Warrendale, PA, November 1, 1989.

AIR 4288, Linear Token Passing Multiplex Data Bus User's Handbook, Draft 4 of Issue 1, Society of Automotive Engineers, Warrendale, PA, April 1991.

AIR 4289, Handbook for the SAE AS4074.2 High Speed Ring Bus, Issue 5, Society of Automotive Engineers, Warrendale, PA, April 1990.

AIR 4291, Test and Validation Plan for the SAE AS4074.2 High Speed Ring Bus Interface Module, Issue 1, Draft 3, Society of Automotive Engineers, Warrendale, PA, April 1991.

American National Standard for Information Systems, "Fiber Distributed Data Interface (FDDI) - Token Ring Media Access Control (MAC)," ANSI X3.139-1987, American National Standards Institute, New York, NY, 1987.

American National Standard for Microprocessor Bus Structures, "IEEE Standard for Multiplexed High-Performance Bus Structure: VSB," ANSI/IEEE 1096-1988, Institute of Electrical and Electronics Engineers, New York, NY, 1989.

ANSI/ISA-RP55.1, "Hardware Testing of Digital Process Computers," Instrument Society of America, Research Triangle Park, NC, May 5, 1983.

ARD 50004, Open System Interconnection (OSI) and Other Reference Models User Perspectives for Real Time Applications, Society of Automotive Engineers, Warrendale, PA, September 25, 1989.

"ARINC 429 Bus Interface Line Driver Circuit," HS-3182 Data Sheet, Harris Semiconductor, Melbourne, FL.

"ARINC 629 Communication Integrated Circuit," National Semiconductor Corporation, Sunnyvale, CA, October 24, 1990.

"ARINC 629 Symposium View Foils," Boeing Commercial Airplane Company, Seattle, WA, March 23, 1991.

"ARINC 629 User's Manual," Boeing Commercial Airplane Company, Seattle, WA, July 26, 1990.

ARINC Project Paper 617, "Guidance for Avionic Certification and Configuration Control," Draft 4, Aeronautical Radio, Inc., Annapolis, MD, December 12, 1990.

ARINC Project Paper 629, "Multi-Transmitter Data Bus, Part 2 - Applications Guide," Draft 1, Aeronautical Radio, Inc., Annapolis, MD, January 26, 1989.

ARINC Project Paper 629, "Multi-Transmitter Data Bus, Part 3 - Data Standards," Draft 4, Aeronautical Radio, Inc., Annapolis, MD, March 10, 1989.

ARINC Project Paper 651, "Design Guidance for Integrated Modular Avionics," Draft 5, Aeronautical Radio, Inc., Annapolis, MD, August 1, 1990.

ARINC Specification 429-12, "Mark 33 Digital Information Transfer System (DITS)," Aeronautical Radio, Inc., Annapolis, MD, July 1, 1990.

ARINC Specification 600-7, "Air Transport Avionics Equipment Interfaces," Aeronautical Radio, Inc., Annapolis, MD, January 1987.

ARINC Specification 607, "Design Guidance for Avionic Equipment," Aeronautical Radio, Inc., Annapolis, MD, February 17, 1986.

ARINC Specification 607, "Design Guidance for Avionic Equipment," Supplement 1, Aeronautical Radio, Inc., Annapolis, MD, July 22, 1987.

ARINC Specification 617, "Guidance for Avionic Certification and Configuration Control," Draft 4, Aeronautical Radio, Inc., Annapolis, MD, December 12, 1990.

ARINC Specification 629, "Multi-Transmitter Data Bus, Part 1 - Technical Description," Aeronautical Radio, Inc., Annapolis, MD, March 7, 1990.

ARINC Specification 629, "Multi-Transmitter Data Bus, Part 1 - Technical Description," Draft 1 of Supplement 2, Aeronautical Radio, Inc., Annapolis, MD, October 24, 1990.

ARINC Specification 629, "Multi-Transmitter Data Bus, Part 1 - Technical Description," Draft 3 of Supplement 1, Aeronautical Radio, Inc., Annapolis, MD, January 21, 1991.

ARINC Strawman Material for Project Paper 629, "Multi-Transmitter Data Bus, Part 4 - Test Plan," Draft, Aeronautical Radio, Inc., Annapolis, MD, September 14, 1990.

ARINC Strawman Material for Project Paper 659, "Backplane Data Bus," Draft, Aeronautical Radio, Inc., Annapolis, MD, January 29, 1991.

ARP 926A, Fault/Failure Analysis Procedure, Society of Automotive Engineers, Warrendale, PA, September 1985.

ARP 1834, Fault/Failure Analysis for Digital Systems and Equipment, Society of Automotive Engineers, Warrendale, PA, August 7, 1986.

AS4074.1, Linear Token Passing Multiplex Data Bus, Society of Automotive Engineers, Warrendale, PA, September 7, 1988.

AS4074.2, High Speed Ring Bus, Society of Automotive Engineers, Warrendale, PA, August 29, 1988.

AS4113, Validation Test Plan for the Digital Time Division Command/Response Multiplex Data Bus Controllers, Society of Automotive Engineers, Warrendale, PA, January 11, 1989.

AS4115, Test Plan for the Digital Time Division Command/Response Multiplex Data Bus System, Society of Automotive Engineers, Warrendale, PA, January 11, 1989.

AS4290, Test and Validation Plan for the AS4074.1 Linear Token Passing Multiplex Data Bus, Draft 8 of Issue 1, Society of Automotive Engineers, Warrendale, PA, April 8, 1991.

Ashmore, A. S., "Certification of Digital Systems for Civil Aircraft," Certification of Avionic Systems: Proceedings of the Symposium, Royal Aeronautical Society, London, United Kingdom, April 1982.

"Avionics Market Data," Avionics, July 1991.

Bailey, John, "Honeywell Joins 777," Flight International, December 1990.

Bakken, David E., "Inter-Partition Data Integrity in the Asynchronous DATAC Environment," Proceedings of the AIAA/IEEE 8th Digital Avionics Systems Conference, American Institute of Aeronautics and Astronautics, Washington, DC, 1988.

Bannister, J. A. et al., "Problems Related to the Integration of Fault-Tolerant Aircraft Electronic Systems," NASA-CR-165926, NASA Langley Research Center, Hampton, VA, June 1982.

Bavuso, Salvatore J. et al., "Applications of the Hybrid Automated Reliability Predictor," NASA-TP-2760, NASA Langley Research Center, Hampton, VA, December 1987.

Becher, Bernice, "Diagnostic Emulation: Implementation and User's Guide," NASA-CR-178391, NASA Langley Research Center, Hampton, VA, December 1987.

Benson, J. W., "Hardware Fault Insertion and Instrumentation System: Mechanization and Validation," DOT/FAA/CT-86/31, U.S. Department of Transportation, Federal Aviation Administration, March 1987.

Bermingham, W. J., E. A. Alfonsi, and W. A. Rosen, "A High Speed Fiber Optic Data Bus for Avionics Applications," IEEE 1987 National Aerospace and Electronics Conference (NAECON), Institute of Electrical and Electronics Engineers, New York, NY, 1987.

Bochmann, G. and C. Sunshine, "Formal Methods in Communication Protocol Design," IEEE Transactions on Communications, Volume COM-28, No. 4, Institute of Electrical and Electronics Engineers, New York, NY, April 1980.

Bondy, Jon and Richard Weaver, "Bus Schedule Change Issues in the On-Board Distributed Data System (ODDS)," Proceedings of the IEEE/AIAA 5th Digital Avionics Systems Conference, Institute of Electrical and Electronics Engineers, New York, NY, 1983.

Bunce, W. L., "Hardware and Software: An Analytical Approach," Proceedings of the Annual Reliability and Maintainability Symposium, Institute of Electrical and Electronics Engineers, New York, NY, 1980.

Card, M. Ace, "Evolution of the Digital Avionic Bus," Proceedings of the IEEE/AIAA 5th Digital Avionics Systems Conference, Institute of Electrical and Electronics Engineers, New York, NY, 1983.

Carter, W. C., "System Validation - Putting the Pieces Together," Proceedings of the IEEE/AIAA 7th Digital Avionics Systems Conference, Institute of Electrical and Electronics Engineers, New York, NY, 1986.

Clarke, Clifton A. and William E. Larsen, "Aircraft Electromagnetic Compatibility," DOT/FAA/CT-86/40, U.S. Department of Transportation, Federal Aviation Administration, June 1987.

Clifton, Daniel B., "Using the HS-3282 ARINC Bus Interface Circuit," Application Note No. 400, Harris Semiconductor, Melbourne, FL.

"CMOS ARINC Bus Interface Circuit," HS-3282-8 Data Sheet, Harris Semiconductor, Melbourne, FL, November 1989.

Cohn, Marc D., "A Proposed Local Area Network for Next-Generation Avionic Systems," Proceedings of the IEEE 1988 National Aerospace and Electronics Conference (NAECON), Institute of Electrical and Electronics Engineers, New York, NY, 1988.

Cooley, William W., "Advanced Fault Insertion and Simulation Methods," Digital Systems Validation Handbook, Volume II, Chapter 5, DOT/FAA/CT-88/10, U.S. Department of Transportation, Federal Aviation Administration, February 1989.

Curd, Hardy P., "Integrated Assurance Assessment," Digital Systems Validation Handbook, Volume II, Chapter 3, DOT/FAA/CT-88/10, U.S. Department of Transportation, Federal Aviation Administration, February 1989.

Daley, E., "An Update on Experience on the Fly by Wire Jaguar Equipped with a Full-Time Digital Flight Control System," AGARD Conference Proceedings - Active Control Systems - Review, Evaluation and Projections, AGARD-CP-384, Advisory Group for Aerospace Research and Development, Neuilly Sur Seine, France, March 1985.

Danthine, A. A. S., "Petri Nets for Protocol Modelling and Verification," Proceedings of the Computer Networks and Teleprocessing Symposium, October 1977.

Danthine, A. A. S., "Protocol Representation with Finite-State Models," IEEE Transactions on Communications, Volume COM-28, No. 4, Institute of Electrical and Electronics Engineers, New York, NY, April 1980.

Dasaro, Joseph A., "The Impact of Future Avionics Technology on the Conduct of Air Warfare," AGARD Conference Proceedings - Improvement of Combat Performance for Existing and Future Aircraft, AGARD-CP-409, Advisory Group for Aerospace Research and Development, Neuilly Sur Seine, France, December 1986.

"DATAC Current Mode Coupler," AMP/Dallas Semiconductor, January 21, 1991.

Dekker, G. J., "Reliability Aspects of Software for Digital Avionics," National Aerospace Laboratory, Amsterdam, Holland, 1985.

DelCoco, Robert J., Brian W. Kroeger, and John J. Kurtz, "A Comparison of the ANSI FDDI and SAE HSRB Token Ring LANs," FOC/LAN '87 and MFOC-WEST Proceedings: The Eleventh Annual International Fiber Optic Communications and Local Area Networks Exposition, Information Gatekeepers, Boston, MA, October 1987.

Denery, D. G. et al., "A Demonstration Advanced Avionics System for General Aviation," Business Aircraft Meeting and Exposition, SAE 790569, Society of Automotive Engineers, Warrendale, PA, 1979.

"Designing for ARINC 429," Avionics, March 1991.

Driscoll, Kevin, "Multi-MicroProcessor Flight Control System," Proceedings of the IEEE/AIAA 5th Digital Avionics Systems Conference, Institute of Electrical and Electronic Engineers, New York, NY, 1983.

Earhart, Leroy, "MIL-STD-1553: Testing and Test Equipment," Proceedings of the 2nd AFSC Standardization Conference, Air Force Systems Command, Wright-Patterson Air Force Base, OH, November 1982.

Earhart, Leroy, "Testing MIL-STD-1553," Avionics, March 1991.

Eldredge, Donald and Ellis F. Hitt, "Digital System Bus Integrity," DOT/FAA/CT-86/44, U.S. Department of Transportation, Federal Aviation Administration, March 1987.

Eldredge, Donald and Susan Mangold, "Digital Data Buses for Aviation Applications," Digital Systems Validation Handbook, Volume II, Chapter 6, DOT/FAA/CT-88/10, U.S. Department of Transportation, Federal Aviation Administration, February 1989.

Elwell, Donald A. et al., Avionic Data Bus Integration Technology, DOT/FAA/CT91/19, Department of Transportation, Federal Aviation Administration, 1992.

Evans, Daniel B., "Fault Tolerant High-Speed Switched Data Network," Proceedings of the IEEE/AIAA 7th Digital Avionics Systems Conference, Institute of Electrical and Electronics Engineers, New York, NY, 1986.

FAA Order 8110.4, "Type Certification," U.S. Department of Transportation, Federal Aviation Administration, June 1985.

Federal Aviation Regulation, Part 21, "Certification Procedures for Products and Parts," October 25, 1989.

Federal Aviation Regulation, Part 23, "Airworthiness Standards: Normal, Utility, Acrobatic, and Commuter Category Airplanes," February 4, 1991.

Federal Aviation Regulation, Part 25, "Airworthiness Standards: Transport Category Airplanes," September 10, 1990.

Federal Aviation Regulation, Part 27, "Airworthiness Standards: Normal Category Rotorcraft," October 22, 1990.

Federal Aviation Regulation, Part 29, "Airworthiness Standards: Transport Category Rotorcraft," October 22, 1990.

Federal Aviation Regulation, Part 33, "Aircraft Engines," January 12, 1983.

Fitzgerald, Gary L. and C. F. Polivka, "Design Considerations for a MIL-STD-1553 System Tester," 1982 IEEE International Automatic Testing Conference, 1982.

"Flight Deck Automation: Promises and Realities," NASA-CP-10036, NASA Ames Research Center, Moffett Field, CA, 1989.

GAMA, "ARINC 429 General Aviation Subset," General Aviation Manufacturers Association, Washington, DC, June 16, 1986.

GAMA, "Avionics Standard Communications Bus (ASCB)," General Aviation Manufacturers Association, Washington, DC, October 15, 1987.

GAMA, "Commercial Standard Digital Bus (CSDB)," General Aviation Manufacturers Association, Washington, DC, June 10, 1986.

Galli, E., "Test Philosophy of the EH101 Integrated Avionic," AGARD Conference Proceedings - The Design, Development and Testing of Complex Avionics Systems, AGARD-CP-417, Advisory Group for Aerospace Research and Development, Neuilly Sur Seine, France, December 1987.

Garbo, Dennis L., "Multiplex Data Bus Simulator," Proceedings of the IEEE/AIAA 5th Digital Avionics Systems Conference, Institute of Electrical and Electronics Engineers, New York, NY, 1983.

"Guide to Federal Aviation Administration Publications," FAA-APA-PG-12, U.S. Department of Transportation, Federal Aviation Administration, May 1990.

Hassenpflug, W. and M. Baumker, "Navigation Systems for the New Generation of Combat and Transport Helicopters and Associated Flight Tests," AGARD Conference Proceedings - Advances in Guidance and Control Systems and Technology, AGARD-CP-411, Advisory Group for Aerospace Research and Development, Neuilly Sur Seine, France, July 1987.

Hatzidakis, Fokion, "Multichannel Data Transmission through a Fiber Optic Cable," AD-A193 842, Naval Post Graduate School, Monterey, CA, December 1987.

Hecht, Herbert, "Problems with Failure Modes and Effects Analysis for Digital Avionics," Proceedings of IEEE/AIAA 7th Digital Avionics Systems Conference, Institute of Electrical and Electronics Engineers, New York, NY, 1986.

Hecht, Herbert and Myron Hecht, Computer Resources Handbook for Flight Critical Systems, ASD-TR-85-5020, Air Force Systems Command, Wright-Patterson Air Force Base, OH, January 1985.

Hecht, Myron, "Fault Tolerant Software," Digital Systems Validation Handbook, Volume II, Chapter 9, DOT/FAA/CT-88/10, U.S. Department of Transportation, Federal Aviation Administration, February 1989.

Held, G., "Data Communications Networking Devices," John Wiley and Sons, New York, NY, 1989.

Henley, Gordon D. and Thomas F. Fiorino, "Avionics/Navigation Architectural Design Considerations," The National Telesystems Conference, Institute of Electrical and Electronics Engineers, New York, NY, 1982.

Highland, David L., "CAML: Digital Avionics in a Real-Time Application," The National Telesystems Conference, Institute of Electrical and Electronics Engineers, New York, NY, 1982.

Hitt, E. et al., Handbook - Volume I, Validation of Digital Systems in Avionics and Flight Control Applications, DOT/FAA/CT-82/115, U.S. Department of Transportation, Federal Aviation Administration, July 1983.

Hitt, Ellis F., "Digital Avionics Design for Validation," Proceedings of the 2nd AFSC Standardization Conference, Air Force Systems Command, Wright-Patterson Air Force Base, OH, November 1982.

Hitt, Ellis F., "Real-Time Fault Tolerant Software in Distributed Avionics Systems Architectures Using Digital Data Buses," Proceedings of the IEEE/AIAA 7th Digital Avionics Systems Conference, Institute of Electrical and Electronics Engineers, New York, NY, 1986.

Hitt, Ellis F. and Don Eldredge, "A Review and Application of Analytical Models in Fault Tolerant Avionics Validation," Proceedings of the IEEE/AIAA 5th Digital Avionics Systems Conference, Institute of Electrical and Electronics Engineers, New York, NY, 1983.

Holmes, David C. E., "Global System Data Bus Using the Digital Autonomous Terminal Access Communication Protocol," Proceedings of the IEEE/AIAA 7th Digital Avionics Systems Conference, Institute of Electrical and Electronics Engineers, New York, NY, 1986.

Hooper, Dean and James Leidy, "New Concept in Data Highway Technology," IEEE 1981 IECI Proceedings, Institute of Electrical and Electronics Engineers, New York, NY, November 1981.

Hubacek, Phil, "The Advanced Avionics Standard Communications Bus," Business and Commuter Aviation Systems Division, Honeywell, Inc., Phoenix, Arizona, July 10, 1990.

Improving Aircraft Safety, National Academy of Sciences, Washington, DC, 1980.

"Integrated Application of Active Controls (IAAC) Technology to an Advanced Subsonic Transport Project - ACT/Control/Guidance System Study," Volume I, NASA-CR-165963, NASA Langley Research Center, Hampton, VA, December 1982.

ISO-7498, "Information Processing Systems - Open System Interconnection - Basic Reference Model," American National Standards Institute, Inc., New York, NY, 1983.

"Isolation of Faults in Air Force Weapons and Support Systems, Volume I," National Research Council, Washington, DC, April 1986.

"Isolation of Faults in Air Force Weapons and Support Systems, Volume II," National Research Council, Washington, DC, July 1986.

Jennings, Randle G., "Avionics Standard Communications Bus - Its Implementation And Usage," Proceedings of the IEEE/AIAA 7th Digital Avionics Systems Conference, Institute of Electrical and Electronics Engineers, New York, NY, 1986.

Karmarker and Clark, Proceedings of the 1982 Summer Computer Simulation Conference, 1982.

Kushnir, Alex and Yehuda Kasirer, "LAVI 1553B Communication System," Proceedings of the IEEE 1988 National Aerospace and Electronics Conference (NAECON), Institute of Electrical and Electronics Engineers, New York, NY, 1988.

Lala, J. H., "Fault Detection, Isolation, and Reconfiguration in FTMP: Methods and Experimental Results," Proceedings of the IEEE/AIAA 5th Digital Avionics Systems Conference, Institute of Electrical and Electronic Engineers, New York, NY, 1983.

Larsen, William E., "Digital Avionics Susceptibility to High Energy Radio Frequency Fields," Proceedings of the IEEE 1988 National Aerospace and Electronics Conference (NAECON), Institute of Electrical and Electronics Engineers, New York, NY, 1988.

Larsen, William E., "Digital Avionics Systems - Overview of FAA/NASA/Industry-Wide Briefing," Proceedings of the IEEE/AIAA 7th Digital Avionics Systems Conference, Institute of Electrical and Electronics Engineers, New York, NY, 1986.

Leveson, Nancy G. and Janice L. Stolzy, "Safety Analysis Using Petri Nets," IEEE Transactions on Software Engineering, Volume SE-13, No. 3, Institute of Electrical and Electronics Engineers, New York, NY, March 1987.

Lindsey, Rodger A., "General Purpose Bus Interface Unit (GPBIU) System Test Software Design," AFIT/GCS/EE/83D-64, Air Force Institute of Technology, Wright-Patterson Air Force Base, OH, December 1983.

Longenecker, Timothy R., "Structured Methods: Specification of Real-Time Systems," Tutorial of the IEEE/AIAA 7th Digital Avionics Systems Conference, Institute of Electrical and Electronics Engineers, New York, NY, 1986.

Mackall, Dale A., "Development and Flight Test Experiences with a Flight-Crucial Digital Control System," NASA-TP-2857, NASA Dryden Flight Research Facility, Edwards, CA, November 1988.

Ma otto, Tom and Linda Alger, "Advanced Information Processing System: Input/Output System Services," NASA-CR-181874, NASA Langley Research Center, Hampton, VA, August 1989.

McCartney, Richard I. and Randy E. Phillips, "A Modular Approach to MIL-STD-1553 Simulation Support," Proceedings of the IEEE 1981 National Aerospace and Electronics Conference (NAECON), Institute of Electrical and Electronics Engineers, New York, NY, 1981.

McConnell, Roger A., "Avionics System Design for High Energy Fields," DOT/FAA/CT-87/19, U.S. Department of Transportation, Federal Aviation Administration, July 1988.

McGough, John, "Evaluation of Data Busses for Flight Critical Control Applications," Proceedings of IEEE/AIAA 7th Digital Avionics Systems Conference, Institute of Electrical and Electronics Engineers, New York, NY, 1986.

McManus, James C., "Logistics Engineering Analysis Techniques for Fault-Tolerant Avionics Systems," Air Force Human Resources Laboratory, Brooks Air Force Base, TX, November 1985.

McSharry, Michael E., "Fault Tolerant Flight Control Avionics Integration Using MIL-STD-1553B," Proceedings of the IEEE/AIAA 5th Digital Avionics Systems Conference, Institute of Electrical and Electronics Engineers, New York, NY, 1983.

Mehler, Leo, "Military Aircraft System Engineering," SAE 861690, SAE Transactions, Volume 95, Society of Automotive Engineers, Warrendale, PA, 1987.

Meijer, Anton and Paul Peeters, Computer Network Architectures, Computer Science Press, Rockville, MD, 1982.

Mejzak, Richard S., "New Technology Impact on Future Avionics Architectures," AGARD Conference Proceedings - Advanced Computer Aids in the Planning and Execution of Air Warfare and Ground Strike Operations, AGARD-CP-404, Advisory Group for Aerospace Research and Development, Neuilly Sur Seine, France, February 1987.

Merlin, Philip M., "Specification and Validation of Protocols," IEEE Transactions on Communications, Volume COM-27, No. 11, Institute of Electrical and Electronics Engineers, New York, NY, November 1979.

Meyer, John W., "SAE AE-9B Draft Standard High Speed Token Passing Data Bus for Avionics Applications," Proceedings of the IEEE/AIAA 7th Digital Avionics Systems Conference, Institute of Electrical and Electronics Engineers, New York, NY, 1986.

"Microcommunications," INTEL Corporation, Order No. 231658-006, Mt. Prospect, IL, 1990.

MIL-E-6051, "Electromagnetic Capability Requirements," February 26, 1988.

MIL-HDBK-217E, "Reliability Prediction of Electronic Equipment," Notice 1 Version, January 2, 1990.

MIL-HDBK-1553A, "Multiplex Applications Handbook," November 1, 1988.

"MIL-STD-1553 Designer's Guide," ILC Data Device Corporation, Bohemia, NY, 1982.

MIL-STD-1553B: Applications, Developments, and Components - Seminar Proceedings, ERA Report No. 86-0237, ERA Technology Ltd., Leatherhead, United Kingdom, 1987.

MIL-STD-1553B, "Digital Time Division Command/Response Multiplex Data Bus," Notice 2 Version, September 8, 1986.

MIL-STD-1629A, "Procedures for Performing a Failure Mode, Effects, and Criticality Analysis," Notice 2 Version, November 24, 1984.

MIL-STD-1773, "Fiber Optics Mechanization of an Aircraft Internal Time Division Command/Response Multiplex Data Bus," May 20, 1983.

Military Avionics Architecture for Today and Tomorrow - Seminar Proceedings, ERA Report No. 88-0437, ERA Technology Ltd., Leatherhead, United Kingdom, 1989.

Mulcare, Dennis, "Specification, Design and Testing of Embedded Software," Tutorial of the IEEE/AIAA 8th Digital Avionics Systems Conference, Institute of Electrical and Electronics Engineers, New York, NY, 1986.

"Multiplex Applications Handbook," Air Force Systems Command, Wright-Patterson Air Force Base, OH, May 1, 1980.

Munoz, Jose L., "Modeling for Architecture Assessment," Tools for the Simulation Profession 1988 - Proceedings of the 1988 Conferences: Tools for the Simulationist and Simulation Sortware, The Society for Computer Simulation, San Diego, CA, 1988.

Nelson, Ronald J., "The Synergistically Integrated Reliability Architecture: A Reliability Analysis of an Ultra-Reliable Fault Tolerant Computer Design," Naval Postgraduate School, Monterey, CA, September 26, 1986.

Nguyen, Viet H. et al., "Space Shuttle Descent Flight Verification by Simulation: A Challenge in Implementing Flight Control System Robustness," AGARD Conference Proceedings - Space Vehicle Flight Mechanics, AGARD-CP-489, Advisory Group for Aerospace Research and Development, Neuilly Sur Seine, France, June 1990.

Notice of Proposed Rulemaking, No. 85-6, Federal Register, Volume 50, No. 31, February 14, 1985, pp. 6186-6188.

Ostgaard, John C. and David A. Zann, "Network Communications for a Distributed Avionics System," AGARD Conference Proceedings - Advanced Concepts for Avionics/Weapon System Design, Development and Integration, AGARD-CP-343, Advisory Group for Aerospace Research and Development, Neuilly Sur Seine, France, October 1983.

Papadopoulos, Gregory M., "Avionic Architectures for Fly-By-Wire Aircraft," The National Telesystems Conference, Institute of Electrical and Electronics Engineers, New York, NY, 1982.

Parhami, Behrooz, "Interconnection Redundancy for Reliability Enhancement in Fault-Tolerant Digital Systems," Digital Processes, Volume 5, No. 3-4, Autumn-Winter 1979.

Petrichenko, Jr., Nick, "Lessons Learned in Building a Hardware in the Loop Simulation," Tools for the Simulation Profession 1988 - Proceedings of the 1988 Conferences: Tools for the Simulationist and Simulation Software, The Society for Computer Simulation, San Diego, CA, 1988.

Plice, William A., "Design for Reliability, Testability, Maintainability," Tutorial of the IEEE/AIAA 7th Digital Avionics Systems Conference, Institute of Electrical and Electronics Engineers, New York, NY, 1986.

Porter, David R., Philip R. Couch, and Jean W. Schelin, "A High Speed Fiber Optic Data Bus for Local Data Communications," Journal on Selected Areas in Communications, Volume SAC-1, No. 3, April 1983.

Proceedings of the IEEE 1985 National Aerospace and Electronics Conference (NAECON), Institute of Electrical and Electronics Engineers, New York, NY, 1985.

Proceedings of the IEEE 1986 National Aerospace and Electronics Conference (NAECON), Institute of Electrical and Electronics Engineers, New York, NY, 1986.

Rich, B. A. et al., "Multibus Avionic Architecture Design Study (MAADS)," AFWAL-TR-83-1141, Air Force Systems Command, Wright-Patterson Air Force Base, OH, October 1983.

RS-422-A, "Electrical Characteristics of Balanced Voltage Digital Interface Circuits," Electronic Industries Association, Washington, DC, December 1978.

RTCA/DO-160C, "Environmental Conditions and Test Procedures for Airborne Equipment," Radio Technical Commission for Aeronautics, Washington, DC, December 1989.

RTCA/DO-178A, "Software Considerations in Airborne Systems and Equipment Certification," Radio Technical Commission for Aeronautics, Washington, DC, March 1985.

Runo, Steven C., "Gulfstream IV Flight Management System," Proceedings of the 1990 AIAA/FAA Joint Symposium on General Aviation Systems, DOT/FAA/CT-90/11, U.S. Department of Transportation, Federal Aviation Administration, May 1990.

Sawtell, R. M. and K. P. Dawson, "Avionic Systems Integration Using the BETA Box," GEC Review, Volume 4, No. 2, 1988.

"Serial Interface Module (SIM) for ARINC 629/DATAC," AMP/Dallas Semiconductor, December 1990.

Shapiro, Albert J., "The Design, Simulation and Developmental Testing of the Space Shuttle Data Bus System," AGARD Conference Proceedings - Guidance and Control Techniques for Advanced Space Vehicles, AGARD-CP-350, Advisory Group for Aerospace Research and Development, Neuilly Sur Seine, France, January 1984.

Shaw, John L., Hans K. Herzog, and Kenji Okubo, "Digital Autonomous Terminal Access Communication (DATAC)," Proceedings of the IEEE/AIAA 7th Digital Avionics Systems Conference, Institute of Electrical and Electronics Engineers, New York, NY, 1986.

Shaw, John L. and Peter L. Sutcliffe, "ARINC 629 Data Bus System," Civil Avionics - The Future International Scene: Proceedings of the Symposium, Royal Aeronautical Society, London, United Kingdom, March 1988.

Shimmin, J. A., "The Certification of the Avionic Systems on the ATP to JAR 25," European Forum: The Evolution of Regional Aircraft Technologies and Certification, DGLR-Bericht-89-02, German Society for Aeronautics and Astronautics, Bonn, Federal Republic of Germany, 1989.

Singh, Avtar and Walter A. Triebel, The 8088 Microprocessor: Programming, Interfacing, Software, Hardware, and Applications," Prentice Hall, Inc., Englewood Cliffs, NJ, 1989.

Smith, Lawrence R., "Digital Bus Standardization for Business Aviation," SAE 830757, Society of Automotive Engineers, Warrendale, PA, 1983.

Snelling, K. S., "Certification Experience of the Jaguar Fly-By-Wire Demonstrator Aircraft Integrated Flight Control System," Flight Mechanics and System Design Lessons from Operational Experience, AGARD-CP-347, Advisory Group for Aerospace Research and Development, Neuilly Sur Seine, France, October 1983.

Special Condition No. 23-ACE-49, Federal Register, Volume 55, No. 30, February 13, 1990, pp. 4986-4991.

Special Condition No. 25-ANM-35, Federal Register, Volume 55, No. 213, November 2, 1990, pp. 46191-46196.

Spieth. James E., "Simulation Model of a High-Speed Token-Passing Bus for Avionics Applications," AFIT/GCS/ENG/85D-15, Air Force Institute of Technology, Wright-Patterson Air Force Base, OH, December 1985.

Spieth, James E. and Walter D. Seward, "Simulation Model of a High-Speed Token-Passing Bus for Avionics Applications," Proceedings of the IEEE/AIAA 7th Digital Avionics Systems Conference, Institute of Electrical and Electronics Engineers, New York, NY, 1986.

Spitzer[1], Cary R., "All-Digital Jets Are Taking Off," IEEE Spectrum, Institute of Electrical and Electronics Engineers, New York, NY, September 1986.

Spitzer[2], Cary R., "Digital Avionics Architectures - Design and Assessment," Tutorial of the IEEE/AIAA 7th Digital Avionics Systems Conference, Institute of Electrical and Electronics Engineers, New York, NY, 1986.

Spitzer, Cary R., Digital Avionics Systems, Prentice Hall, Englewood Cliffs, NJ, 1987.

Spitzer, Cary R., "Digital Avionics - The Best is Yet To Come!!!," IEEE Transactions on Aerospace and Electronic Systems, Volume AES-20, No. 4, Institute of Electrical and Electronics Engineers, New York, NY, July 1984.

Spradlin, Richard E., "Flight Management Systems: Where Are We Today and What Have We Learned?," Guidance and Control Conference, American Institute of Aeronautics and Astronautics, Gatlinburg, TN, August 1983.

Sunshine, Carl A., "Formal Techniques for Protocol Specification and Verification," Computer, Volume 12, Institute of Electrical and Electronics Engineers, New York, NY, September 1979.

Swihart, Jr., J. D., "Certification of Advanced Systems," NASA Ames Research Center Technical Workshop: Advanced Helicopter Cockpit Design Concepts, NASA CP 2351, Federal Aviation Administration, Fort Worth, TX, December 1984.

Sylvester and Hung, Proceedings of the 1982 Summer Computer Simulation Conference, 1982.

Tanenbaum, Andrew S., Computer Networks, Prentice Hall, Englewood Cliffs, NJ, 1981.

Thomas, Ronald E., "A Standardized Digital Bus For Business Aviation," Proceedings of the IEEE/AIAA 5th Digital Avionics Systems Conference, Institute of Electrical and Electronics Engineers, New York, NY, 1983.

Thorpe, Duane J. and Kumar V. Vakkalanka, "MIL-STD-1553B Validation Testing," Proceedings of the 2nd AFSC Standardization Conference, Air Force Systems Command, Wright-Patterson Air Force Base, OH, November 1982.

"Tutorial: MIL-STD-1553 Multiplex Data Bus," Proceedings of the 2nd AFSC Standardization Conference, Air Force Systems Command, Wright-Patterson Air Force Base, OH, November 1982.

"User's Manual for AC-XX-XX, High Intensity Radiated Fields (HIRF)," U.S. Department of Transportation, Federal Aviation Administration, Draft, January 15, 1992.

Van Baal, Jozef B. J., "Hardware/Software FMEA Applied to Airplane Safety," 1985 Proceedings of the Annual Reliability and Maintainability Symposium, Institute of Electrical and Electronics Engineers, New York, NY, 1985.

Veatch, Michael H. et al., "Logistics Engineering Analysis Techniques for Fault-Tolerant Avionics Systems," Air Force Systems Command, Wright-Patterson Air Force Base, OH, November 1985.

Verdi, James S., "High Speed Multiplex Bus Protocol Study," NADC-81049-50, Naval Air Development Center, Warminster, PA, June 15, 1980.

Vesely, W. E. et al., Fault Tree Handbook, NUREG-0492, U.S. Nuclear Regulatory Commission, January 1981.

Vorwerk, John, "Chips for ARINC 629," Avionics, March 1991.

Warr, Robert E., "Failure Modes and Effects Analysis Method for New Product Introductions," SAE 841600, Advances in Aerospace Propulsion, SP-594, Society of Automotive Engineers, Warrendale, PA, December 1984.

"WD193X Synchronous Data Link Controller," Western Digital Corporation, Irvine, CA, 1983

"WD1931/WD1933 Compatibility Application Notes," Western Digital Corporation, Irvine, CA, 1983

"WD1993 ARINC 429 Receiver/Transmitter and Multi-Character Receiver/Transmitter," Western Digital Corporation, Irvine, CA, 1983

Williams, James H., "Issues Concerning Certification of Integrated Cockpit Avionics," SAE Conference on General Aviation, Wichita, KS, April 12, 1989.

Zempolich, Bernard A., "Integrated Digital Avionic Systems - Promise and Threats," Astronautics and Aeronautics, Volume 21, October 1983.

## GLOSSARY

ACCESS. The process of a transmitting bus user obtaining control of a data bus in order to transmit a message.

ADVISORY CIRCULAR. An external FAA publication consisting of nonregulatory material of a policy, guidance, and informational nature.

AIR TRANSPORT AIRCRAFT. Aircraft used in interstate, overseas, or foreign air transportation.

AIRWORTHINESS STANDARDS. Parts 23, 25, 27, 29, and 33 of the Code of Federal Regulations, Title 14, Chapter 1, Subchapter C.

ARCHITECTURE. The design and interaction of components of a computer system.

AVIONIC. Electronic equipment used in aircraft.

BABBLING TRANSMITTER. A bus user that transmits outside its allocated time.

BALANCED CONFIGURATION. A bus using the HDLC protocol that connects only primary stations.

BIDIRECTIONAL DATA BUS. A data bus with more than one user capable of transmitting.

BIT-ORIENTED PROTOCOL. A communication protocol where message frames can vary in length, with single bit resolution.

BRIDGE. A BIU that is connected to more than one bus for the purpose of transferring bus messages from one bus to another, where all the buses follow the same protocol.

BROADCAST DATA BUS. A data bus where all messages are transmitted to all bus users.

BUFFER. Memory used to hold segments of the data transferred between asynchronous processes.

BUS. A conductor that serves as a common connection of a signal to multiple users.

BUS CONTROLLER. The electronic unit that is designed to control the bus communication of all users for a centrally controlled bus.

BUS INTERFACE UNIT. The electronics that interface the host CPU of an LRU to a bus medium.

BUS MESSAGE. A complete set of bits that can be transferred between two bus users.

BUS NETWORK. The collection of all BIUs and bus media associated with one bus.

BUS OVERLOAD. The condition that exists when the time it takes to transmit outstanding messages on a bus exceeds the time allotted for those transmissions.

BUS USER. Any LRU attached to a bus.

CENTRAL BUS CONTROL. The bus control approach where a single electronic unit attached to a bus controls all the communication of the bus users.

CERTIFICATION. The process of obtaining FAA approval for the design, manufacture, and/or sale of aircraft and associated systems, subsystems, and parts.

CHARACTER-ORIENTED PROTOCOL. A communication protocol where messages can vary in length, with single character resolution.

CHECKSUM. An error detection code produced by performing a binary addition, without carry, of all the words in a message.

CLOSED-LOOP. A system where the output is a function of the input and the system's previous output.

COMMAND/RESPONSE DATA BUS. A data bus whose protocol initiates each data transfer with a command and terminates the transfer after a proper response is received.

CONFIGURATION MANAGEMENT. The precise control and documentation of the configuration of an entity at any time during its development and deployment.

CONTENTION PROTOCOL. A protocol that allows users to randomly access the bus at any time. When bus contention results, each user tries again to access the bus without contention.

CONTROL REGISTER. A register in an IC controller that receives commands from a host processor.

DATA BUS. A bus that carries electronic signals that represent information.

DATA BUS PROTOCOL. The set of rules that governs the transfer of data between data bus users.

DATA LATENCY. The delay in transferring data from its source to various users. This can result in using an old sample of data in a system after a new sample is available.

DATA REASONABLENESS CHECK. A check performed to see if a value of data is within reasonable bounds for the given context.

DEFAULT DATA. An alternative value used for a parameter whenever the normal data is not supplied.

DETERMINISTIC PROTOCOL. A protocol where all parameters are known so that its various states are predictable in sequence and time.

DIGITAL DATA BUS. A data bus that uses digital electronic signals.

DISSIMILAR REDUNDANCY. The redundancy of systems that provide a redundancy of function, but by a different form.

DISSIMILAR SOFTWARE. Redundant computer programs that provide a redundancy of function, but by a different form.

DISTRIBUTED BUS CONTROL. The bus control approach where the total communication control job is distributed across the bus users, each controlling the communications during its period of responsibility.

EMULATION. The duplication of the behavior of a system with a different system.

ERROR MASKING. The process of masking the presence of avionic errors, possibly by using an electronic voter to override an erroneous input with the values of substitute inputs.

FAIL-SAFE. A design philosophy that ensures that any failure in a system does not result in an unsafe condition after the failure.

FAULT TOLERANCE. The ability of a system to continue operation after a fault, possibly in a degraded condition.

FEDERAL AVIATION REGULATIONS. Subchapter C of the Code of Federal Regulations, Title 14, Chapter 1.

FINITE STATE MACHINE. A state machine with a finite number of states.

FLIGHT-CRITICAL FUNCTION. A function whose failure would contribute to or cause a failure condition that would prevent the continued safe flight and landing of the aircraft.

FLIGHT-ESSENTIAL FUNCTION. A function whose failure would contribute to, or would cause, a hazardous failure condition that would significantly impact the safety of the aircraft or the ability of the flight crew to cope with adverse operating conditions.

FLIGHT-NONESSENTIAL FUNCTION. A function whose failure could not significantly degrade aircraft capability or crew ability.

FRAME. A formatted block of data words or bits that is used to construct messages.

FUNCTIONAL PARTITIONING. The partitioning of system functions by placing each group of users, which share a common function, on different data buses.

GATEWAY. A bus user that is connected to more than one bus for the purpose of transferring bus messages from one bus to another, where the buses do not follow the same protocol.

GENERAL AVIATION AIRCRAFT. The non-air transport civil aircraft.

GENERATOR POLYNOMIAL. The polynomial code that is used to generate the remainder in the division of the CRC check.

GLOBAL STATE. A state that represents the condition of the entire network being modeled, including senders, receivers, and the communication link.

HALF-DUPLEX. Bidirectional communication between two entities on a single channel by each having a turn to control the channel.

HAMMING CODE. An error detection and correction code based on the Hamming distance.

HAMMING DISTANCE. The number of bit positions in which two binary words differ.

HANDSHAKING. The reciprocal responses given by two electronic systems to sequence the steps of a transfer of data between them.

HARDWARE-IN-THE-LOOP SIMULATION. A partial simulation of a system; part of the actual system is used in the simulation.

INTERRUPT VECTOR. The address that points to the beginning of the service routine for an interrupt.

INTERRUPT VECTOR TABLE. The table of interrupt vectors for all interrupts serviced by a system.

LINE REPLACEABLE UNIT. An electronics unit that is made to be replaced on the flight line, as opposed to one that requires the aircraft be taken to the shop for repair.

LINEAR BUS. A bus where users are connected to the medium; one on each end, with the rest connected in between.

MANCHESTER II MODULATION. A non-return to zero, bipolar modulation of a voltage that encodes bits based on the zero-crossing direction of the signal.

MODELING. Creating a system of mathematical equations that formulate all the significant behavior of a system.

MULTIVERSION PROGRAMMING. N-version programming.

N-VERSION PROGRAMMING. The independent coding of a number, N, of redundant computer programs that are run concurrently for the purpose of comparing their outputs.

OVERHEAD. The message timing gaps, control bits, and error detection bits added to some data to satisfy the data bus protocol.

PARITY. An error detection bit added to a data word based on whether the number of "one" bits is even or odd.

PARTITIONED. Colocated hardware or software functions that are designed so that adverse interactions between them cannot occur.

PETRI NET. A state analysis diagram that tracks the status of the state transition conditions of a state machine.

POLLING. A method whereby a CPU monitors the status of a peripheral by periodically reading its status signals.

POLYNOMIAL CODE. A sequence of bits that represents the coefficients of each term in a polynomial.

PRIMARY STATION. An intelligent HDLC protocol user, usually used to manage the access of other bus users to the bus.

PROPAGATION DELAY. The time it takes an electrical signal to travel from its source to its destination.

PROTOCOL. The set of rules by which all bus users must abide to access the bus and ensure its specified operation.

RECONFIGURATION. The process of a system reassigning which hardware performs a particular function.

RECOVERY BLOCK. A block of code executed upon detection of a fault to recover from the erroneous condition that results.

REGISTER. A single word of RAM located within an IC controller that is used for transferring data and control information.

REMOTE TERMINAL. The BIU portion of a MIL-STD-1553 bus user.

RING BUS. A bus where users are connected only to the two adjacent users in a continuous ring; each connected to the next and the last one connected to the first one.

SECONDARY STATION. A simple HDLC protocol user.

SENSOR. Any transducer that converts the measurement of a physical quantity to an electrical signal.

SERIAL DATA BUS. A data bus capable of sending only one bit at a time, in series.

SERVICE SPECIFICATION. The specification of the service provided by a protocol layer.

SIMULATION. An approximated representation of the behavior of a system with a similar system.

SINGLE-POINT FAILURE. A failure of a component that, by itself, causes the failure of the system in which it is contained.

SPECIAL CONDITION. A regulatory document that adds to, or otherwise alters, the airworthiness standards for particular aircraft.

STATION. Bus user.

STATIONARY BUS CONTROL. Bus control that is continually performed by a single bus controller, or by one of its backups.

STATUS REGISTER. A register in an IC controller that holds the status of the state of certain controller functions.

STUB. The short length of cable used to attach a single LRU to a data bus.

SYSTEM INTEGRATOR. The developer who has the responsibility to integrate the various subsystems into a working system.

TOKEN PASSING PROTOCOL. A protocol that limits bus access to the user that has just received the token word.

UNBALANCED CONFIGURATION. A bus using the HDLC protocol that connects one primary and one or more secondary stations.

UNIDIRECTIONAL DATA BUS. A data bus with only one user that is capable of transmitting.

VALIDATION. The process of evaluating whether or not items, processes, services, or documents accomplish their intended purpose in their operating environment.

VERIFICATION. The act of reviewing, inspecting, testing, checking, auditing, or otherwise establishing and documenting whether or not items, processes, services, or documents conform to specified requirements.

## ACRONYMS AND ABBREVIATIONS

| | |
|---|---|
| μs | Microsecond |
| AC | Advisory Circular |
| ACK | Acknowledge |
| ACO | Aircraft Certification Office |
| AEEC | Airlines Electronic Engineering Committee |
| AFSC | Air Force Systems Command |
| AIAA | American Institute of Aeronautics and Astronautics |
| AIM | Advanced Integrated MUX |
| AIR | Aerospace Information Report |
| AIRLAB | Avionics Integration Research Laboratory |
| AP | Application Processor |
| ARINC | Aeronautical Radio Incorporated |
| ARP | Aerospace Recommended Practice |
| ASCB | Avionics Standard Communications Bus |
| BAC | Balanced Asynchronous Configuration |
| BC | Bus Controller |
| BCAC | Boeing Commercial Airplane Company |
| BCD | Binary Coded Decimal |
| BFCS | Beacon Frame Check Sequence |
| BIT | Built-In-Test |
| BIU | Bus Interface Unit |
| BNR | Binary |
| BOCP | Bit-Oriented Communications Protocol |
| BP | Basic Protocol |
| BUSY | Destination Busy |
| CA | Criticality Analysis |
| CE | Certification Engineer |
| CMC | Current Mode Coupler |
| CP | Combined Protocol |
| CPU | Central Processing Unit |
| CRC | Cyclic Redundancy Check |
| CSDB | Commercial Standard Data Bus |
| CSMA | Carrier Sense-Multiple Access |
| CTS | Clear To Send |
| DATAC | Digital Autonomous Terminal Access Communication |
| DC | Display Computer |
| DET | Driver Enable Timer |
| DITS | Digital Information Transfer System |
| DMA | Direct Memory Addressing |
| DME | Distance Measuring Equipment |
| DTSA | Dynamic Time Slot Allocation |
| EEC | Electronic Engine Control |
| EES | Electromagnetic Emission and Susceptibility |
| EFID | Electronic Flight Instrument Display |
| EFIS | Electronic Flight Instrument System |

| | |
|---|---|
| EIA | Electronic Industries Association |
| F/FA | Fault and Failure Analysis |
| FAA | Federal Aviation Administration |
| FAR | Federal Aviation Regulation |
| FCC | Flight Control Computer |
| FCS | Frame Check Sequence |
| FMEA | Failure Mode and Effects Analysis |
| FMECA | Failure Mode, Effects, and Criticality Analysis |
| FSM | Finite State Machine |
| FTA | Fault Tree Analysis |
| FTMP | Fault Tolerant Multi-Processor |
| GA | General Aviation |
| GAMA | General Aviation Manufacturers Association |
| HA | Hazard Analysis |
| HARP | Hybrid Automated Reliability Predictor |
| HDLC | High-Level Data Link Control |
| HERF | High Energy Radio Frequency |
| HIRF | High Intensity Radiated Frequency |
| HSRB | High Speed Ring Bus |
| HW | Hardware |
| Hz | Hertz |
| I/O | Input/Output |
| IACS | Integrated Avionic Computer System |
| IC | Integrated Circuit |
| ID | Identifier |
| IEEE | Institute of Electrical and Electronics Engineers |
| IFCS | Information Frame Check Sequence |
| IMR | Interrupt Mask Register |
| ISO | International Standards Organization |
| IVT | Interrupt Vector Table |
| LRU | Line Replaceable Unit |
| LSB | Least Significant Bit |
| LSI | Large Scale Integration |
| LTPB | Linear Token Passing Bus |
| m | Original Address of Last Transmission |
| MC | Mode Code |
| MCFCS | Message Control Frame Check Sequence |
| MFCS | Message Frame Check Sequence |
| MHz | megahertz |
| MIL-HDBK | Military Handbook |
| MIL-STD | Military Standard |
| ML | Message Length |
| MPSC | Multi-Protocol Serial Controller |
| ms | millisecond |
| MSB | Most Significant Bit |
| MT | Message Time |
| MTBF | Mean Time Between Failure |
| MTTR | Mean Time to Repair |
| MUX | Multiplexer |
| n | Address of User Performing Computation |
| N | Maximum Number of Users |
| NASA | National Aeronautics and Space Administration |

| | |
|---|---|
| NCTS | Not Clear To Send |
| OSI | Open Systems Interconnection |
| PMA | Parts Manufacturer Approval |
| PROM | Programmable Read-Only Memory |
| RAM | Random Access Memory |
| RAT | Ring Admittance Timer |
| RF | Radio Frequency |
| RIM | Ring Interface Module |
| RIU | Ring Interface Unit |
| RR | Read Register |
| RRT | Ring Rotation Time |
| RS | Recommended Standard |
| RT | Remote Terminal |
| RTCA | Radio Technical Commission for Aeronautics |
| RTE | Real-Time Executive |
| RTS | Request To Send |
| SAE | Society of Automotive Engineers |
| SAI | Systems Architecture and Interfaces |
| SC | Special Condition |
| SCC | System Configuration Controller |
| SCM | Software Configuration Management |
| SCP | Self-Checking Pair |
| SG | Synchronization Gap |
| SIM | Serial Interface Module |
| SIR | Shared Interface RAM |
| SMF | Self Monitor Function |
| SQA | Software Quality Assurance |
| SSA | System Safety Assessment |
| STC | Supplemental Type Certificate |
| SW | Software |
| $T_c$ | Count Duration |
| TC | Type Certificate |
| TCAS | Traffic Alert and Collision Avoidance System |
| TDMA | Time Division Multiple Access |
| $T_F$ | Frame Time |
| TFCS | Token Frame Check Sequence |
| TFEDF | Token Frame Ending Delimiter Field |
| TG | Terminal Gap |
| THT | Token Holding Timer |
| TI | Transmit Interval |
| $T_m$ | Wait Time for User |
| TRT | Token Rotation Timer |
| TSDF | Token Starting Delimiter Field |
| TSO | Technical Standard Order |
| UAC | Unbalanced Asynchronous Configuration |
| UNC | Unbalanced Normal Configuration |
| V&V | Verification and Validation |
| VLSI | Very Large Scale Integration |
| VOR | VHF Omnidirectional Range |
| VT | Validation Testing |
| WR | Write Register |



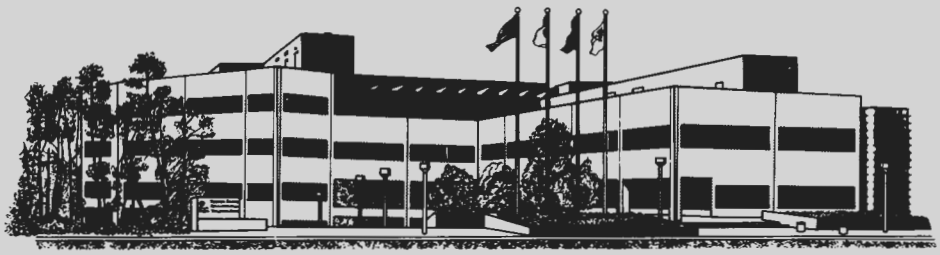
☆U.S. GOVERNMENT PRINTING OFFICE: 1993 - 504-078/80199

U.S. Department of Transportation
**Federal Aviation Administration**

**DOT/FAA/CT-88/10**

# HANDBOOK - VOLUME II
# DIGITAL SYSTEMS VALIDATION

## CHAPTER 19
## PILOT - VEHICLE INTERFACE



FAA WJH Technical Center
Tech Center Library
Atlantic City, NJ 08405

FAA TECHNICAL CENTER
Atlantic City, N.J.

AVAILABLE IN ELECTRONIC FORMAT

**FEDERAL AVIATION ADMINISTRATION**
**TECHNICAL CENTER**
**ATLANTIC CITY INTERNATIONAL AIRPORT, NEW JERSEY 08405**

**Technical Report Documentation Page**

| 1. Report No. | 2. Government Accession No. | 3. Recipient's Catalog No. |
|---|---|---|
| DOT/FAA/CT-88/10 | | |

| 4. Title and Subtitle | 5. Report Date |
|---|---|
| PILOT-VEHICLE INTERFACE<br>(DIGITAL SYSTEM VALIDATION HANDBOOK-VOLUME II) | November 1993 |
| | 6. Performing Organization Code |
| 7. Author(s) | 8. Performing Organization Report No. |
| L. Harrison, J. Janowitz, and M. Castronuovo | DOT/FAA/CT-88/10 |
| 9. Performing Organization Name and Address<br>Galaxy Scientific Corporation<br>2500 English Creek Avenue<br>Building 11<br>Pleasantville, New Jersey 08232 | 10. Work Unit No. (TRAIS) |
| | 11. Contract or Grant No.<br>DTFA03-89-C-00043 |
| | 13. Type of Report and Period Covered<br>Tutorial<br>Handbook Chapter 19 |
| 12. Sponsoring Agency Name and Address<br>U.S. Department of Transportation<br>Federal Aviation Administration<br>Technical Center<br>Atlantic City International Airport, NJ 08405 | 14. Sponsoring Agency Code<br>ACD-230 |

15. Supplementary Notes

Pete Saraceni, FAA Technical Center, Program manager, (609) 485-5577
(Note: This Tutorial is a condensed version of FAA Technical Center Final Report DOT/FAA/CT-92/21)

16. Abstract

The cockpits of the early transport aircraft were quite different from those produced today. Older cockpits contained numerous "steam gauge" style indicators. As technology advanced, these older electromechanical indicators were gradually replaced by newer, more reliable digital systems. Digital flight control and avionic systems are being used increasingly in modern aircraft. This trend yields cockpits of greater complexity and has swelled the amount of information with which the crew must deal. The way the pilot controls and monitors the state of the aircraft has also been greatly influenced by the increased use of digital systems.

Additionally, new methods of aircraft system monitoring and control are being researched and implemented. These systems use new display technology, programmable display formats, voice input and output, and other new input and control devices. Systems and their cockpit interfaces were added as technological advances were made and new requirements generated. The human interface was given little consideration in the layout of the cockpit. Human qualities and failure modes were not taken into account in the cockpit design process. As the number of systems, components, indicators, and switches multiplied, the potential for error also grew.

The emphasis of this report is on civil transport aircraft. This includes technologies such as displays, controls, and design methodologies, along with human factors concerns. The report consists of four sections: civil transport cockpit, cockpit standards, cockpit technology, and cockpit human factors. This report is to serve as a guide to Certification Engineers who are faced with the task of certification of new cockpit technologies.

| 17. Key Words | 18. Distribution Statement |
|---|---|
| Pilot-Vehicle Interface, Man-Machine Interface, Human Factors, Situational Awareness, Flight Deck Automation, Cockpit Design, Pilot/Crew-Workload/ Station/Tasks, Crew Resource Management | Document is available to the public through the National Technical Information Service, Springfield, Virginia 22161 |

| 19. Security Classif. (of this report) | 20. Security Classif. (of this page) | 21. No. of Pages | 22. Price |
|---|---|---|---|
| Unclassified | Unclassified | 121 | |

Form DOT F 1700.7 (8-72) Reproduction of completed page authorized

TABLE OF CONTENTS


| Chapter | | | Page |
|---|---|---|---|
| 1. | INTRODUCTION | | 19-1 |
| | 1.1 | Background | 19-1 |
| | 1.2 | Scope | 19-1 |
| 2. | THE FLIGHT DECK ENVIRONMENT | | 19-5 |
| | 2.1 | Cockpit Displays and Controls | 19-5 |
| | | 2.1.1 Cockpit Regulations | 19-6 |
| | | 2.1.2 The Analog Cockpit | 19-7 |
| | | 2.1.3 The Digital Cockpit | 19-8 |
| | | 2.1.4 The Integrated Cockpit | 19-9 |
| | 2.2 | Aircraft System and Equipment Implementations | 19-11 |
| 3. | CERTIFICATION REGULATIONS AND GUIDELINES | | 19-13 |
| | 3.1 | Certification and Regulations | 19-13 |
| | 3.2 | Pilot-Vehicle Interface Guidelines Developed by SAE Committees | 19-13 |
| | 3.3 | G-10 Committee - Aerospace Behavioral Engineering Technology | 19-15 |
| | | 3.3.1 ARP4155 - Human Interface Design Methodology for Integrated Display Symbology | 19-15 |
| | 3.4 | A-4 Committee - Aircraft Instruments | 19-16 |
| | | 3.4.1 AS8034 - Minimum Performance Standard for Airborne Multipurpose Electronic Displays | 19-16 |
| | | 3.4.2 ARP1874 - Design Objectives for Cathode Ray Tube Displays for Part 25 (Transport) Aircraft | 19-17 |
| | | 3.4.3 AIR1093 - Numeral, Letter, and Symbol Dimensions for Aircraft Instrument Displays | 19-17 |
| | 3.5 | S-7 Committee - Flight Deck Handling Qualities Standards for Transport Aircraft | 19-18 |
| | | 3.5.1 ARP4101 - Flight Deck Layout and Facilities | 19-18 |
| | | 3.5.2 ARP4101/4 - Flight Deck Environment | 19-18 |
| | | 3.5.3 ARP4102 - Flight Deck Panels, Controls, and Displays | 19-19 |
| | | 3.5.4 ARP4102/4 - Flight Deck Alerting System | 19-21 |
| | | 3.5.5 ARP4102/6 - Communications and Navigation Equipment | 19-22 |

5. HUMAN FACTORS AND THE PILOT-VEHICLE INTERFACE 19-63

5.1 Introduction 19-63

5.1.1 The National Plan for Aviation Human Factors 19-63
5.1.2. Scope and Organization 19-63

5.2 Safety and the Pilot-Vehicle Interface 19-64

5.2.1 Factors Contributing to Aviation Safety 19-64
5.2.2 Factors Contributing to Aviation Incidents and Accidents 19-64
5.2.3 Safety and Situational Awareness 19-65
5.2.4 Digital Systems/Aircraft Incidents, Accidents, and Lessons Learned 19-65

5.3 Cockpit Automation and the Pilot-Vehicle Interface 19-68

5.3.1 Definition 19-68
5.3.2 The National Plan Objectives for Automation 19-69
5.3.3 Reasons to Automate 19-69
5.3.4 Automation Concerns 19-70
5.3.5 Guidelines for Cockpit Automation 19-72
5.3.6 The Role of the Pilot and Crew in an Automated Cockpit 19-73
5.3.7 Error Management in the Automated Cockpit 19-74

5.4 Design Issues and the Pilot-Vehicle Interface 19-75

5.4.1 Cockpit Design Philosophies 19-75
5.4.2 Cockpit Layout and Devices 19-75
5.4.3 Information Presentation, Organization, and Access 19-77

5.5 Crew Workload and the Pilot-Vehicle Interface 19-83

5.5.1 Definition 19-83
5.5.2 Workload Assessment and Measurement 19-83
5.5.3 The Effects of Automation on Workload 19-84
5.5.4 Attention and Workload 19-84

5.6 Crew Resource Management, Training, and the Pilot-Vehicle Interface 19-85

5.6.1 Crew Interaction and the Cockpit 19-85
5.6.2 Crew Resource Management and Safety 19-87
5.6.3 Pilot and Crew Training 19-87

LIST OF ILLUSTRATIONS


| Figure | | Page |
|---|---|---|
| 2.1-1 | PRIMARY FLIGHT DISPLAY LAYOUT | 19-6 |
| 2.1-2 | SYMBOLOGY LOCATION DIAGRAM | 19-8 |
| 2.1-3 | MCDONNELL-DOUGLAS MD-11 COCKPIT | 19-10 |
| 3.5-1 | RECOMMENDED GEOMETRY FOR THE HORIZON LINE | 19-24 |
| 4.2-1 | SCHEMATIC DIAGRAM OF MONOCHROME CATHODE RAY TUBE | 19-30 |
| 4.2-2 | PRINCIPLE OF SHADOW-MASK CATHODE RAY TUBE | 19-32 |
| 4.2-3 | OPERATING PRINCIPLE OF TRINITRON COLOR CATHODE RAY TUBE | 19-33 |
| 4.2-4 | CONSTRUCTION OF BEAM INDEX CATHODE RAY TUBE | 19-34 |
| 4.2-5 | CONSTRUCTION OF PENETRATION PHOSPHOR CATHODE RAY TUBE | 19-35 |
| 4.2-6 | BEAM FORMING REGION OF CATHODE RAY TUBE | 19-36 |
| 4.2-7 | MAGNETIC AND ELECTROSTATIC FOCUSING METHODS | 19-37 |
| 4.2-8 | RASTER SWEEP PATTERN | 19-39 |
| 4.2-9 | SCHEMATIC DIAGRAM OF LIGHT EMITTING DIODE | 19-41 |
| 4.2-10 | RELATIVE LUMINOUS INTENSITY VERSUS FORWARD CURRENT | 19-41 |
| 4.2-11 | LIGHT EMITTING DIODE OPTICAL CROSSTALK | 19-43 |
| 4.2-12 | RESPONSE TIMES FOR A TWISTED NEMATIC LIQUID CRYSTAL CELL | 19-45 |
| 4.2-13 | CROSS SECTIONAL VIEW OF THIN-FILM TRANSISTOR LIQUID CRYSTAL DISPLAY | 19-48 |
| 4.2-14 | THIN-FILM ELECTROLUMINESCENCE SANDWICH CONSTRUCTION | 19-50 |
| 4.2-15 | AC PLASMA DISPLAY PANEL STRUCTURE | 19-53 |
| 4.3-1 | BASIC HEAD-UP DISPLAY COMPONENTS | 19-57 |

# 1. INTRODUCTION

## 1.1 Background

The cockpits of the early transport aircraft were quite different from those produced today. Older cockpits contained numerous "steam gauge" style indicators. These related to essential aircraft information such as altitude, airspeed, fuel, and various engine parameters. As technology advanced, these older electromechanical indicators were gradually replaced by newer, more reliable digital systems.

Digital flight control and avionic systems have significant advantages, such as higher speed, less power consumption, reduced weight, lower volume, and lower cost when contrasted to older electromechanical systems. These systems are being used increasingly in modern aircraft. This trend yields cockpits of greater complexity and has swelled the amount of information with which the crew must deal. The way the pilot controls and monitors the state of the aircraft has also been greatly influenced by the increased use of digital systems.

Initially, each of these digital systems was a "stand-alone" component performing its intended function. Systems and their cockpit interfaces were added as technological advances were made and new requirements generated. The human interface was given little consideration in the layout of the cockpit. Human qualities and failure modes were not taken into account in the cockpit design process. As the number of systems, components, indicators, and switches multiplied, the potential for error also grew.

These stand-alone systems, along with their related indicators and switches, have now been implemented using digital technology. The use of this technology has opened the door for massive integration of the once stand-alone systems, through the use of data bus interconnections. Additionally, new methods of aircraft system monitoring and control are being researched and implemented. These systems use new display technology, programmable display formats, voice input and output, and other new input and control devices.

## 1.2 Scope

The Pilot-Vehicle Interface (PVI) is the man-machine interface of the aircraft flight deck. PVI concerns itself with numerous technologies, especially when viewed from a military perspective. Some of the PVI-related technologies include configuration concepts, the pilot's associate, helmet systems, displays, controls, design methodology, escape systems, simulation, weapon system integration, supportability, and survivability.

Since the emphasis of this report is on civil transport aircraft, many of the military PVI technologies receive little or no consideration. The difference in mission dictates differences in technologies used. The emphasis for commercial

This report identifies issues dealing with the PVI and their impact on flight safety. This report was assembled using reviews of the literature, interviews with experts in the field, and reviews of accident and incident reports. It is to serve as a guide to CEs who are faced with the task of certification of new cockpit technologies.

## 2. THE FLIGHT DECK ENVIRONMENT

Cockpit complexity can range from very simple for general aviation aircraft to highly complex for the modern civil transport. This section provides fundamental information which is helpful for understanding this complexity by introducing the reader to some of the basic regulations that influence displays and instruments in the civil transport cockpit and showing how these regulations were satisfied from older cockpit designs through newer ones.

### 2.1 Cockpit Displays and Controls

There are many influences that shape cockpit design philosophy. New technology has replaced the cockpit full of electromechanical indicators with a few large Cathode Ray Tube (CRT) displays. Numerous switches controlling various systems have been integrated into Control Display Units (CDUs). The flight engineer's position has been eliminated and his functions have been integrated into captain and first officer duties. The human interface has also been the target of extensive research and consideration for cockpit design. Some of the other fundamental influences on cockpit equipment include reliability, power consumption, volume, weight, maintenance considerations, and cost.

These considerations have led to the development of a wide variety of input and output devices in the cockpit. Many devices are used in the cockpit to input information into the various aircraft systems. This information is used for communication, navigation, and other tasks required for flying. Following is a list of some input devices in current use or planned for use in the near future:

- Yoke,
- Rudder pedals,
- Side stick controller,
- Trackball,
- Capacitive touch pad,
- Keyboard,
- Voice, and
- Discrete switches, push buttons, knobs, and levers.

Cockpit output devices include:

- Printers,

Note that although there is freedom to configure the display presentation in infinite ways, the same basic T configuration is maintained as when individual indicators are used. The attitude is in the center of the display. Airspeed is adjacent to and directly to the left of the attitude display. Altitude is adjacent to and directly to the right of the attitude display. The heading is adjacent to and directly below the attitude display.

Standardization of the PVI is not only driven by the FARs, but also by recognized standards such as the SAE ARP4102/7, "Electronic Displays". Appendix A of this SAE document covers symbology location for the PFD as shown in figure 2.1-2. Other regulations and standards are examined in section 3 of this report.

In order to meet the FAR requirements for failure conditions, designers implement redundant systems and displays. For instance, if the pilot's PFD fails, another is available in front of the other crew member. If both of these fail, then separate indicators providing the same function may be used.

The influence of technology on cockpit design is seen by comparing the older and newer cockpits. Older analog cockpits are full of discrete electromechanical indicators, each representing a separate system or parameter. When digital systems were introduced into the cockpit, many of the indicators and switches were replaced by digital displays, CRTs, and keyboards. Modern integrated cockpits are now manufactured with large CRTs completely spanning the instrument panel and several Multifunction Control Display Units (MCDUs) mounted in the pedestal.

Older aircraft often are refurbished by manufacturers to extend the operating life of the aircraft. The cockpit is frequently the focus of major changes during this time. For instance, the Boeing 747 had 971 lights, gauges, and switches on the earlier models. This number was reduced to 365 on the Boeing 747-400 model in part by using MCDUs and 6 large CRTs.

### 2.1.2 The Analog Cockpit

The Boeing 727 is an example of the analog cockpit. There are over 40 indicators and numerous switches, status lamps, and buttons just on the front panel. In addition, there is a third crew member's panel with further rows of indicators, switches, buttons, and status lamps.

The basic "T" configuration of the primary flight displays is directly in front of the flight crew on each side of the cockpit. The displays consist of a Horizontal Situation Indicator (HSI), Flight Director Indicator (FDI), Airspeed Indicator, and Altitude Indicator.

The engine status indicators are located in the center panel along with the landing gear interface and other discrete annunciators and indicators. Analog cockpits, in general, have a cluttered appearance, requiring more eye movement and time spent by the crew with eyes focused in the cockpit. An additional difficulty occurs when new systems are added to the existing aircraft. Finding space in the cockpit for indicators is a major problem.

parameter. With the digital cockpit, MCDUs with integral CRT displays became the preferred manner of systems control for new designs.

Much of the clutter associated with the analog cockpit has been removed in the digital cockpit. The McDonnell-Douglas MD-80 has 492 switches, gauges, and lights compared to 833 on the earlier McDonnell-Douglas DC-10 (Bekemeyer and Shannon 1986).

### 2.1.4 The Integrated Cockpit

The McDonnell-Douglas MD-11 is an example of an integrated cockpit and is shown in figure 2.1-3. The front panel consists primarily of six large (8 inch x 8 inch) color CRTs across the panel along with the landing gear interface. The CRTs are used to display the primary flight data, navigation data, and engine and systems data. There are three pedestal-mounted MCDUs. Two MCDUs are dedicated to the Flight Management System (FMS) and the third is used for maintenance functions. Flight director controls are located on the glareshield.

The McDonnell-Douglas DC-10, predecessor of the McDonnell-Douglas MD-11, had a total of 833 switches, gauges, and lights as compared to 362 on the McDonnell-Douglas MD-11. There also was a 75 percent reduction in the Line Replaceable Unit (LRU) count due to digital integration (Bekemeyer and Shannon 1986). In addition, the crew size was reduced from three on the McDonnell-Douglas DC-10 to two on the McDonnell-Douglas MD-11. This was accomplished by automating the functions formerly performed by the flight engineer.

Functions of the flight guidance and control systems were initially stand-alone implementations on aircraft such as the Boeing 727. In modern automated aircraft, such as the McDonnell-Douglas MD-11, many functions of the flight guidance and control systems are integrated into the Flight Control Computer (FCC). These systems include (Bekemeyer and Shannon 1986):

- Autopilot,
- Flight director,
- Yaw damper,
- Longitudinal stability augmentation,
- Pitch trim,
- Autothrottle, and
- Stall warning.

The cockpit has a clean, uncluttered appearance because of the integration of many systems into a common CRT display. The problem of clutter, however, has not been eliminated. Individual displays can become cluttered with extraneous information. Careful consideration should be given to the information requirements of particular tasks and the format of presented data.

## 2.2 Aircraft System and Equipment Implementations

Many systems that exist in transport aircraft require an output device as well as a method of input. Some systems use CRTs dedicated to that particular system, while other systems share a common, multifunctional display device.

The SAE ARP4102/7 recommends criteria for displays associated with various cockpit systems. The identified displays include:

- PFD,
- Navigation Display (ND),
- Flight Management Display for FMS,
- System Display (SD), and
- Alerting Display (AD).

Systems requiring a flight crew interface will continually be developed due to changing technologies and the overriding concern for flight safety. How information is presented on the flight deck is a major concern and is the focus of considerable research. One such study (Pomykacz 1991) identifies displays that airline operators intend to use for the Data Link system. Likewise, if an Enhanced Vision System (EVS) or Microwave Landing System (MLS) becomes a requirement in the future, similar studies will be necessary.

Some systems may require a dedicated CRT while other systems such as Data Link may share use of an existing display device. An advantage of digital display systems is that they are easily made multifunctional. When a new system requires a display interface, it is easily accommodated due to the wealth of programmable devices existing in the modern transport cockpit. What is not always obvious, however, is when, where, and what to display.

# 3. CERTIFICATION REGULATIONS AND GUIDELINES

## 3.1 Certification and Regulations

Certification is the process of obtaining FAA approval for the design, manufacture, and/or sale of a part, subsystem, system, or aircraft, by demonstrating that it complies with all applicable government regulations (Elwell et al. 1991). The requirements for the certification of flight deck systems are covered in the FARs. FAR Part 25, sections 1303, 1309, 1321, and 1322 contain regulations related to PVI.

FAR 25.1303, "Flight and Navigation Instruments", calls for each pilot station to have an airspeed indicator, altimeter, rate-of-climb indicator, bank-and-pitch indicator, rate-of-turn indicator, and direction indicator (non-stabilized magnetic compass). In addition, a free air temperature indicator, a clock, and a direction indicator (gyroscopically stabilized, magnetic, or nonmagnetic) must be visible from each pilot station. FAR 25.1321, "Arrangement and Visibility", specifies the flight deck instrument installation, arrangement, and visibility. This section establishes the basic instrument "T". FAR 25.1322, "Warning, Caution and Advisory Lights", defines the color code for warning, caution, and advisory lights. FAR 25.1309, "Equipment, Systems, and Installations," requires that systems and equipment be designed to perform their intended functions under any foreseeable operating conditions.

In order to assist the equipment manufacturer in meeting the requirements of certain FAR sections, the FAA publishes documents known as Advisory Circulars (ACs). The ACs specify various methods to comply with the FARs. AC 25-11 entitled "Transport Category Airplane Electronic Display Systems" provides direction for the certification of CRT displays that are used by pilots for guidance control or decision making. It contains general certification considerations; color, symbology, coding, clutter, dimensionality, and attention getting requirements; display visual characteristics; failure modes; information display and formatting; and integrated display and mode considerations.

Technical Standard Orders (TSOs) are other documents or standards published by the FAA. TSOs outline the minimum performance standards a part must meet. They are used by manufacturers for equipment or part production approval. The TSOs reference documents such as the FARs and other industry standards. A more in-depth discussion of the certification process and the relationship among regulatory documents can be found in Elwell et al. 1991.

## 3.2 Pilot-Vehicle Interface Guidelines Developed by SAE Committees

The SAE is a world-wide organization of over 60,000 members representing different engineering disciplines. These members come from civil, mechanical, electrical, chemical, and metallurgical engineering, academic, and scientific backgrounds. The SAE Technical Standards Board is responsible for managing and

performance recommendations that are intended as guides toward standard engineering practice.

The following sections provide summaries of PVI-related documents developed by the three SAE Committees. All the documents contain references to reports pertinent to the subject being reviewed and most contain a glossary of key terms. It is important to note that these summaries are intended to give the reader the general nature of the document. If more detailed information is needed, it is recommended that the reader obtain the appropriate SAE document.

### 3.3 G-10 Committee - Aerospace Behavioral Engineering Technology

#### 3.3.1 ARP4155 - Human Interface Design Methodology for Integrated Display Symbology

##### 3.3.1.1 Scope

This document provides an approach to designing symbology for integrated displays. This approach stresses the relationship among symbols, the information they encode, the context in which the symbols are displayed, and the tasks these symbols support.

##### 3.3.1.2 Background

In developing integrated display symbology, the designer must ensure that display symbols:

- Convey the information they encode,
- Remain effective when used in combination with related symbols, thus enabling the user to achieve a prescribed level of performance, and
- Do not interfere with the interpretation of other flight deck symbols as well as the performance of other flight deck tasks.

As the amount of information in flight decks increases and display devices become more flexible, the ability to satisfy the above requirements diminishes without some type of compromise. In order to produce an effective display, the designer must implement a structured design approach early in the development cycle.

One way to produce an effective display is to use consistent symbol-task pairings. This facilitates pilot training as pilots operate different subsystems in one aircraft or transition from one aircraft type to another. As flight decks become more integrated and increasingly complex, symbol consistency becomes more difficult to maintain. Other factors that contribute to the difficulty in symbol consistency include changes in flight deck configuration throughout the operating life of an aircraft, improvements in technology, and changes in regulatory, airline, and ATC requirements.

This ARP illustrates a sample interconnection configuration among these components.

3.4.1.2 Standards

Standards covered in this document are organized as:

- General Standards,
- Minimum Performance Standards Under Standard Conditions,
- Minimum Performance Standards Under Environmental Conditions, and
- Test Procedures.

3.4.2 ARP1874 - Design Objectives for Cathode Ray Tube Displays for Part 25 (Transport) Aircraft

3.4.2.1 Scope

This SAE document contains information that is related to SAE document AS8034 (covered above). It recommends performance criteria for direct view CRT instrumentation on the flight deck of FAR Part 25 aircraft. ARP1874 covers general requirements for CRT displays such as equipment functions, environmental conditions, implosion protection, and malfunction indications.

3.4.2.2 Detail Requirements

The detail requirements include the CRT's physical characteristics, photo-colormetric characteristics, and operating time. As part of physical characteristics, this ARP recommends useful screen size, viewing characteristics, etc. Photo-colormetric characteristics include luminance, color, and the color difference between symbols and between symbols and background. The operating time of CRT displays includes warm-up time, lag time, and time for data updating. Warm-up time is specified to be less than one minute from initial turn-on. At this point, the CRT will display usable and accurate information. The lag time between the selection of a format on the CRT and the actual display of the data on-screen should not exceed one second. The lag-time between selection of flight/mode data and actual display should not be more than 1/4 of a second. Other lag time requirements are found in ARP1068B "Flight Deck Instrumentation, Display Criteria and Associated Controls for Transport Aircraft." The rate for data update is specified at a minimum of 15 Hertz (Hz).

The remaining portion of ARP1874 covers the definition of key display terms.

3.4.3 AIR1093 - Numeral, Letter, and Symbol Dimensions for Aircraft Instrument Displays

3.4.3.1 Introduction

There are many factors that determine the readability of instrument dial characters. This makes the task of establishing a concise set of rules for all

### 3.5.3 ARP4102 - Flight Deck Panels, Controls, and Displays

#### 3.5.3.1 Scope and Operational Requirements

The scope of this document is to recommend criteria for the design, installation, and operation of flight deck panels, controls, and displays. It is to be used for transport category aircraft. The operational requirements for this ARP are general in nature. For example, controls and displays should be simple, rational, and user-friendly to ensure effective and error-free operation. The requirements also mention the cases where controls and displays can or should be integrated.

#### 3.5.3.2 Panels

This ARP provides details on the location and function as well as the display arrangement for the following panels:

- Pilot's Primary Instrument Panel,
- Center Instrument Panel,
- Glareshield Panel,
- Pedestal,
- Pilot's Overhead Panel,
- Third Crew Member's Panel, and
- Circuit Breaker Panel (information provided in ARP4101/5).

As an example, the Pilot's Primary Instrument Panel should be located directly in front of the pilot and as close to the windshield lower cutoff as possible. The latter criteria is required in order to minimize the pilot's vertical scan between the head-down displays and the outside environment. In addition, the panel should be oriented so that parallax is minimized and reflections and glare are prevented. Functionally, this panel should display vertical and horizontal situation, command information, and navigation data.

The display arrangement for the above panels specifies the location of the panels' respective components. For example, in specifying the display arrangement of the Pilot's Primary Instrument Panel, this ARP calls for the following:

- Pitch and roll attitude, flight path situation, and command information shall be displayed in the central upper area of the panel.
- In situations where the Primary Attitude Instrument (PAI) is located above the Primary Navigation Instrument (PNI), the PAI should be located no more than one inch from the vertical plane passing through the DEP and parallel to the longitudinal axis of the aircraft. When the PAI is located beside the PNI, the vertical center line of the PAI should not be more that 2.5

### 3.5.4 ARP4102/4 – Flight Deck Alerting System

#### 3.5.4.1 Scope

This document provides design criteria for the Flight Deck Alerting System (FAS). There are four levels of alerts defined for the FAS, level 0 through level 3. These levels are defined as:

- Level 3 – Warning (an emergency situation, e.g. serious system failures),
- Level 2 – Caution (an abnormal situation, e.g. failure with no immediate impact on safety),
- Level 1 – Advisory (recognition situation, e.g. failure leading to loss of redundancy), and
- Level 0 – Information.

#### 3.5.4.2 Operational Requirements

In general, the FAS should possess a master aural attention-getting sound for alert levels 1, 2, and 3. The FAS should also contain one or more alphanumeric displays so that a corresponding message can be shown on the display during an alert. The urgency level of the alert should vary as a function of the flight phase and current aircraft conditions.

As part of the functional requirements, four types of alerts and their respective applications are described. The four alerts are aural, discrete aural, voice, and tactile. The aural alerts, also known as "attenson," are unique attention-getting sounds. The same sound cannot be used for different functions on the flight deck. The attenson is to be applied to level 2 and 3 alerts and optionally to level 1. The sound is to be transmitted through cockpit speakers and crew headsets. The use of discrete aural alerts is not recommended for the FAS. The attenson, along with a display alert, is considered to provide a warning superior to discrete aural alerts. The voice alert is a warning method that can be used to supplement the aural alert and the display for level 3 time-critical alerts. A tactile warning (i.e. a stick shaker) is considered acceptable when it is supplemented by aural and visual alerts for Level 3.

#### 3.5.4.3 Panels, Controls, and Displays

The panel that houses the FAS should be visible and accessible to both pilots when the flight deck seats are located such that the pilot's eyes are at the DEP. The control panel, if it exists, should include all the controls needed for the operation of the FAS.

Displays should include the FAS visual display and alerting lights. The FAS display should be alphanumeric. Its function is to supply an alphanumeric readout that will direct attention to an indicator or control device at another location. The initial alphanumeric readout should be cancelled automatically if the fault is corrected. The alerting lights of the FAS are to be an independent

indicate to each pilot all communication frequency and mode information. This can be accomplished on one display visible to both pilots.

3.5.6 ARP4102/7 - Electronic Displays

3.5.6.1 Scope

This ARP supplies recommendations for flight deck electronic displays on transport category aircraft. These electronic displays include electronic flight instruments, alert displays, aircraft system displays, and control/display units for flight management and radio management systems.

3.5.6.2 Operational Requirements

The electronic display must be legible while operating in all lighting conditions found in the flight deck and while operating in a monochrome or degraded mode. Displays on a pilot's panel that are required to be monitored by a third crew member should be visible and legible from his/her seated position. Data viewed on the display should always enter and leave the display in a smooth manner. When power fluctuations or power transfers occur, the display should operate without serious interruption. In addition, display content or values chosen by the pilot should not change.

3.5.6.3 Displays

General guidelines and specific details are supplied for the following types of displays:

- PFD/EADI,
- Electronic HSI/ND,
- Flight Management Display,
- Alerting Display, and
- Systems Display.

Generally, it is important to ensure that electronic displays have the proper lag time between the selection of data and the resulting display of data. Primary flight data should not experience more than a 1/4 second lag. For nonessential data, lag time should be four seconds or less.

The specific information to be presented on each display also is provided.

3.5.7 ARP4102/7 Appendix A - Electronic Display Symbology for EADI/PFD

3.5.7.1 Scope and Contents

This Appendix is an extension of ARP4102/7. It recommends the symbols to be used for the EADI/PFD. The geometry and location of these symbols are also provided. All symbol geometries are given in a table format. For example, figure 3.5-1

display parameters should be integrated with head-down instruments whenever possible.

3.5.8.4 Display Criteria

HUD systems can be used to repeat, augment, or replace Head-Down Displays (HDDs) for any or all phases of flight. The format used on the display should allow the pilot to transition from head-up to head-down quickly and easily. The operation of the system should be intuitive and require minimal training. In addition to providing basic information such as attitude, heading, flight path angle, airspeed, and altitude, the HUD display should provide:

- Visual Approach information,
- See-to-Land Instrument Approach/Go-Around information,
- Non-Visual Takeoff and Landing information, and
- Warnings (i.e. HUD system failure and input data failure).

HUD system displays also have associated symbology criteria such as symbol quality, character size, symbol accuracy, stability, and brightness.

3.5.8.5 Colors

HUD system displays should implement color only when there is an improvement over monochrome displays. If color is used, color selections should be consistent with those used for the corresponding symbol on the HDD.

3.5.9 ARP4103 - Flight Deck Lighting for Commercial Transport Aircraft

3.5.9.1 Scope and Requirements

ARP4103 recommends criteria for the lighting systems and the interface required for flight deck areas, controls, and displays. Operational requirements include instrument lighting; panels (lightplates); circuit breakers; warnings, cautions, and advisory system lighting; and general and utility flight deck lighting.

3.5.9.2 Lighting Controls and Locations

Lighting controls are to include brightness controls in addition to the requirements found in ARP4102, paragraph 5.3. Controls should be located conveniently for the appropriate crew member. The light switches that are to be controlled by both pilots should be placed so that both pilots can reach them while seated with seat belts fastened. Dome or ceiling light controls should be located on the light or near the doorway into the area to be lighted. The knobs for white lighting intensity controls should be transilluminated with a white engraved arrow on top. The only exception to this are instrument bezel knobs. Standby lighting controls should also be furnished. They are to be activated during loss of aircraft power using an emergency battery supply. The minimum intensity of these lights should enable pilots to read instrumentation.

# 4. PILOT-VEHICLE INTERFACE TECHNOLOGY

This section examines some of the design tools used in the development and testing of aircraft cockpits. It also considers the various technologies relating to displays and controls in the modern transport cockpit. Particular focus is given to the digital facets of the input and output devices.

## 4.1 Cockpit Design Tools

Design of the cockpit necessarily involves the melding of numerous technologies relating to PVI. The diversity of these technologies, magnitude of the tasks, safety demands, cost, and other factors require the use of design tools to aid the design engineers in this challenging task. Design tools allow initial system development, testing, and modification to take place before a final product is produced. Changes made early in the design process have a significantly smaller impact on product cost than changes made later in the design process. Tools such as rapid prototyping and Computer Aided Design (CAD) assist the engineers in the design and development process.

The rapid configuration of a test environment is beneficial to evaluating new concepts, technologies, and cockpit arrangements. Displays, keyboards, and voice applications, as well as other issues such as lighting can be examined early in the design cycle. Also, the myriad human factors issues can be examined and problems can be identified and corrected before the product is manufactured.

### 4.1.1 Rapid Prototyping

Rapid prototyping allows a system model to be developed quickly and inexpensively. Arkusinski and Hoenbach (1988) studied rapid prototyping in the development of CDUs. Mock-ups of CDUs for a navigation system were built to allow a test crew to "fly" the unit in simulation before production began. This approach allowed changes to be incorporated into the system at an early stage and avoided major overhauls late in the project.

### 4.1.2 The Computer Mock-Up

The Computer Mock-Up (CMU) is a flexible, inexpensive design tool that can be used in the phase between initial theoretical specifications and flight tests. Schumacher and Starke (1987) describe a CMU implemented as a full-size model cockpit for a light transport helicopter. Simulation models are initiated by user inputs. The aim of the tool is to collect information about the ergonomic design, technical realization, and operational ability of future cockpits.

### 4.1.3 Reconfigurable Cockpit

The Reconfigurable Cockpit (RCP) is a PVI design tool for simulating the hardware and software capabilities of a proposed crew station. The objective of the RCP

### 4.2.1 Display Technology

The pilot obtains information about the state of all the monitored aircraft systems through some type of indicator or display. This section analyzes various technologies that are used in the manufacture of displays. Various display technologies are examined, such as the CRT, LC display, Electroluminescent (EL) display, Vacuum Fluorescent Display (VFD), and plasma display. In addition, failure modes are identified.

### 4.2.2 Cathode Ray Tubes

The CRT was used prior to World War II to display raw analog signals from airborne radar and airborne radio navigation equipment (Hunt 1983). For instance, the CRT could display the return radar pulse from a target aircraft.

During the 1950s the need was recognized that more information concerning the target should be presented to the pilot of a combat aircraft. This, coupled with the need for the pilot to maintain visual contact led to the development of the HUD. Parameters such as airspeed and attitude could now be displayed to the pilot along with target-related data.

CRTs began to be used widely in many applications outside the aerospace field. Many advances were made in CRT technology including the development of the color CRT. At the same time, electromechanical displays were being replaced by the newer electronic displays using a variety of display technologies. These two paths began to merge as CRTs were used increasingly to integrate the cockpit information into a common display. The use of the CRT in the commercial transport cockpit has become the "norm" for information display, replacing many discrete displays. The fact that present generation cockpits are being referred to as "glass cockpits" confirms this trend.

The CRT dominates the arena of high resolution video display devices. It is capable of eye-pleasing animated displays and also well suited for alphanumeric and graphic displays. Advantages of the CRT over other technologies for high-speed, high-resolution displays are many.

- The CRT response time (bandwidth) and resolution satisfy viewer requirements for high quality dynamic image presentation.
- Scanning techniques are simple to implement. Scanning is also quite versatile, lending itself to raster, stroke, and random scanning.
- The CRT itself has few parts and a correspondingly small number of electrical connections.
- The CRT is versatile in its application. The same CRT may be used for alphanumerics as well as graphic information.
- Luminous efficiencies associated with CRTs are high when compared to other technologies.
- The CRT is easily mass-produced.

faceplate upon which the phosphor for the viewing screen is deposited. Modifications may be made to the face of the tube for safety reasons, such as implosion protection and emission reduction.

The electron gun supplies an intense beam of electrons which is focused and modulated along the path to the faceplate. Focusing can be accomplished either electrostatically or magnetically. The deflection mechanism is used to direct the electron beam, originating in the electron gun, to a particular location on the faceplate. Either electrostatic or magnetic deflection can be used. When electrons leave the deflection region, they enter the drift region. There are no magnetic or electric fields acting on the beam at this point, but the entire region is at the high voltage level of the screen. Electrons travel across this region and arrive focused on a spot on the phosphor screen.

The faceplate is the focal point for the beam of electrons from the electron gun. These electrons strike the phosphor particles coated on the faceplate causing light to be emitted. A thin film of aluminum is deposited over the phosphor and is used to establish the proper screen voltage.

The electron beam, which is generated by the electron gun when the proper voltages exist in the CRT, is accelerated, focused, and directed to various positions on the screen. As movement of the beam over the screen area occurs, the video information modulates the electron beam intensity in synchronization with the deflection mechanism. One of the grids of the electron gun is typically used for control of the beam intensity.

Due to the nature of CRT technology, continual refresh of the CRT screen is required. Typically this occurs 50 or more times per second and is driven by the requirement to present dynamic information and to eliminate the sensation of flicker to the viewer. This refresh rate depends on the characteristics of the phosphor used and the type of information being displayed.

### 4.2.3 Color Cathode Ray Tube

By modification of the basic elements of the CRT, color operation becomes possible. Several methods for color production exist. These are the mask, beam index, and penetration phosphor methods.

There are three main differences in the construction of a color CRT. These are the addition of two more electron guns, the addition of a mask between the guns and the screen, and the use of a specially coated screen.

### 4.2.4 Shadow-Mask Cathode Ray Tube

One particular mask technique is the shadow-mask. The basic principle of the mask technique can be seen by an examination of figure 4.2-2. This uses a metal mask placed between the three guns and the screen. The screen phosphor consists of sets of three dots arranged in a delta configuration. The electron beam of each gun passes through a particular hole in the mask at any given instant. Each beam then strikes the screen on its associated phosphor dot. Due to the construction, the beam from one gun can only fall on one colored dot in the set of three. Hence, by feeding each gun with its independent signal, the three

must be taken in the CRT design to ensure that vibrations do not translate into movement of the shadow-mask.

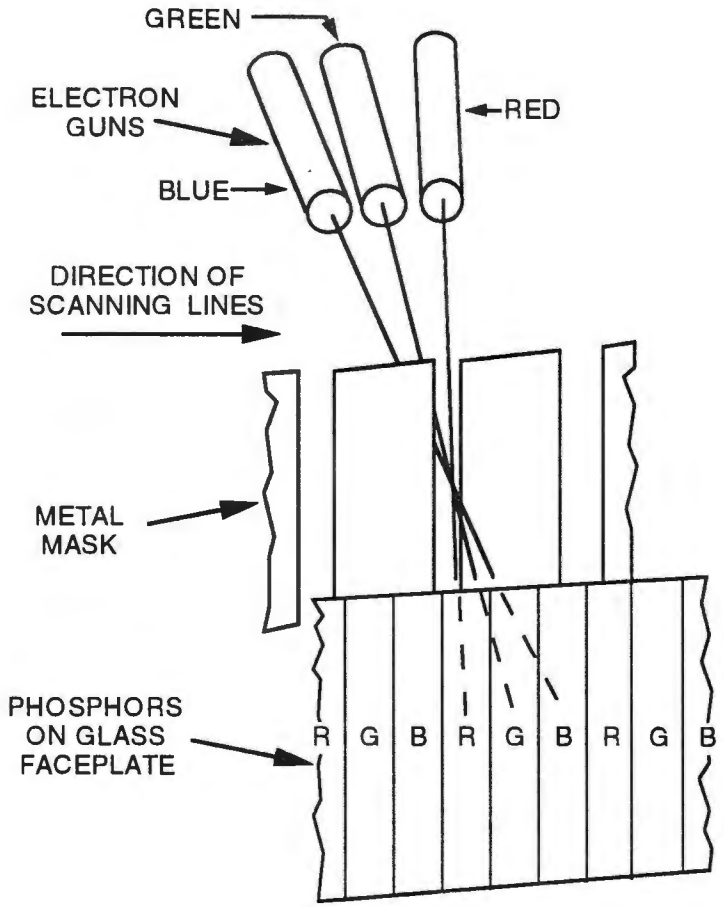



FIGURE 4.2-3. OPERATING PRINCIPLE OF TRINITRON COLOR CATHODE RAY TUBE (Tannas 1985)

The effect of any mask movement would be misalignment in relation to the phosphor screen. The misalignment would then result in color impurity on the CRT screen.

4.2.5 Beam Index Cathode Ray Tube

This CRT avoids the problems associated with the use of the shadow-mask. It uses vertical stripes of phosphor on the screen with no mask between the gun and the screen. Indexing stripes are used on every fourth position between each set of three color stripes. A single electron beam, capable of being focused on the width of one phosphor element, is used. The indexing stripes are part of a control loop. When the beam falls on the indexing position, emission from this element is detected and used as a synchronizing signal for the deflection circuit. The beam current modulator is synchronized to the scan deflection

of phosphor is applied. A thin aluminum film is then deposited as the last layer. The aluminum film provides the acceleration potential for the electron gun.

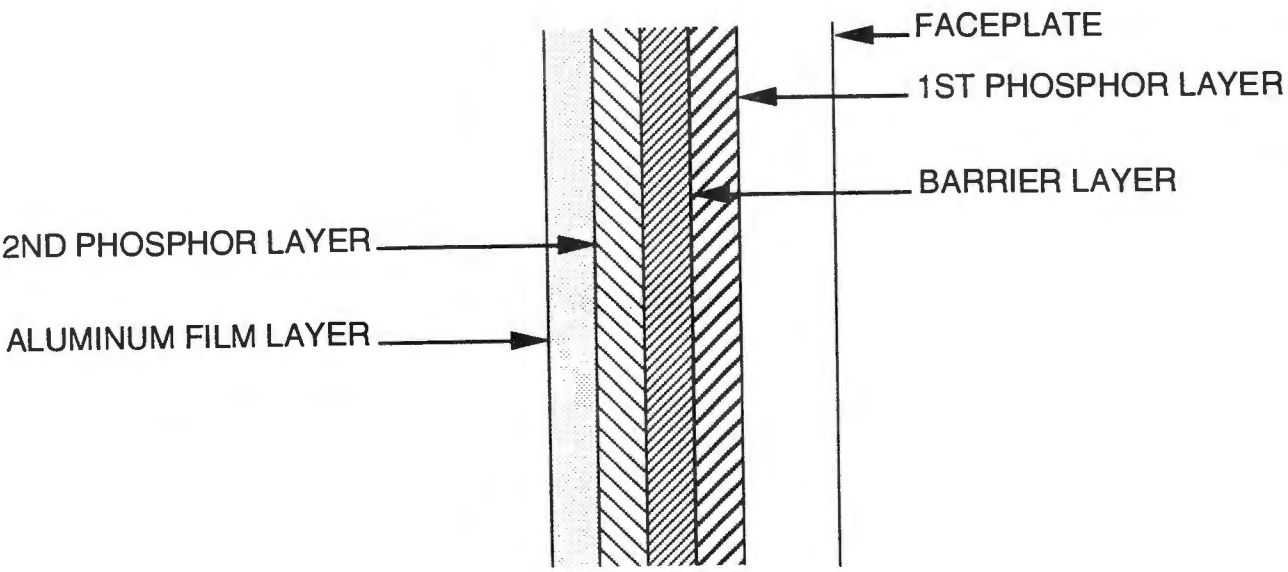



FIGURE 4.2-5. CONSTRUCTION OF PENETRATION PHOSPHOR CATHODE RAY TUBE

With a low potential applied to the aluminum film layer, the electrons are absorbed in the aluminum and do not pass through to the phosphor layers. When the potential is increased, electrons pass through the aluminum and enter the second layer of phosphor where they are absorbed, causing luminescence of that layer. As the potential is increased further, electrons are partially absorbed by the second layer, while the remainder pass through this layer to the first layer. This causes the color generated to be a mixture of the characteristics of the two types of phosphors used. At the highest applied potential, the beam energy becomes sufficient to pass completely through the second layer and is entirely absorbed in the first layer which is deposited on the screen. The CRT now emits light based on the characteristics of the phosphor of this first layer.

The penetration phosphor CRT has the advantage of rugged mechanical construction and therefore is suitable for environments with a high amount of vibration. High resolution is another advantage of this CRT. Resolution is limited by the size of the electron beam width (Michel 1983).

Several problems are characteristic of the penetration phosphor CRTs. The luminous efficiency of the phosphors used in the penetration phosphor CRT is greatly reduced when compared to their use in a monochrome CRT. This is because the second phosphor layer is operated at reduced efficiency and a significant

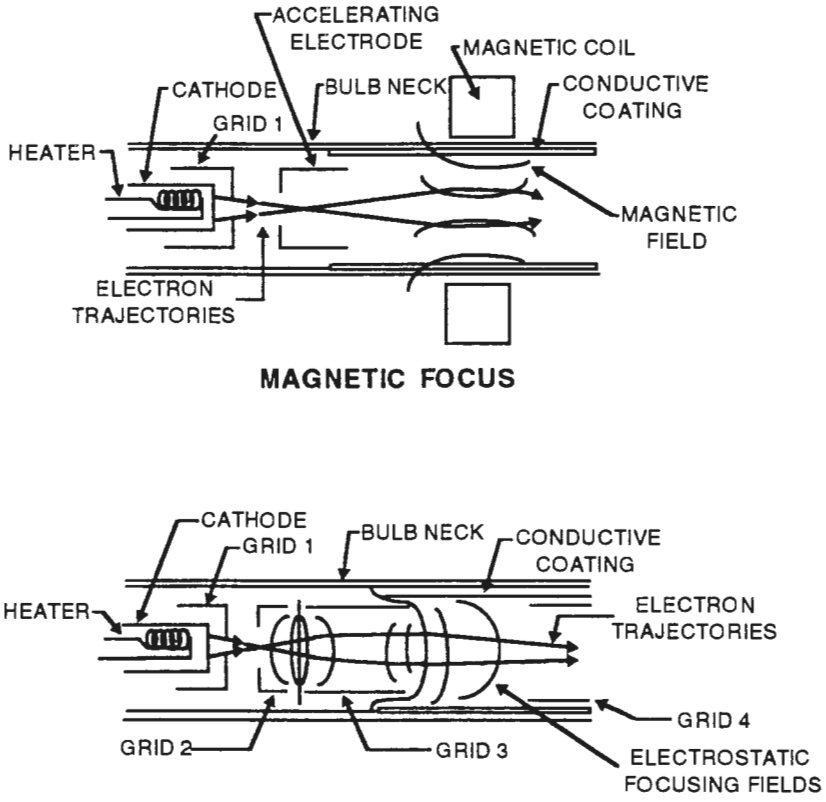



FIGURE 4.2-7. MAGNETIC AND ELECTROSTATIC FOCUSING METHODS (Tannas 1985)

electrodes and the magnitude of the applied voltages are used to focus the beam on the screen.

For bipotential focusing, 15 to 25 percent of the screen voltage is used for focusing. The focusing mechanism has two elements, one element between the g2 electrode and the focus electrode, grid 3 (g3), and the other located between focus electrode and grid 4 (g4) electrode which is maintained at the viewing screen voltage. Due to the high potential of the focusing electrode, the problems associated with grid currents may need to be addressed. With this focusing method, however, there is a 30 to 50 percent resolution improvement over the unipotential focusing lens.

With unipotential focusing the focus elements on both sides of the focus electrode are at the same voltage. A much lower focusing voltage is required.

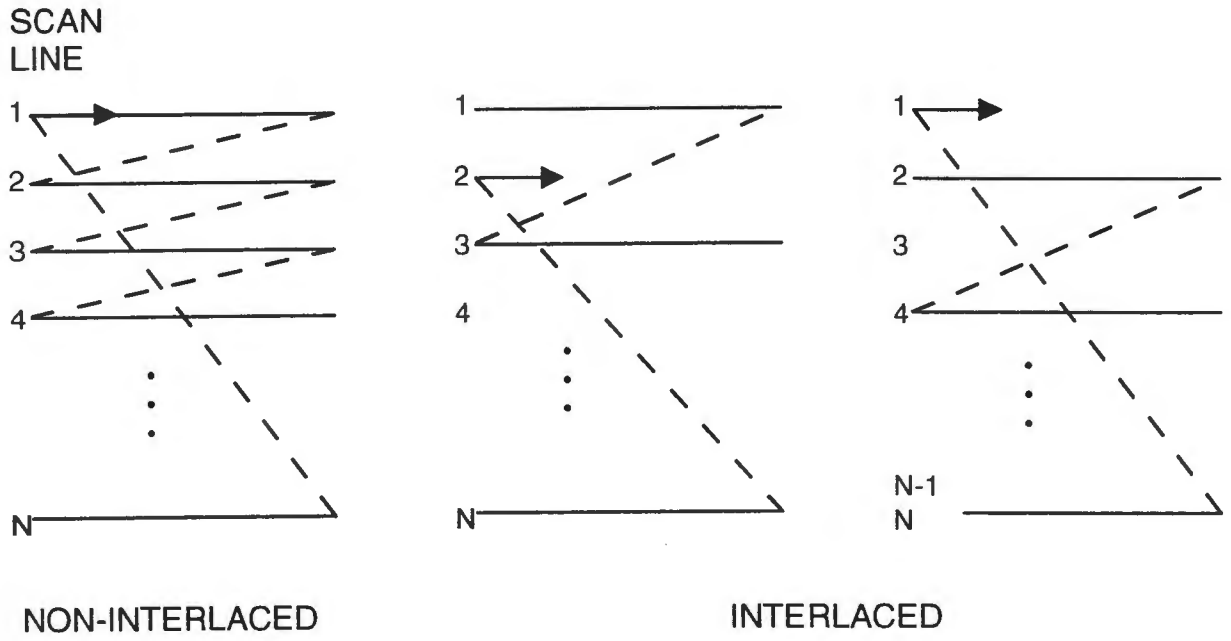



FIGURE 4.2-8. RASTER SWEEP PATTERN

4.2.6.5 The Cathode Ray Tube Driver Circuit

CRTs have associated analog circuitry supporting the many functions required for operation. Typical circuits include the horizontal and vertical deflection, focus, high and low voltage, and the video and blanking circuits. The horizontal and vertical deflection circuits amplify the input signal and drive the associated magnetic or electrostatic deflection devices in synchronization with the blanking circuit. The blanking circuit, which normally is connected at the cathode, turns off the electron beam during beam retrace times.

The filament and grid bias are both part of the low voltage CRT circuitry. Since the cathode and filament are in close proximity, the voltage difference between the two is a critical design constraint. A short between these two supplies can lead to total failure of the CRT. The g2 supply is also considered a low voltage supply, typically being several hundred to under one thousand volts (V). Sometimes the g2 and focus supplies are combined and are both operated at the same voltage.

The high voltage supply is used for the viewing screen and sometimes the focusing circuit. Since high voltages have an increasing tendency to arc under conditions of high moisture and dust and thus draw heavier currents, a limiting resistor in

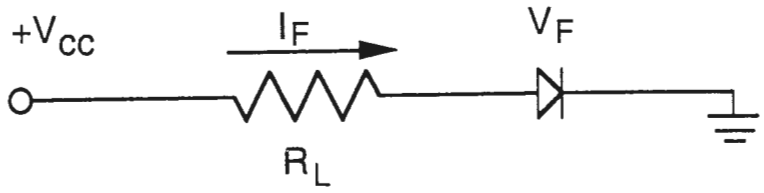



FIGURE 4.2-9. SCHEMATIC DIAGRAM OF LIGHT EMITTING DIODE

The values of $V_F$ and $I_F$ are found on the respective data sheets for the particular LED device. Using typical values of 5 volts for $V_{CC}$, 1.6 volts for $V_F$ and 20 mA for $I_F$ yields a value of 170 ohms for $R_L$.

Variations in $R_L$ cause variations in the current flowing through the LED. Figure 4.2-10 shows how these variations in current can affect the luminous intensity of a typical Gallium Arsenide Phosphide (GaAsP) LED.

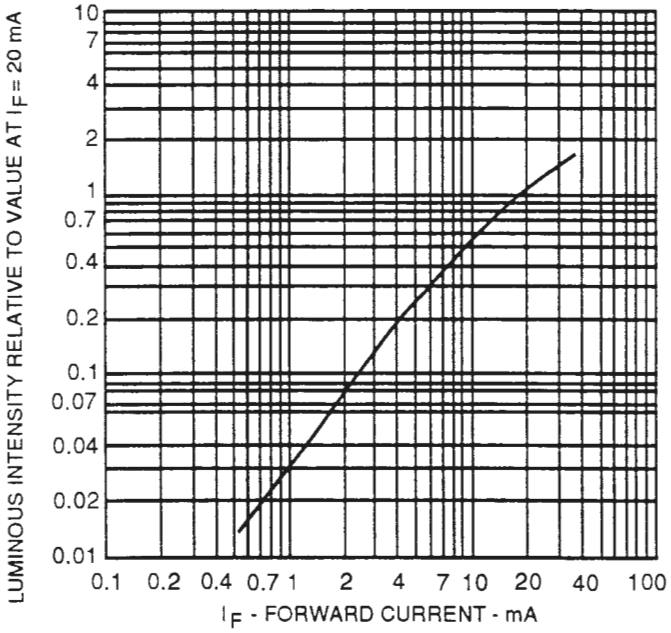



FIGURE 4.2-10. RELATIVE LUMINOUS INTENSITY VERSUS FORWARD CURRENT (TI Optoelectronics Data Book 1984)

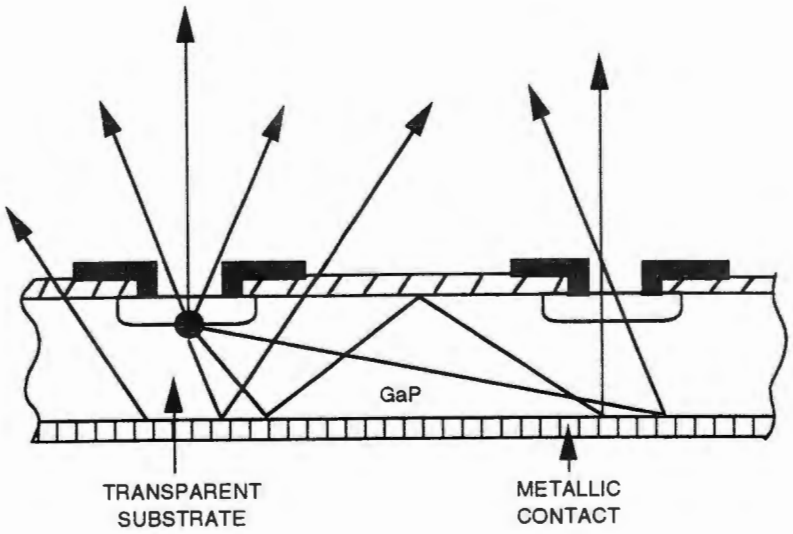



FIGURE 4.2-11. LIGHT EMITTING DIODE OPTICAL CROSSTALK
(Tannas 1985)

What is specified for a device is the absolute maximum rating at a given temperature, normally 25°C. The first four items in the list specify the operational parameters while the last two are concerned with the storage and manufacturing environment. The maximum lead temperature is specified to prevent device destruction in the soldering process. It is given as the maximum lead temperature at x distance from the device base for y seconds.

If the current density of an LED is exceeded, excessive junction temperatures will be produced. This can lead to permanently altered luminance/current density characteristics or complete device destruction. Also, exceeding the reverse breakdown voltage by a sufficient amount will destroy the device. Reverse voltages of five volts or less will destroy most LEDs (Catalog of Optoelectronic Products, 1985). Proper design will reduce the probability of any of these failures.

Display flicker can be a problem for designs incorporating LEDs. Other display technologies that require periodic refresh also must deal with this problem. There is a frequency, known as the fusion frequency, above which the eye no longer perceives this refresh effect. Display designers need to account for this as well as other factors that may have an effect, such as vibration of the aircraft.

Disadvantages of LED display technology include high power consumption (compared with passive displays), and high peak currents for long digit strings. There is

property maintains the alignment of the LC in the absence of an applied electric field.

The LC is enclosed by flat glass substrates with uniform spacing. The LC material should be non-toxic and stable in any possible operating environment. On the inside of the glass the electrode patterns are formed. If the display is transmissive, a thin layer of transparent conductive material is used. For a reflective display, a metallic deposit may be used as the rear electrode.

Temperature has a great effect on the LC material. At low temperatures, the material may change into another LC phase, or even into a solid. Response times of the material decrease dramatically with lower temperatures. Figure 4.2-12 shows the relationship of temperature to the turn-on and turn-off times of a Twisted Nematic (TN) display cell. This effect necessitates the use of a heater in low temperature operating environments. At high temperatures the material becomes an isotropic liquid and the display properties are lost.

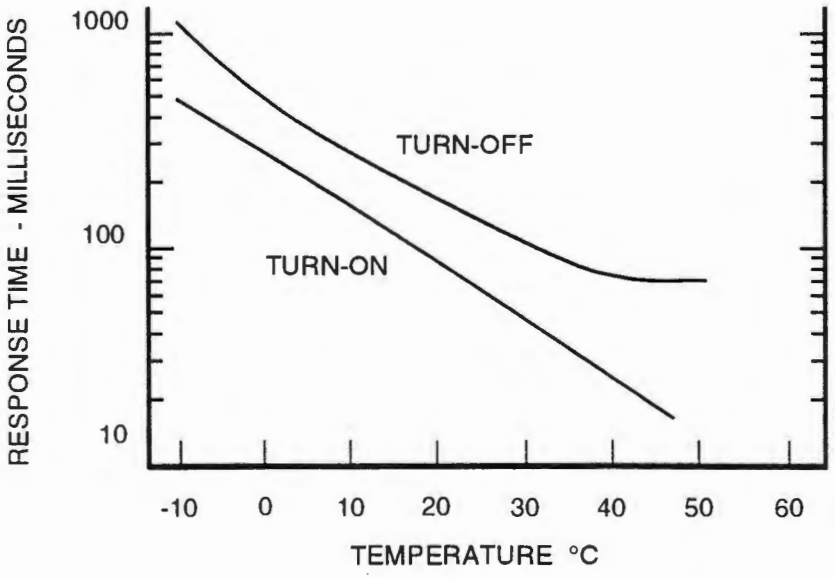



FIGURE 4.2-12. RESPONSE TIMES FOR A TWISTED NEMATIC LIQUID CRYSTAL CELL (Hughes 1983)

The better LC materials exhibit an impedance of more than $10^7$ ohms per square cm in parallel with a capacitance of 1000 to 3000 picofarads per square cm. The resistivity is generally greater than $10^{10}$ ohm-cm. Due to these properties, the power dissipation is limited to several microwatts per square cm (Hughes, 1983). Power consumption of LC displays is much less than other popular display technologies. Power supply requirements are lower, display volume is smaller

The advantages and disadvantages of the LC displays are summarized below:

Advantages of LC displays:

- Does not emit light and does not suffer washout from high ambient light as other displays do.
- Requires much less power to operate and hence uses smaller, lighter, less expensive power supplies. LC is the display of choice for battery operation.
- Operates on low voltage and is compatible with digital drivers.
- Has a long life expectancy.
- Is cost competitive.
- Can be viewed directly or projected.

Disadvantages of LC displays:

- The LC phase exists for only a limited temperature range. Heating of the display is required in cold environments.
- The display response is slow at lower temperatures.
- The most common method of LC display, the TN effect, requires polarizers which limit the display brightness.
- The TN LC displays require the use of matrix addressing with a limited number of lines. Active element addressing is required for complex displays (Hughes 1983).

4.2.9.4 Active-Matrix Liquid Crystal Display

The Active-Matrix (AM) LC technology has evolved considerably in recent years. For the AM cell, an active device is placed at each pixel, which is defined by the crossing of a row and a column line. Typically, a Thin-Film Transistor (TFT) as well as a capacitor are used. Figure 4.2-13 shows a cross sectional view of the TFT LC cell.

The polarizers are used for TN operation of the display. A layer of glass encloses and seals the front and back of the display. Proper uniform spacing is maintained between the two glass layers by spacer rods. Light entering from the back of the display passes through the polarizer, glass, and the transparent TFT matrix circuit. Light which passes through an enabled LC cell then passes through the color filter, glass, and front polarizer of the display. The TFT output electrode is composed of Indium Tin Oxide (ITO) which acts as a transparent electrical conductor.

In this design, the TFT and associated circuitry are deposited directly on a glass substrate in a vacuum chamber. This sometimes yields short circuits

should be small enough so that the appearance of the display is not affected. For panels that use bi-level voltages, the pixel gate voltage in the OFF state should not exceed the threshold level and turn the pixel on. Also, the pixel gate voltage in the ON state should not decay to a level where it may appear partially on or off. The charge on the LC capacitor may also leak through the LC material. In order to reduce the effect of this leakage, a capacitor may be added to the output electrode. This in turn will increase the circuit complexity and cost, and reduce the yield (Luo 1990).

A color display requires proper alignment of the color filter with the pixels in the TFT array. Different types of color filters have been developed for this application, but the most widely used is the dyed gelatin color filter. This is used to form a layer between 1 to 2 $\mu$m in thickness. It is dyed three separate times for the red, green, and blue colors (Luo 1990). The color pixels are fabricated in patterns that correspond to their respective TFT electrodes.

An additional, but important part of the AM LC display is the backlighting device. An efficient, uniform light of highly variable output is needed for this display. An efficient lamp will produce more light with less heat. Uniform output is required so that the display does not appear brighter in one area than another. A variable output is required so that the display may be viewed in direct sunlight as well as during night flying. A florescent lamp with peak output at the red, green, and blue wavelengths is commonly used for this purpose. The overall display efficiency is enhanced when the spectrum of the backlighting source is matched with that of the color filter.

As with any LC device, operation at low temperatures requires special consideration. Response times of a LC cell progressively decrease with temperature and are in the range of 1 to 2 seconds at -40°C. Operation in cold environments requires the use of an integral heater for satisfactory response times.

The AM LC display technology is seen as the one that will replace the CRT in aircraft cockpits. Some military cockpits are already using this technology, with devices developed by Optical Imaging Systems. Both new designs and retrofits of existing aircraft are planned. The same trend is occurring in the commercial transport arena with Boeing planning to use AM LC for the primary displays of the Boeing 777. If the manufacturing cost for AM LC approaches the cost of the CRT, it is anticipated that the AM LC display will completely eliminate the CRT in the cockpit.

A display device in development which marries the technologies of both the CRT and LC is the Liquid Crystal Shutter (LCS) from Tektronix. This device has a LC shutter placed over the face of a single electron gun CRT. Color is generated by switching a series of polarizing filters on and off during the normal screen refresh for each of the three primary colors. The color field rate is maintained at 180 Hz so that all three colors are updated at 60 Hz. The viewer's eye integrates each color frame and hence sees the full color images.

The LCS is designed for military aircraft and flight simulators. Since it uses a monochrome, high intensity CRT, the beam convergence problems of multi-gun CRTs are non-existent. In addition, this display benefits from the LC characteristics of high contrast and sunlight readability.

about 1 cm of additional length and width. If driver ICs are added to the assembly, then additional depth may be required. The thickness of the display panels is generally less than 1.25 cm (Gurman 1983).

4.2.10.1 Reliability

Reliability depends on the quality of material used for the phosphor and that of the dielectric layers. Materials used need to be free from contaminants. The TFEL structure can be very sensitive to moisture, and requires care in the sealing process. Pin holes in dielectric can cause local burn-out damage due to high conduction currents there. This damage however is self-limiting, generating an open-circuit at that point and preventing further damage.

4.2.10.2 Failure Modes

The TFEL display acts as a capacitor due to its construction. In the active display area a high voltage rapidly charges the capacitor plates (the dielectric layers). If breakdown occurs, a high current flow will result, with the possibility of device failure due to the high amount of heat generated at this breakdown point. The larger the dielectric area is, the greater the amount of energy transferred into the breakdown area.

The overall efficiency of the TFEL display depends not only on the composition of the phosphor material, but also on the display driver circuitry. The row and column drivers are required to switch high voltages in short times, especially when using matrix addressing. This results in inefficiencies due to the largely capacitive loading that the TFEL cells present.

Most displays operate in the binary mode with pixels either completely on or completely off. Graphic and video displays can be produced by using a driver system capable of producing gray shades on the display. The TFEL display emits a short burst of light upon application of the high voltage waveform. Before the next pulse can occur, a pulse of the opposite polarity must be applied to the TFEL cell. Essentially, the device requires an ac waveform. It is possible to control the luminance by varying the frequency at which the display is operating (Gurman 1983).

Display flicker can occur at low refresh rates or under any other condition that may produce a stroboscopic effect, such as high frequency vibration. High resolution displays can be achieved with EL technology.

Color can be achieved with TFEL technology by the use of color activators in the phosphor material. The generation of blue is the most difficult due to the low luminance level and the low sensitivity of the eye to this wavelength.

Due to the transparency of the TFEL, it is possible to produce color displays without the loss of resolution characteristic of most other display technologies. Other display technologies, such as the shadow mask CRT sacrifice resolution when color areas are added to the display screen. With TFEL it is possible to place the different color cells in series without sacrificing the resolution.

4.2.12.1 AC Plasma Displays

Figure 4.2-15 shows the structure of an ac plasma display panel. The outside consists of two flat glass plates separated by a small distance. On each glass plate there are rows of conductors, the conductors of one plate being orthogonal to the conductors of the other plate. These conductors are insulated from the gas by a dielectric layer. The gas consists typically of a neon-argon mixture which emits an orange glow when excited. Display cells exist at the intersections of the rows and columns of conductors.

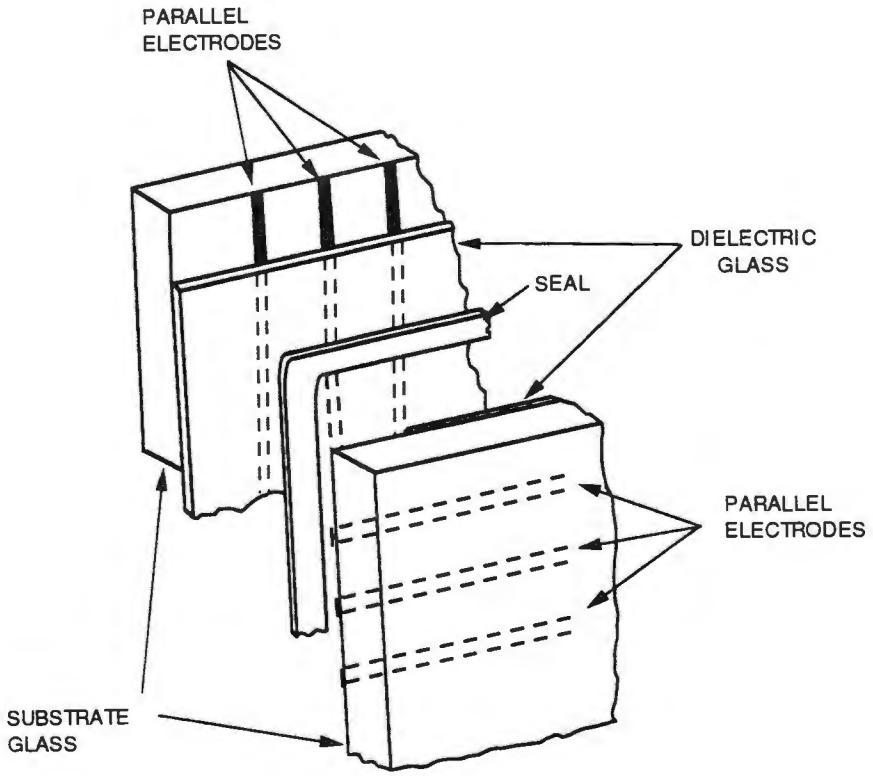



FIGURE 4.2-15. AC PLASMA DISPLAY PANEL STRUCTURE (Tannas 1985)

For normal operation, a positive and negative continuously pulsed sustaining voltage, similar to an ac waveform, is applied to all rows and columns. The positive and negative peaks do not exceed $V_B$. Hence, with only this signal applied, the display is blank. The gas in a particular cell discharges and emits light when a short duration pulse is generated in phase and exceeding the value of $V_B$, which is greater than the normally applied ac waveform.

The thickness of the dc plasma display is the same as the ac display. It is difficult to obtain high resolution displays using dc plasma due to problems such as cross-talk between adjacent cells. However, sharp images due to the fabrication technique, are characteristic of this technology. Hence, the commercial displays which use this technology are typically numeric or alphanumeric.

The emission from the dc plasma display is a characteristic neon orange glow. The refresh rate of display panels using this technology needs to be chosen carefully to avoid the appearance of flicker.

Low temperature operation of this type of display should be avoided for devices containing mercury (Hg). Hg vapor is used in the gas mixture to avoid sputtering, but the operating life may be impaired when the display is used at low temperatures.

High voltage drivers are necessary for the dc plasma display as well. Variation of the display intensity is accomplished by modulating the duty cycle of the high voltage drivers which perform the refresh. As with some of the other display technologies, a trade-off exists between the overall display brightness and the refresh method. If the duty cycle for each character is reduced to save on the overall supply current, then the display brightness is correspondingly reduced. This may happen if too many characters are refreshed in a particular scan cycle in order to reduce the driver hardware requirements.

## 4.3 Display Applications

### 4.3.1 Head-Down Display

The HDD is the primary type of display used on modern commercial transport aircraft. It is generally a direct view, computer driven CRT display with capabilities for graphic and video images. HDDs are normally designated to have a single function, such as display of primary flight data or weather radar data, but with the throw of a switch, data on the displays can be swapped around. This allows for maintaining the PFD data in case the normal display CRT malfunctions.

Examples of HDDs for various commercial transports are seen in section 2. Modern transports such as the Airbus 320 and McDonnell-Douglas MD-11 use multiple CRTs across the cockpit, replacing older electromechanical indicators. Older aircraft are often upgraded with glass cockpits along with the removal of a number of the electromechanical indicators.

### 4.3.2 Head-Up Display

SAE ARP4102/8 states "HUD systems shall provide the pilot with a display that enables him to assess aircraft attitude and flight path in relation to and in association with external visual cues." The primary use of the HUD is to aid the pilot during approach, landing, and takeoff. This is done by projecting the primary flight information on the windshield, giving the pilot access to this information without a need to look down in the cockpit during this critical time. The familiar "T" arrangement should be used as the basic data format. A HUD is

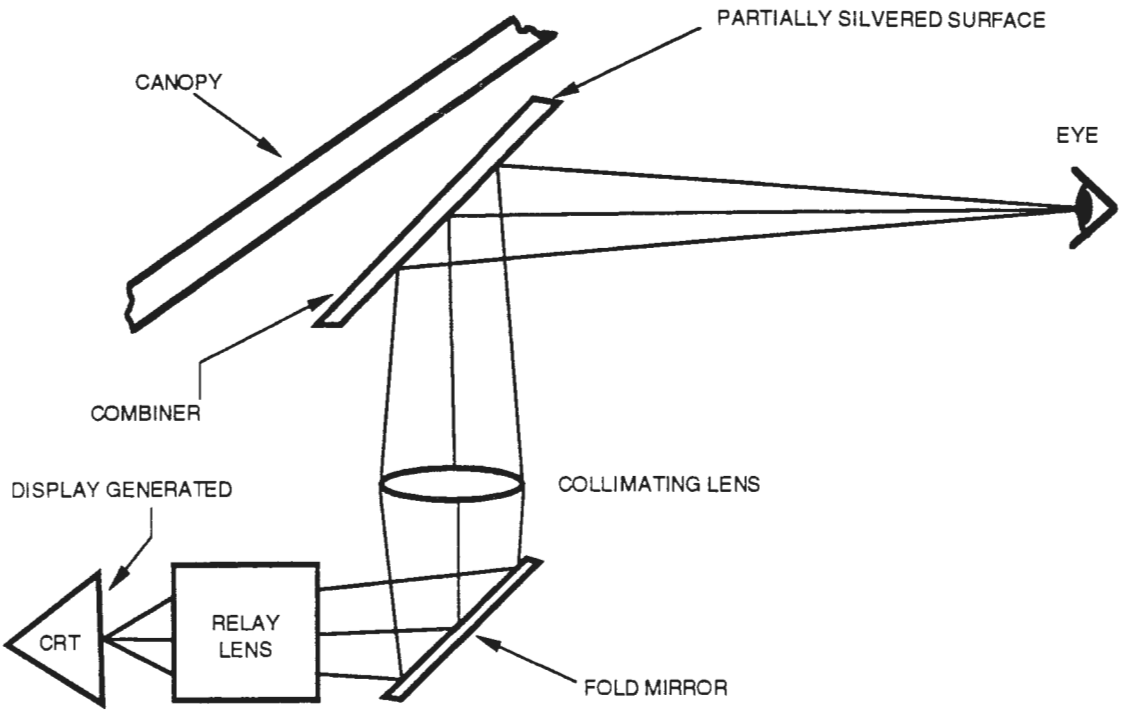



FIGURE 4.3-1. BASIC HEAD-UP DISPLAY COMPONENTS
(Pinkus and Task 1988)

4.3.3 Head-Up Display Failures

SAE ARP4102/8 specifies that the probability of hazardous or misleading information being presented by the HUD without warning or blanking should be extremely remote. Displayed parameters that lose their validity should be removed from the display by the controlling software. In addition any failure of system computation or monitors should result in the complete blanking of the display, along with annunciation of this fact.

4.3.4 Head-Level Display

A product which uses a combination of a HUD and a Head-Level Display (HLD) in a single package has been developed by Sextant Avionique. The HLD is mounted directly below and in line with the HUD. It is a collimated display to allow easy transition between the HUD and HLD.

The HLD presents more information to the pilot by displaying HUD related or synthetic images. It has the potential of eliminating HUD clutter by a splitting

the latter technology which remains to be perfected before reliable and convenient use can be made of this technology in the cockpit.

Some of the disadvantages of speech technology are that:

- Various physical conditions can alter speech characteristics.
- Various psychological conditions can alter speech characteristics.
- Other audio range emissions can interfere with the desired source.
- Speech synthesis may interfere with other aural indicators.

The use of speech technology holds profound benefits in a well-designed system. It can be a cost-effective man-machine interface. It can reduce manpower requirements. It can reduce visual information overload and workload as well as permitting operator mobility. In addition, speech technology works in a completely dark environment and frees the eyes and hands for other tasks.

### 4.4.1 Voice Synthesis

Humans produce sound when air from trachea passes by the vocal cords. The sounds are modified by the cavities of the throat, mouth, and nose, which act as audio filters and therefore determine the waveform of the produced sound. Speech generating hardware should model these characteristics faithfully to produce speech that sounds human, rather than poor quality, mechanical sounding speech.

There are three methods for digital speech generation. They are digitized, word-generated synthesized, and phoneme-generated synthesized.

Digitized speech is reproduced from a digitally recorded voice. If the input is sampled at a sufficiently high rate, then the voice that is reproduced will sound very natural. A problem with this technique is that large amounts of memory are required to store the digitized speech.

Another form of digital speech generation is synthesized, word generated speech. This technique uses a compressed pattern produced from the original speech input. This method inherently introduces some distortion in the reproduced signal, and sound reproduction is less natural. The amount of compression applied to the input speech generally determines the amount of distortion in the reproduced waveform.

The last method of digital speech generation is synthesized, phoneme generated speech. Phonemes are the smallest elements of speech. Combinations of phonemes are put together to form any word of our vocabulary. Since the number of phonemes used to represent the set of different sounds is small, the amount of digital memory used for speech generation is not a problem.

Speech is generated from text messages which exist in memory in a table format. The messages are pre-programmed sequences of phonemes which are combined to form words. Each byte from a particular selected table is presented to the phoneme generator in sequence. The generator then produces the sound associated with the

Other difficulties are encountered in voice recognition. Since speech is continuous, there may be no clearly defined boundaries between spoken words. The recognizer needs to be able to pick out the individual words or else the speaker needs to modify the normal manner of speaking by distinctly pausing between each spoken word. Ambiguity can be a problem, since there is no acoustic difference between some words, such as "to" and "two". Words need to be interpreted in their context or the vocabulary carefully chosen and limited. Voice recognizers must not only recognize the correct words, but also must reject the incorrect ones.

The percent of correct words for a word recognition device is an important rating for a voice recognition system. Any input errors, which may be caused by the recognition device or the operator, require a method of correction. Feedback of the matched word can be performed using an alphanumeric display device to ensure proper entry. If an incorrect word is displayed, then the system user also needs to be able to reject that word by voice command.

Voice input is not viewed as a viable method of aircraft systems control for the modern commercial transport. Improvements are necessary in the areas of phonetic accuracy, vocabulary size, response time of the recognition system, and performance in high noise environments. In addition, other human factors issues associated with the use of voice technology in the cockpit need to be addressed.

### 4.4.3 Voice Technology Applications

The designer's task for a voice synthesis system is greatly simplified by the availability of Large Scale Integration (LSI). Manufacturers produce LSI devices that contain the many functions required for generation of high fidelity speech. Single LSI devices, like the Texas Instruments TSP50C14 IC, contain functions such as CPU, digital filters, Read-Only-Memory (ROM), clock oscillator, digital/analog converter, and audio amplifier.

Systems that include voice synthesis in the commercial cockpit include the Ground Proximity Warning as well as a checklist management system. Heads Up Technologies manufactures a digital speech annunciator which combines synthetically generated audio with an annunciator panel. It also can be connected as a closed loop system to certain critical systems for checking take-off configuration, giving an aural warning if not correct. This particular device is planned for use on the Citation jet and is used in certain military aircraft such as the C-130.

## 4.5 Keyboard Technology

Prior to the introduction of the keyboard into the cockpit, the interface to the numerous avionic systems was through discrete dials and switches. When the number of dials and switches grew beyond what was deemed reasonable, keyboards were introduced. The keyboard is now the normal method of interacting with modern and complex digital avionic systems. For this reason, keyboards are ubiquitous in modern transport cockpits. Other input devices such as discrete knobs and switches are used in the cockpit. Capacitive touch pads and trackballs will appear in the future.

# 5. HUMAN FACTORS AND THE PILOT-VEHICLE INTERFACE

## 5.1 Introduction

Human Factors and the PVI consider the capabilities of pilot performance in relation to the design of the cockpit environment. The task of designing digital systems for the cockpit is monumental. Adding the pilot to this environment presents an important factor for designers to consider. The capabilities, limitations, and physical stereotypes of the pilot population must be a vital part of the system design.

Although a considerable amount of information is known about how pilots interface with their aircraft, a great deal remains to be discovered. This section examines a number of human factors issues related to the PVI. Research germane to these areas is presented. One valuable resource for this endeavor has been the FAA National Plan for Aviation Human Factors.

### 5.1.1 The National Plan for Aviation Human Factors

In November 1988 the U.S. Congress enacted "The Aviation Safety Research Act of 1988" (Public Law 100-591). This law directs the FAA to augment its research efforts in human factors and to coordinate its efforts with those of National Aeronautics and Space Administration (NASA). The National Plan for Aviation Human Factors represents the first step of an intensive effort to coordinate human factors-related research in aviation.

The National Plan has been a major resource for identifying human factors issues and problems concerning human factors and the PVI.

### 5.1.2 Scope and Organization

The field of human factors in aviation is broad. Although the pilot interacts with the crew, the systems, the aircraft, and the airspace environment, the major focus of this report is on the pilot and the cockpit interface. Of necessity, the scope of this section is limited to the relationship between the PVI and safety, cockpit automation, design issues, crew workload, Crew Resource Management (CRM), and training.

For each area, an overview of significant research findings will be presented. This information is culled from representative current studies, professional journal articles, conference proceedings, textbooks, government documents, interviews with experts in the field, and incident and accident reports. Additional information can be obtained from the technical report, "Pilot-Vehicle Interface" (Harrison, Janowitz, and Castronuovo 1993), and from the resources listed in the bibliography of this chapter.

- Peer pressure.

### 5.2.3 Safety and Situational Awareness

Maintaining situational awareness is crucial to aircraft safety. Although one goal of automation is to help the pilot maintain situational awareness, automation also is cited as a major cause of accidents and incidents. Problems can result from inappropriate application of automation technologies either by the designer or the user, rather than from "overautomation."

Loss of situational awareness may occur as a result of a chain of events, or an error chain. The error chain is a concept that describes human error accidents as being made up of a series of errors. Links are identifiable by means of specific clues. By breaking any one link, a flight crew potentially can break the chain and prevent an accident. Analysis of a number of accidents indicates that at least five clues are present and identifiable in most human error accidents. Links in an error chain tend to occur sequentially, may or may not be related to each other, and may not be readily visible to the crew. Clues to loss of situational awareness include:

- Ambiguity – two or more independent sources of information differ,
- Fixation or preoccupation – focus of attention on any one item or event to the exclusion of all others,
- Confusion – a sense of uncertainty or anxiety about a particular situation – falling behind the aircraft,
- No one flying the aircraft,
- No one looking out of the window,
- Use of an undocumented procedure,
- Violating minimums,
- Failure to meet targets, and
- Departure from SOPs.

Barriers to situational awareness in the cockpit include complacency and pilots "falling out of the loop." Strategies to prevent this are training for wise use of the equipment and installing error-evident displays (Wiener 1990).

### 5.2.4 Digital Systems/Aircraft Incidents, Accidents, and Lessons Learned

Through accident and incident investigation, information can be gained to increase aviation safety. Digital systems/cockpits present new investigation requirements. Automated cockpits are changing the types of data needed to perform accident investigation work. Some proposed changes include the addition of video recorders on the flight deck to tape what the electronic displays are showing pilots, and more parameters to be stored in flight data recorders.

While flight deck automation strives to reduce flight crew workload, many incidents were related to workload. Some typical comments include:

- "On a two-man high performance aircraft, automation is supposed to relieve the pilots of a high workload, but when even minor problems occur, one's attention can be greatly diverted. I believe that in our present day two-man aircraft, one pilot is routinely taken out of the loop by even ordinary changes necessary in our high density environment."

- "...we did not fly the airplane first and program the FMC second, we relied too much on the FMCs in a situation where they require too much input and monitoring and increase the workload."

- "In a two pilot environment in which what was formerly the second officer/flight engineer's functions are now totally automated, an apparent failure of the automation is particularly distracting to the captain and first officer. The crew member flying becomes immediately absorbed in determining which flight instruments are reliable while the remaining crew member seeks the source of the problem. This results in a brief interval when the heading and altitude are of secondary concern. Stabilized flight is first. ...A two pilot crew concept works great, but only as long as the automatic black box items which have replaced the second officer are feeding the captain and first officer accurate information."

- "...I was so busy watching for traffic, monitoring our departure, trying to sort out electrical/electronic problems, talk on the radio, realize my error, safely correct it, watch for reported traffic, climb, etc., that by the time I was ready to tell them about our problems, they were already giving the phone number...A two-man cockpit on a short leg in a widebody with electronic problems is too busy."

Inadequate situational awareness can occur through lack of vigilance, complacency, and over-reliance on automated systems. The incident reports reflect this in the frequent use of statements such as the reporter was unsure of why a particular event occurred.

Training concerns were revealed through the report narratives. Reporters mentioned areas that received little attention or were not taught at all:

- "We were attempting to utilize the new full up features of the FMC, but neither of us were proficient in its use. Nor had we been given any hands on training on the new features."

- "...Lack of complete knowledge of the autoflight capabilities and vagaries led to a nonstandard and ultimately unsatisfactory flight procedure."

- "These two-man cockpits have become so complex. ...Perhaps we need more training, or at least an inoperable components training."

Interaction, communication, and a clear delineation of tasks among crew members have been shown to affect flight safety. A number of reporters cited problems in this area.

### 5.3.2 The National Plan Objectives for Automation

The objectives for automation stated in the National Plan for Aviation Human Factors (1990) are:

A. To identify the information required by flight crews to fly commercial air carriers and general aviation aircraft safely in the evolving National Airspace System (NAS) and to develop guidelines for designing and evaluating flight deck controls and displays with regard to their safety and efficiency of use by flight crews.

B. To provide a standard set of assessment procedures and criteria based upon pilot performance and judgement that are useful for designing and evaluating instrument approach procedures.

C. To develop and evaluate advanced flight deck systems technologies that provide flight crews with safe and effective ways and means to plan and replan flights, manage aircraft systems, and effectively respond to the external environment in dealing with contingencies.

D. To provide human centered guidelines for automation technologies that ensure full situation awareness.

E. To develop the technology necessary for the design of error-tolerant flight decks, including intelligent monitoring devices for flight deck procedures and concepts that provide assistance to the flight crew in investigating strategies that satisfy specific goals subject to system and aircraft constraints.

F. To develop automation technology that supports the flight crew in time-critical situations.

G. To develop intelligent flight crew aids that provide extensive fault monitoring and diagnosis capability.

These objectives offer a springboard for current and forthcoming research efforts.

### 5.3.3 Reasons to Automate

Cockpit automation is a topic of much debate. While researchers ponder which tasks should be automated or whether automation already has gone too far, a number of reasons for cockpit automation have emerged:

- Increased safety,
- Error reduction,
- Workload reduction,

#### 5.3.4.2 Design Concerns

Concerns about design pertain to the actual devices or systems and how they operate. These concerns include:

- CRT-displayed legends that vary according to the current mode of operation,
- Multi-function switches that reduce the number of switches in the cockpit, but introduce complexity and the need for a greater amount of training,
- Devices or system designs that limit the pilot's ability to respond when the unexpected occurs,
- FMSs with cumbersome interaction, or where improper programming can introduce error,
- High false alarm rates that lead to failure to respond to genuine alarms,
- Inadequately designed keyboards which result in input errors,
- Automation that reduces the number of small errors but invites new forms of operational errors,
- Premature introduction of equipment,
- Fault intolerant systems,
- Technology that allows silent failures,
- Failures induced by automation, and
- Systems that are too complex and allow pilots to be "surprised" by automatic features.

#### 5.3.4.3 Information Management Concerns

Information management becomes especially critical in glass cockpits where a small number of display devices replace numerous dials and gauges. These concerns were among those expressed in the literature:

- The displayed information may be less accessible. The pilot must learn how to access information.
- Designers rather than pilots may determine what information is essential.
- Too much programming and "heads down" time takes place below 10,000 feet and in the terminal area.
- Lack of situational awareness in identifying and correcting the problem can exist when an automated system fails.

- Ensure that overall system performance will be insensitive to different options, or styles of operation.

- Provide sufficient training for pilots working with automated equipment, to ensure proper operation and set-up, and ensure knowledge of correct operation and malfunction procedures.

- Provide a means for checking the set-up and information input to automatic systems.

- Train and motivate pilots to monitor automated systems.

- Provide meaningful duties to maintain pilot involvement and resistance to distraction if automation reduces task demands to low levels.

- Keep false alarm rates within acceptable limits.

- Clearly indicate which condition is responsible for an alarm display. This is especially critical for multimode systems.

- Provide alarm information in the proper format so that a validity check can be made quickly and accurately and not become a source of distraction.

- Provide the pilot with information and controls to diagnose the automatic system and warning system operation.

- Format the alarm to indicate the degree of emergency.

- Devise training techniques and possibly training hardware to ensure that flight crews are exposed to all forms of alerts and to many of the possible combinations of alerts, and that they understand how to deal with them.

### 5.3.6 The Role of the Pilot and Crew in an Automated Cockpit

With the introduction of cockpit automation, the nature of the pilot's role is changing. Automated systems have assumed some of the pilot's responsibilities while system operation and monitoring responsibilities have been added to the pilot's job.

One fear expressed by the aviation community is that the pilot will be automated "out of the loop" and will be ill-prepared to respond to system failures or emergencies. To help the pilot maintain situational awareness it has been suggested that automated systems be designed for a "human-centered" cockpit. In the human-centered cockpit, automated systems provide maximum assistance while allowing the pilot to remain in control. The system works in concert with the strengths of the human and the machine.

It should be stated, however, that pilots of modern transport aircraft are not "in the loop" in the traditional sense. The degree of sophistication and complexity of newer transport aircraft require some form of computer assistance at all times to fly the aircraft. Factors such as economics and safety ensure

## 5.4 Design Issues and the Pilot-Vehicle Interface

### 5.4.1 Cockpit Design Philosophies

There is a move toward human-centered automation and toward involving pilots and human factors specialists in the design phase of automated systems. Many manufacturers retain a human factors group and qualified pilots as part of automated systems design teams. Since cockpit systems have become so complex, it is impossible for one designer to be responsible for an entire system. The team approach is favored.

Maher (1992) offers general design and operational philosophy recommendations:

- Present information in language that relates to the way pilots think when they fly.
- Design the pilot and crew into the loop during low workload events. Provide antidotes to boredom during long legs of flights. Assist the crew during periods of high workload.
- Airplanes can operate using basic raw data, using the flight director to control the plane, or using full autoflight. Design equipment and training to enable proficiency at each level and simple transitions between levels.
- Apply error checking Artificial Intelligence (AI) programs to all programming functions.
- Do not build hard electronic protective cocoons. They have caused crashes and can deprive pilots of tactile feedback in events such as low level windshear recoveries.
- Use technology to build, maintain, and reinforce teamwork.
- If the aviation system is unwilling to fund electronic glidescope guidance on all runways served by air carriers, build aircraft technology that will generate internal precision glidescopes.

Pilots must have an active role in controlling or managing the systems to which they delegate control of the aircraft. To be involved, the pilot must be informed. The pilot must be able to monitor the automated systems and the systems must be predictable. Air crews must be able to evaluate system performance and detect and recognize deviations from normal operation quickly. Automated systems must be able to monitor the human. Because pilots are fallible as well, automated systems need the capability of monitoring crew performance. Each element of the system must know the other's intentions (Phillips 1992).

### 5.4.2 Cockpit Layout and Devices

The design of control layouts in World War II aircraft was particularly haphazard. This lead to many accidents and incidents caused by misidentification or misoperation of a control. These mishaps spurred research on the placement and design of cockpit instruments.

### 5.4.3 Information Presentation, Organization, and Access

Pilots in a digital cockpit exist in an information-rich environment. The challenge for designers is to organize and present information in a meaningful and accessible manner. During abnormal conditions, pilots do not have the time to select critical information from unimportant information. The task for the system designer is to provide the necessary information in a timely and comprehensible manner while maintaining pilot workload within an acceptable range (Allen 1989).

Williges and Williges (1984) have compiled an extensive review of dialogue design literature. This document was not intended to be a handbook, rather it was an attempt to bring together the variety of guidelines for dialogue design in the form of user considerations. Because relevant user considerations can vary from application to application, some of the considerations may be conflicting. The designer must decide which are appropriate for the specific display application. The review includes an extensive reference list. Findings relevant to cockpit display for the PVI include topics such as:

- Information Coding,
- Information Density,
- Labeling,
- Format,
- Prompts,
- Tabular Data,
- Graphics,
- Textual Data,
- Numeric Data, and
- Screen Layout.

Additional information about these topics is presented in the "Pilot-Vehicle Interface" technical report (Harrison, Janowitz, and Castronuovo 1993).

Stokes and Wickens (1988) cite some advantages and cautions for use of color and color coding in displays:

Advantages

- Color facilitates location and processing of information.
- Color reduces confusion resulting from visual clutter and high workload.

- Consistency – minimize the difference in dialogue both within and across various human–computer interfaces.

- Memory – minimize the amount of information that the user must maintain in short–term memory.

- Structure – assist the user in developing a conceptual representation of the structure of the system so that he can navigate through the interface.

- Feedback – provide the user with feedback and error correction capabilities.

- Workload – keep the user mental workload within acceptable limits.

- Individualization – accommodate individual differences among users through automatic adaptation or user tailoring of the interface.

Green et al. (1991) describe guidelines and problems for system warnings:

Guidelines

- Warnings should be attention–getting but not startling. A problem for designers is deciding what is important enough to warrant attracting the pilot's attention.

- Warnings should inform or report. Guidance also may be given.

- Important failure annunciation should be an audio warning.

- An ideal organization for aircraft warning systems is for a single audio warning to alert the pilot to any failure and to direct his attention to a single central warning panel that announces the nature of the problem.

Problems

- Meanings of specific audio warnings have not been consistent.

- Some aircraft assign a different meaning to the same sound depending on whether the aircraft is on the ground or in flight.

- Audio warnings may be missed by the pilot if they are sounded only once.

- Repeated audio warnings can become irritating and disruptive of communications.

- Warning systems must be reliable and must not generate false alarms.

- Mode awareness must be evident to the pilot. Since automatic flight and engine management systems can be set up in many modes, it is possible for the pilot to believe that the aircraft is programmed to carry out one function when it is, in fact, performing another.

- Quick perception of auditory alerts;
- Quick response to audio warnings as compared to similar warnings presented visually;
- No need for adjustment of gaze to receive the message;
- Usefulness in situations where vision is likely to be degraded;
- Less influenced by high acceleration forces, anoxia, darkness, bright sunlight, glare, and vibration;
- May be less costly than equivalent visual displays; and
- Require little or no instrument panel display space.

Limitations of audio annunciations are that:

- Overuse can lead to auditory clutter,
- Concentration may be disrupted by intrusive or distracting annunciations,
- Presentation of comparison data may not be optimal because of the serial nature of speech, and
- Parts of long messages may be forgotten after or during transmission.

Research findings for speech versus nonspeech annunciation indicate that:

- Complex information is best presented by verbal signals (speech or visual) rather than nonverbal auditory signals (bells, tones, etc.).
- Visual displays, when combined with a speech warning, provide a shorter response time than when combined with a nonspeech warning.
- Voice warnings have the potential to be more flexible and informative than simple tone warnings.
- Use of speech is likely to be more effective during periods of high workload and stress when the meaning of coded signals may be forgotten.
- The simplicity of nonspeech annunciations make their use advantageous under certain conditions.

Nonspeech annunciations are comprised of bells, whistles, horns, buzzers, clackers, and electronic tones. They may vary in pitch, duration, and intensity. Problems associated with the use of non-speech annunciations are:

- Forgetting the meanings of nonspeech annunciations over time,
- Lack of standardization of nonspeech annunciations among aircraft,

- Speech requires time to be spoken and may be misunderstood in the presence of other competing voice messages, aural signals, or noise.

- Speech may not always provide the most rapid way of interacting with the system.

- The receiver of the speech message has difficulty processing more than one message at a time thus speech is a single-channel code.

- Messages have a transitory existence unless they are recorded for later playback.

- Speech recognition technology requires a vocabulary of acoustically distinct words.

## 5.5 Crew Workload and the Pilot-Vehicle Interface

### 5.5.1 Definition

Workload can be considered the demands of a task. It is the "work" that is "loaded" onto an operator. Imposed workload involves factors such as task objectives, environmental variables, task structure, temporal organization, system resources, operator qualifications, and incidental variables that may make task performance substantially different from one instance to the next. Perception of task demands, operator capacity, and operator behavior are operator variables that also affect workload (Hart 1984).

### 5.5.2 Workload Assessment and Measurement

#### 5.5.2.1 The Importance of Workload

The assessment and measurement of workload have played an important role in the development of automated digital cockpits. Workload can be used as a predictive tool during new system development to ensure that the pilot will have sufficient spare information processing capacity to cope with failures or unexpected events.

Workload assessment techniques can be used to identify and assess the source of temporary bottlenecks which may occur in the system when the demands of a task exceed the capabilities of the pilot and performance degrades.

Workload assessment can be used to compare operator performance on two alternative pieces of equipment or to enable comparisons to be made between individuals. In addition, workload assessment can be a factor for evaluation of new cockpits or changes to systems. Workload assessment can aid the evaluation of operating procedures and the determination of crew complement (Green et al. 1991).

Safety is another area in which workload plays an important role. Workload is an important factor in causing human error (Kantowitz and Sorkin 1983). Humans are most reliable under moderate levels of workload that do not change suddenly and unpredictably. Extremes of workload increase the likelihood of human error.

looking down at the airspeed indicator. Becoming consumed by one task can lead to error.

Stress can influence attention. The level of arousal influences the scanning pattern of an individual. For example, during the quiet period of the cruise portion of flight, when arousal is low, a pilot does not scan the instruments as frequently as he does during the approach and landing phase when arousal level is significantly higher. Under conditions of high stress and arousal the sampling rate may increase but the pattern of sampling is reduced to a narrower range of stimuli as a consequence of attention being restricted to the primary task. Important information could be missed in an emergency situation because a stressful situation caused a pilot to restrict his attention to the primary source of a problem.

Vigilance is another factor affecting attention. Over time there often is a decrease in signal detection (Edwards 1990). This can be prevented by adequate rest periods, feedback of performance on each signal, breaking the monotony with some other activity, and making the signal more noticeable (brighter, louder, etc.).

Mollard, Coblentz, and Cabon (1990) found that there is a decrease of vigilance during periods of low workload. Often the co-pilot or flight engineer experienced this low period at the same time as the pilot.

Behaviors can be changed to reduce the number of missed signals through proper training (Green et al. 1991). A pilot's skill of scanning can be altered by additional mental workload. Some findings indicate that (Edwards 1988):

- With extensive training searchers report that they do not see nontargets and that the targets stand out from the rest of the scene.
- With practice, searchers reached a very high level of stable performance.
- It is quicker to search for a target than determine its absence.
- The greater the contrast between the target and its background, the better the search.
- Color is the best cue for targets, size is next, shape is poorest.

## 5.6 Crew Resource Management, Training, and the Pilot-Vehicle Interface

### 5.6.1 Crew Interaction and the Cockpit

Straus and Cooper (1989) examined the interaction of technological and social factors and their effect on group communication. They found that willingness to exchange information in the cockpit is a precursor to effective crew performance and coordination. However, crew members often are not willing to exchange information freely. Characteristics of group structure can inhibit the kinds of communications that facilitate effective performance.

Accurate communication has been shown to be critical to group process in flight crews. Incidents from the ASRS database report cases where crew members thought they understood the intentions of other crew members but in reality did not.

### 5.6.2 Crew Resource Management and Safety

The National Plan for Aviation Human Factors (1990) indicates that there is a need for improved Cockpit Resource Management (CRM) to enhance safe operations. The importance of the cockpit social environment is stressed. The social environment is defined as flightcrew coordination, or CRM. (More recent literature favors the term Crew Resource Management in place of Cockpit Resource Management.) The Plan states that "the effect of the social environment on the crew's interaction has been demonstrated to have a significant impact on the crew's ability to perform the job safely and effectively."

Many errors in the cockpit can be traced to how the crew worked together as a team. Accident analysis reveals that often one member of the crew may have possessed crucial information but this information was not effectively communicated to the rest of the crew. CRM is based on the observation that crews who were the most effective in coping with simulated malfunctions in full mission research studies were those who communicated most (Palmer 1990).

### 5.6.3 Pilot and Crew Training

Pilot and crew training has changed since the early days of aviation where a pilot trainee for the French Foreign Legion received ground lectures then was left on his own to master basic airplane maneuvers. This trial and error method destroyed many planes, and resulted in a high trainee injury and death rate (Caro 1988).

Today, flying skills as well as crew coordination are emphasized in training. Pilot trainees may take part in full mission simulation or may learn difficult tasks through part-task training. Trainees may receive instruction in a classroom from a human teacher; independently, using a computer-based or Intelligent Tutoring System (ITS); or in an airplane simulator complete with motion, communication with ATC, and realistic flight instruments. Many operations too difficult or dangerous to perform in the air, now can be mastered on the ground through simulation. The potential for thorough and safe pilot and crew training is within grasp.

#### 5.6.3.1 Crew Resource Management Training

A typical CRM training course is multi-day and provides experiential learning in interpersonal communications, leadership, decision making, conflict resolution, stress and stress management, and related topics.

Although CRM training programs generally are quite effective, occasionally a boomerang effect or a change in the opposite direction occurs. Those who most need the training may be least influenced by the training. This problem is reflected in the National Plan for Aviation Human Factors. The section on Training and Selection describes problems associated with the role of individual differences in the effectiveness of CRM training programs. It indicates that

- Possibility of providing instruction without the presence of a teacher – although it may be beneficial to have one available for consultation, and

- Capacity to standardize instruction across a wide area.

While CBI offers a number of advantages, it does not emulate the skills of an expert instructor. CBI does not automatically adapt the simulation or level of task difficulty to match the ability of the student, identify and correct misconceptions, or choose analogies to explain complex concepts (Caro 1988). This is the realm of AI.

AI seeks to emulate the behavior of an expert. It applies rules, "learns" by experience, and solves problems not anticipated by programmers, knowledge engineers, and subject matter experts (Caro 1988).

Intelligent Tutoring Systems make use of AI. They are designed to include an instructional environment, a student model, an expert model, and an instructor model. The student's performance is logged and compared to that of the subject matter expert. The ITS provides remediation and instruction based upon comparison between student and expert actions. ITS technology can be combined with simulation to produce a high level of instruction (Johnson and Norton 1991).

Cockpit simulation can range from simple cardboard mock-ups to elaborate training devices that attempt to replicate all aspects of flight. Simulation allows dangerous maneuvers and a range of exceptional or unlikely conditions to be practiced on the ground. Instrument flight and emergency drills also can be practiced safely on the ground. Simulator training can save fuel costs and loss of revenue caused by removing a plane from service for purposes of training. For initial training, low cost simulators effectively deliver high "transfer of training" (O'Hare and Roscoe 1990).

5.6.3.5 Recurrent Training

Recurrent training allows pilots to maintain their flying skills. The Air Line Pilot's Association (ALPA) has made recommendations for improved recurrent training.

- Recurrent pilot training should be conducted on an annual basis in an approved flight simulator.

- Flight simulator training should include a review of instrument approaches and selected abnormal and emergency procedures.

- Training should be designed to represent realistic line operational concepts with emphasis on crew coordination.

- Training may include selected weather, wind shear, turbulence, and unusual approach conditions, or other circumstances that are likely to cause accidents.

## 6. CONCLUSIONS

The Pilot-Vehicle Interface should be treated as any other aircraft system. No failure mode at this interface can be taken lightly. Formal and comprehensive engineering from initial concept to completion is necessary. The design of the cockpit does not stop at the panels. Design techniques that address the complexities and variables associated with human characteristics in relation to highly integrated digital computers are required.

Improvements to flight safety can be made in several key areas. They are:

- Developing certification requirements that include human factors,
- Requiring cockpit specifications that address all aspects of the PVI,
- Integrating the human in all areas of the PVI technology,
- Generating methods of verification and validation of the PVI, and
- Using design tools that are flexible enough to include all facets of the PVI.

More economical operation of aircraft and resolutions to the problems presented by overcrowded airways can be addressed by new and innovative technology. To make flight safer, however, the solutions should not raise new problems. New and unique failure modes presented by new technology need to be considered carefully along with their solutions. Cockpit automation needs to consider a multitude of human factors.

Safety is at the heart of human factors and the PVI. The National Plan for Aviation Human Factors represents a massive effort to identify and coordinate human factors research in the field of aviation. In addition to outlining issues and problems and providing specific objectives, the plan also cites work in progress. This is a valuable document for guiding research and preventing duplication of effort. As the National Plan is implemented, it will be interesting to observe the impact of research it generates on the design and safety of cockpits.

Improved safety, economy, efficiency, and performance have been cited as reasons for automating the cockpit. Of major concern is implementing automation to a point where the pilot is "out of the loop." When automatic systems assume most control functions, the pilot's chief role becomes that of a monitor. Among the risks of this situation are loss of aviation skills, boredom and complacency, inadequate preparation to deal with system failures, and loss of job satisfaction. Key questions in this area are which functions should be automated and to what degree should the cockpit be automated. Keeping the pilot informed and

BIBLIOGRAPHY

92-0014/EG, 1992 SAE Cooperative Engineering Program, Report to Aerospace Industry, Society of Automotive Engineers, 1992.

Adam, Eugene C., John M. Reising and Bertram G. Dunlop, "Tactical Cockpits, The Coming Revolution," Tutorial of the IEEE/AESS NAECON 1992 Conference, Institute of Electrical and Electronics Engineers, Dayton Section, Dayton, Ohio, May 1992.

Advisory Circular No. 21-303.1A, Certification Procedures for Products and Parts, U.S. Department of Transportation, Federal Aviation Administration, August 10, 1972.

Advisory Circular No. 25-11, Transport Category Airplane Electronic Display Systems, U.S. Department of Transportation, Federal Aviation Administration, June 21, 1988.

Advisory Circular No. 25-1309.1A, System Design and Analysis, U.S. Department of Transportation, Federal Aviation Administration, July 16, 1987.

AGARD-CP-329, Advanced Avionics and the Military Aircraft Man/Machine Interface, April 1982.

AGARD-CP-371, Human Factors Considerations in High Performance Aircraft, May 1984.

AGARD-CP-417, The Design Development and Testing of Complex Avionic Systems, May 1987.

AGARD-CP-473, Computer Aided System Design and Simulation, August 1990.

AGARD-CP-478, Situational Awareness in Aerospace Operations, April 1990.

AGARD-CP-495, Progress in Military Airlift, December 1990.

AGARD Lecture Series Number 125, Human Factors Aspects of Aircraft Accidents, October 1982.

AGARD Lecture Series Number 126, Modern Display Technologies for Airborne Applications, April 1983.

AGARD Lecture Series Number 129, Speech Processing, May 1983.

AGARD Lecture Series Number 156, Visual Effects in the High Performance Cockpit, April 1988.

ARP4103, Flight Deck Lighting for Commercial Aircraft, Society of Automotive Engineers, February 1989.

ARP4105, Abbreviations and Acronyms for use on the Flight Deck, Society of Automotive Engineers, July 1992.

ARP4107, Aerospace Glossary for Human Factors Engineers, Society of Automotive Engineers, August 1988.

ARP4155, Human Interface Design Methodology for Integrated Display Symbology, Society of Automotive Engineers, October 1990.

AS8034, Minimum Performance Standard for Airborne Multipurpose Electronic Displays, Society of Automotive Engineers, December 1982.

Ashmore, A. S., "Certification of Digital Systems for Civil Aircraft," Certification of Avionic Systems: Proceedings of the Symposium, Royal Aeronautical Society, London, United Kingdom, April 1982.

Aviation Safety/Automation Program Conference Proceedings, NASA Conference Publication 3090, October 1989.

Aviation Week and Space Technology, "Avionics Companies Keep Pilots in Mind in Developing Computerized Systems," March 23, 1992.

Aviation Week and Space Technology, "Sextant Develops Enhanced Vision Systems, LCDs for Commercial Transport Cockpits," May 11, 1992.

Barnes, Valerie E. and William P. Monan, "Cockpit Distractions: Precursors to Emergencies," Proceedings of the Human Factors Society 34th Annual Meeting, Volume 2, The Human Factors Society, Santa Monica, CA, 1990.

Barthelemy, Kristen, K., Kim M. Mazur, and John M. Reising, "Color Coding and Size Enhancement of Switch Symbol Critical Features," Proceedings of the Human Factors Society 34th Annual Meeting, Volume 1, The Human Factors Society, Santa Monica, CA, 1990.

Beek, Bruno and Richard S. Vonusa, "General Review of Military Applications of Voice Processing," AGARD LS-129, Speech Processing, May 1983.

Beringer, Dennis B., " Target Size, Location, Sampling Point and Instructional Set: More Effects On Touch Panel Operation," Proceedings of the Human Factors Society 34th Annual Meeting, Volume 1, The Human Factors Society, Santa Monica, CA, 1990.

Billings, Charles E., "Toward a Human-Centered Aircraft Automation Philosophy," The International Journal of Aviation Psychology, Volume 1, No. 4, 1991.

Bitton, Foster and Rick Evans, Report on Head-Up Display Symbology Standardization, USAF IFC-TR-91-01, USAF Instrument Flight Center, Randolph AFB, Texas, November 1990.

Craven, D. A., "A Demonstration of an Aircraft Navigator's Moving Map Display," ARL-SYS-TM-141, Defence Science and Technology Organization, Aeronautical Research Laboratory, Melbourne, Victoria, February, 1990.

Craven, D. A., "Programmable Cockpit - Inter-Computer Communications and Data Flow," ARL-SYS-TM-139, Defence Science and Technology Organization, Aeronautical Research Laboratory, Melbourne, Victoria, March 1990.

Curran, James, "Electronic Display of Navigation Chart Information," Conference Proceedings: Cockpit Electronic Display Workshop, FAA/Volpe National Transportation Center, Cambridge, MA, April 1992.

Davis, Michael C. and Paul Pencikowski, "Reconfigurable Cockpit Development," IEEE 1988 National Aerospace and Electronics Conference (NAECON), IEEE Service Center, Piscataway, NJ, 1988.

Diehl, Alan E., "Human Performance and System Safety Considerations in Aviation Mishaps," The International Journal of Aviation Psychology, Volume 1, No. 2, 1991.

Display Devices, "VFD Advances Promote Expanding Applications as Automotive Instrument Display Panels," Serial No. 5, 1992.

Donoghue, J. A., "CRT Cockpits: Two Answers to the Same Question," Air Transport World, April 1983.

Earhart, Leroy, "MIL-STD-1553: Testing and Test Equipment," Proceedings of the 2nd AFSC Standardization Conference, Air Force Systems Command, Wright-Patterson Air Force Base, OH, November 1982.

Edwards, David C., Pilot: Mental and Physical Performance, Iowa State University Press, Ames, Iowa, 1990.

Edwards, Elwyn, "Man and Machine: Systems for Safety," Proceedings of British Airlines Pilots Association Technical Symposium, London, 1972.

Edwards, Elwyn, "Introductory Overview," in Earl L. Weiner and David C. Nagel, eds., Human Factors in Aviation, Academic Press, Inc., San Diego, 1988.

Elwell, D., et al., Avionic Data Bus Integration Technology, DOT/FAA/CT-91-19, FAA Technical Center, Atlantic City International Airport, NJ, 1991.

Engel, S. E. and R. E. Grenada, Guidelines for Man/Display Interfaces, Technical Report 00.2720, IBM, Poughkeepsie, NY, 1975.

Erickson, Jeffery B., "The Commercial Transport Cockpit," Avionics, October 1991.

Farmer, Eric, ed., Stress and Error In Aviation, Avebury Technical, Brookfield, VT, 1991.

Hecht, Herbert and Myron Hecht, Computer Resources Handbook for Flight Critical Systems, ASD-TR-85-5020, Air Force Systems Command, Wright-Patterson Air Force Base, OH, January 1985.

Helmreich, Robert L. and John A. Wilhelm, "Outcomes of Crew Resource Management Training, The International Journal of Aviation Psychology, Volume 1, No. 4, 1991.

Henderson, Breck W., "NASA Ames Pushes Automation Toward Human-Centered Design," Aviation Week and Space Technology, March 23, 1992.

Hockenbrock, Richard, "New Technology Advances for Brighter Color CRT Displays," Proceedings of the International Society for Optical Engineering, March 1989.

Hughes, A. J., "Liquid Crystal Displays," AGARD LS-126, Modern Display Technologies for Airborne Applications, Introduction and Overview, April 1983.

Hughes, David, "Pilots Support 767 Automated Cockpit, But Cite Mismatch with ATC System," Aviation Week and Space Technology, March 23, 1992.

Hughes, David, "Pilots, Research Studies Give Mixed Reviews to Glass Cockpit, "Aviation Week and Space Technology, March 23, 1992.

Human Error Avoidance Techniques: Proceedings of the Second Conference, Herndon, VA. September 18-19, 1989.

Human Factors and Risk Management in Advanced Technology: Proceedings, 40th Annual International Air Safety Seminar, Flight Safety Foundation, October 1987.

Hunt, Geoffrey H., "Introduction and Overview," Modern Display Technologies for Airborne Applications, AGARD LS-126, April 1983.

Hurst, Ronald and Leslie R. Hurst, Pilot Error: The Human Factor, 2nd ed., Jason Aronson, Inc., NY, 1982.

Improving Aircraft Safety, National Academy of Sciences, Washington, DC, 1980.

Johnson, William B. and Jeffrey E. Norton, "Using Intelligent Simulation to Enhance Human Performance in Aircraft Maintenance," Proceedings of the International Conference on Aging Aircraft and Structural Airworthiness, Federal Aviation Administration and National Aeronautics and Space Administration, Washington, DC, 1991.

Ineson, J., "Imagery for a Virtual Cockpit," Displays Technology and Applications, Volume 12, No. 3/4, July/October 1991.

Kantowitz, Barry H. and Patricia A. Casper, "Human Workload in Aviation," in Earl L. Weiner and David C. Nagel, Human Factors in Aviation, Academic Press, San Diego, 1988.

McDaniel, Joe W., "Rules for Fighter Cockpit Automation," IEEE 1988 National Aerospace and Electronics Conference (NAECON), IEEE Service Center, Piscataway, NJ, 1988.

MacPherson, Malcolm, ed., The Black Box, Quill, New York, 1984.

Mejzak, Richard S., Matthew L. Sparta, and Norman W. Warner, "Crew Systems Engineering Methodology: Process and Display Requirements," Proceedings of the IEEE/AIAA 10th Digital Avionics Systems Conference, Institute of Electrical and Electronics Engineers, New York, NY, 1991.

Metalis, S. A., "Heart Period as a Useful Index of Pilot Workload in Commercial Transport Aircraft," The International Journal of Aviation Psychology, Volume 1, 1992.

Michel, J. P., "Cathode Ray Tubes," AGARD LS-126, Modern Display Technologies for Airborne Applications, April 1983.

Michel, J. P., "Large Area Gas Discharge Displays or Plasma Displays," AGARD LS-126: Modern Display Technologies for Airborne Applications, April 1983.

Mollard, R., A. Coblentz and P. Cabon, "Vigilance in Transport Operations: Field Study In Air Transport and Railways," Proceedings of the Human Factors Society 34th Annual Meeting, Volume 2, Human Factors Society, Santa Monica, CA, 1990.

Moray, N., "Attention to Dynamic Visual Displays In Man-Machine Systems," in R. Parasuraman and R. Davies, eds., Varieties of Attention, Academic Press, New York, 1984.

Morrison, Jeffrey G., Jonathan P. Gluckman, and John E. Deaton, "Adaptive Function Allocation for Intelligent Cockpits," NADC-91028-60, Naval Air Development Center, Warminster, PA., January 1991.

Muckler, Frederick A., ed., Human Factors Review: 1984, The Human Factors Society, Inc., Santa Monica, California, 1984.

Murch, Gerald, "Ambient Illuminance Levels in the F16C Cockpit," Technical Report No. HFG-802-1, Human Factors Research, Tektronix Laboratories, August 1987.

Murch, Gerald M., Visual Demands for Avionics Displays, Tektronix Laboratories, 1985.

Mykityshyn, Mark, "Design and Evaluation of Electronic Instrument Approach Charts and Research Issues Related to Dynamic Formatting and De-cluttering," Conference Proceedings: Cockpit Electronic Display Workshop, FAA/Volpe National Transportation Center, Cambridge, MA, April 1992.

NASA-CP-10036, Flight Deck Automation: Promises and Realities, NASA Ames Research Center, Moffett Field, CA, 1989.

Poduval, Ashok, "The Over-Loaded Flight Deck Computer," 40th Annual International Air Safety Seminar Proceedings: Human Factors and Risk Management in Technology, Flight Safety Foundation, Arlington, VA, 1987.

Pomykacz, Michael A., Report of Study on Airlines' Anticipated Near Future Cockpit Control and Display Capabilities and Plans for Data Link Communication, DOT/FAA/CT-TN91/7, U.S. Department of Transportation, Federal Aviation Administration, February 1991.

Prince, D. and K. T. Burnette, "Light Emitting Diodes," AGARD LS-126, Modern Display Technologies for Airborne Applications, April 1983.

Private Pilot Manual, Jeppesen Sanderson, Inc. Englewood, CO., 1989.

Proceedings of the 1990 AIAA/FAA Joint Symposium on General Aviation Systems, DOT/FAA/CT-90/11, U.S. Department of Transportation, Federal Aviation Administration, May 1990.

Proceedings of the IEEE/AIAA 5th Digital Avionics Systems Conference, Institute of Electrical and Electronics Engineers, New York, NY, 1983.

Proceedings of the IEEE/AIAA 7th Digital Avionics Systems Conference, Institute of Electrical and Electronics Engineers, New York, NY, 1986.

Proceedings of the AIAA/IEEE 8th Digital Avionics Systems Conference, American Institute of Aeronautics and Astronautics, Washington, DC, 1988.

Proceedings of the AIAA/IEEE 9th Digital Avionics Systems Conference, American Institute of Aeronautics and Astronautics, Washington, DC, 1990.

Proceedings of the AIAA/IEEE 10th Digital Avionics Systems Conference, American Institute of Aeronautics and Astronautics, Washington, DC, 1991.

Proceedings of the Human Factors Society 33rd Annual Meeting, Volume 1,2, Human Factors Society, Santa Monica, CA, October 1989.

Proceedings of the Human Factors Society 34th Annual Meeting, Volume 1,2, Human Factors Society, Santa Monica, CA, October 1991.

Proceedings of the Seventh Aerospace Behavioral Technology Conference, April 1989.

Proceedings of IEEE 1985 National Aerospace and Electronics Conference (NAECON), Institute of Electrical and Electronics Engineers, New York, NY, 1985.

Proceedings of IEEE 1986 National Aerospace and Electronics Conference (NAECON), Institute of Electrical and Electronics Engineers, New York, NY, 1986.

Proceedings of IEEE 1987 National Aerospace and Electronics Conference (NAECON), Institute of Electrical and Electronics Engineers, New York, NY, 1987.

Scott, William B., "Douglas Uses MD-11 Cockpit Success As Springboard for Designing MD-12," Aviation Week and Space Technology, March 23, 1992.

Selcon, Stephen J., "Decision Support in the Cockpit: Probably a Good Thing," Proceedings of the Human Factors Society 34th Annual Meeting, Volume 1, Human Factors Society, Santa Monica, CA, 1990.

Sexton, G. A., "Crew Systems and Flight Station Concepts for a 1985 Transport Aircraft," NASA Contractor Report 166068, NASA-Langley Research Center, VA., 1983.

Sexton, G. A., "Cockpit-Crew Systems Design and Integration," in Earl L. Weiner and David C. Nagel, Human Factors In Aviation, Academic Press, San Diego, 1988.

Shimmin, J. A., "The Certification of the Avionic Systems on the ATP to JAR 25," European Forum: The Evolution of Regional Aircraft Technologies and Certification, DGLR-Bericht-89-02, German Society for Aeronautics and Astronautics, Bonn, Federal Republic of Germany, 1989.

Silverstein, Louis D., and Robin M. Merrifield, The Development and Evaluation of Color Systems for Airborne Applications, Phase I: Fundamental Visual, Perceptual, and Display System Considerations, DOT/FAA/PM-85/19, U.S. Department of Transportation, Federal Aviation Administration, July 1985.

Simpson, Carol A., et al., "System Design for Speech Recognition and Generation," Human Factors, April 1985.

Smith, S. L., J. N. Mosier, Guidelines for User-System Interface Software, Technical Report ESD-TR-84-190, The Mitre Corporation, Bedford, MA, 1984.

Spitzer, Cary R., "All-Digital Jets Are Taking Off," IEEE Spectrum, Institute of Electrical and Electronics Engineers, New York, NY, September 1986.

Spitzer, Cary R., "Digital Avionics Architectures - Design and Assessment," Tutorial of the IEEE/AIAA 7th Digital Avionics Systems Conference, Institute of Electrical and Electronics Engineers, New York, NY, 1986.

Spitzer, Cary R., "Digital Avionics," Tutorial of the IEEE/AIAA 8th Digital Avionics Systems Conference, Institute of Electrical and Electronics Engineers, New York, NY, 1986.

Spitzer, Cary R., Digital Avionics Systems, Prentice Hall, Englewood Cliffs, NJ, 1987.

Steiner, Bruce A. and Monica A. Camacho, "Situation Awareness: Icons vs. Alphanumerics," Proceedings of the Human Factors Society 33rd Annual Meeting, Volume 1, The Human Factors Society, Santa Monica, CA, 1989.

Stokes, Alan F. and Christopher D. Wickens, "Aviation Displays," in Earl L. Weiner and David C. Nagel, eds., Human Factors in Aviation, Academic Press, Inc., San Diego, 1988.

Wiener, Earl L., "Reflections on Human Error: Matters of Life and Death, "Proceedings of the Human Factors Society 33rd Annual Meeting, Volume 1, The Human Factors Society, Santa Monica, CA, 1989.

Wiener, Earl L., "Barriers to Situational Awareness In the Cockpit and Possible Intervention Strategies," Proceedings of the Human Factors Society 34th Annual Meeting, Volume 2, The Human Factors Society, Santa Monica, CA, 1990.

Wiener, Earl L. and R.E. Curry. "Flight-deck Automation: Promises and Problems," Ergonomics, 23, 1980.

Wightman, Dennis C. and Gavan Lintern, "Part-Task Training for Tracking and Manual Control," Human Factors, June 1985.

Wightman, Dennis C. and Frank Sistrunk, "Part-Task Training Strategies in Simulated Carrier Landing Final-Approach Training," Human Factors, June 1987.

Williams, James H., "Issues Concerning Certification of Integrated Cockpit Avionics," SAE Conference on General Aviation, Wichita, KS, April 12, 1989.

Williges, Beverly H. and Robert C. Williges, "Dialogue Design Considerations for Interactive Computer Systems," Human Factors Review, Human Factors Society, Inc., Santa Monica, California, 1984.

Williges, Robert C., Beverly H. Williges, and J. Elkerton, "Software Interface Design," in G. Salvendy, ed., Handbook of Human Factors, Wiley, NY, 1987.

Williges, Robert C., Beverly H. Williges, and Robert G. Fainter, "Software Interfaces for Aviation Systems," in Earl L. Weiner and David C. Nagel, eds., Human Factors in Aviation, Academic Press, Inc., San Diego, 1988.

Wilson, Glenn F., "Classification of Flight Segment Using Pilot and WSO Physiological Data," Proceedings of the Human Factors Society 34th Annual Meeting, Volume 1, Human Factors Society, Santa Monica, CA, 1990.

## GLOSSARY

ANISOTROPIC. Having different, direction dependent, physical characteristics.

ATTENSON. A unique sound used on the flight deck for alerting the flightcrew of warning, caution, and advisory conditions.

AVIONIC. Electronic equipment used in aircraft.

BIREFRINGENCE. The splitting of light into two components, where each component travels at a different velocity.

CHOLESTERIC. A liquid crystal phase with molecules parallel to each other but twisting slightly from layer to layer.

COMBINER. A screen designed to reflect selected wavelengths of light while remaining transmissive for others.

DATA BUS. One or more wires or optical fibers that carry signals that represent information.

DICHROIC. Having different light absorption characteristics based on incident polarization direction.

DIELECTRIC. Having an insulating property with respect to an electrical field.

FEDERAL AVIATION REGULATIONS. Subchapter C of the Code of Federal Regulations (CFR), Title 14, Chapter 1.

GETTER. A metal alloy used in a vacuum tube to absorb residual gasses.

GRAY SCALE. A series of tones, varying from black to white.

HERMETIC. Having an airtight seal.

ISOTROPIC. Having the same physical properties in all directions.

LINE REPLACEABLE UNIT. An electronics unit that is made to be replaced on the flight line, as opposed to one that requires the aircraft be taken to the shop for repair.

MONOLITHIC. A single substrate used for an integrated circuit.

NEMATIC. A liquid crystal phase with molecules having a single optical axis in line with an applied magnetic field.

## ACRONYMS AND ABBREVIATIONS

| | |
|---|---|
| $A^3I$ | Army–NASA Aircrew/Aircraft Integration |
| ABET | Aerospace Behavioral Engineering Technology |
| ac | Alternating Current |
| AC | Advisory Circular |
| AD | Alerting Display |
| ADF | Automatic Direction Finder |
| AI | Artificial Intelligence |
| AIR | Aerospace Information Report |
| ALPA | Air Line Pilot's Association |
| AM | Active–Matrix |
| AMS | Aerospace Material Specification |
| ARP | Aerospace Recommended Practice |
| AS | Aerospace Standard |
| ASRS | Aviation Safety Reporting System |
| ATC | Air Traffic Control |
| BITE | Built–in Test Equipment |
| CAA | Civil Aviation Authority |
| CAD | Computer Aided Design |
| CALSEL | Call Select |
| CBI | Computer Based Instruction |
| CDU | Control Display Unit |
| CE | Certification Engineer |
| CFR | Code of Federal Regulations |
| cm | Centimeter |
| CMU | Computer Mock–Up |
| COMBIMAN | Computerized Biomechanical Man–model |
| CPU | Central Processing Unit |
| CRM | Cockpit Resource Management (also Crew Resource Management) |
| CRT | Cathode Ray Tube |
| CSERIAC | Crew System Ergonomics Information Analysis Center |
| dc | Direct current |
| DEP | Design Eye Position |
| DME | Distance Measuring Equipment |
| DoD | Department of Defense |
| EADI | Electronic Attitude Director Indicator |
| EFIS | Electronic Flight Instrument System |
| $E_g$ | Energy gap |
| EL | Electroluminescent |
| ELS | Electronic Library System |
| EVS | Enhanced Vision System |
| FAA | Federal Aviation Administration |
| FAR | Federal Aviation Regulation |
| FAS | Flight Deck Alerting System |
| FCC | Flight Control Computer |
| FDI | Flight Director Indicator |

| | |
|---|---|
| PNI | Primary Navigation Instrument |
| PVI | Pilot–Vehicle Interface |
| RCP | Reconfigurable Cockpit |
| $R_L$ | Load Resistor |
| ROM | Read–Only–Memory |
| SAE | Society of Automotive Engineers (now known as Engineering Society for Advancing Mobility Land Sea Air and Space) |
| SATCOM | Satellite Communications |
| SATNAV | Satellite Navigation |
| SD | System Display |
| SELCAL | Select Call |
| SOP | Standard Operating Procedure |
| STN | Supertwisted Nematic |
| TACAN | Tactical Air Navigation |
| TFEL | Thin–Film Electroluminescence |
| TFT | Thin–Film Transistor |
| TN | Twisted Nematic |
| TSO | Technical Standard Orders |
| TTL | Transistor–Transistor Logic |
| UHF | Ultra High Frequency |
| UV | Ultra–Violet |
| V | Volt |
| $V_B$ | Breakdown voltage |
| $V_{CC}$ | Collector Circuit Voltage |
| $V_F$ | Forward Voltage |
| VFD | Vacuum Florescent Display |
| VHF | Very High Frequency |
| VLF | Very Low Frequency |
| VOR | Very High Frequency Omnidirectional Range |
| $V_S$ | Sustain–Voltage |
| ZnS | Zinc Sulfide |
| ZnSe | Zinc Selenide |



☆U.S. GOVERNMENT PRINTING OFFICE: 1993 - 504-078/80201

U.S. Department of Transportation
Federal Aviation Administration

COPY — 1

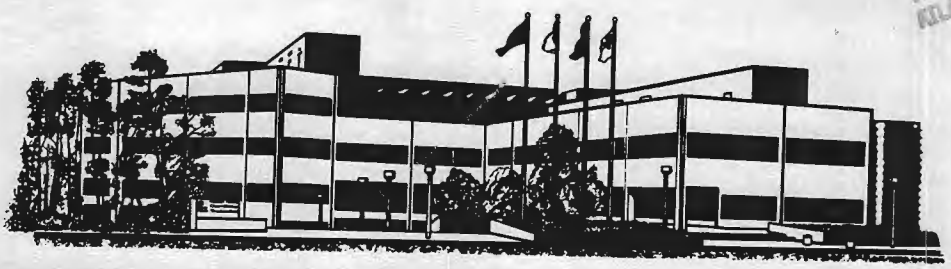
DOT/FAA/CT-88/10

# HANDBOOK - VOLUME II
# DIGITAL SYSTEMS VALIDATION

## CHAPTER 20
## ARTIFICIAL INTELLIGENCE WITH APPLICATIONS FOR AIRCRAFT

FAA WJH Technical Center
*00026268*

FEDERAL AVIATION ADMINISTRATION
NOV 9 1994
TECHNICAL CENTER LIBRARY
ATLANTIC CITY INT'L AIRPT, NJ 08405



FAA TECHNICAL CENTER
Atlantic City, N.J.

FEDERAL AVIATION ADMINISTRATION
TECHNICAL CENTER
ATLANTIC CITY INTERNATIONAL AIRPORT, NEW JERSEY 08405

## NOTICE

This document is disseminated under the sponsorship of the U.S. Department of Transportation in the interest of information exchange. The United States Government assumes no liability for the contents or use thereof. The United States Government does not endorse products or manufacturers. Trade or manufacturer's names appear herein solely because they are considered essential to the object of this report.

Technical Report Documentation Page

| 1. Report No. | 2. Government Accession No. | 3. Recipient's Catalog No. |
|---|---|---|
| DOT/FAA/CT-88/10 | | |

| 4. Title and Subtitle | 5. Report Date |
|---|---|
| HANDBOOK-VOLUME II<br>DIGITAL SYSTEMS VALIDATION CHAPTER 20<br>ARTIFICIAL INTELLIGENCE WITH APPLICATIONS FOR AIRCRAFT | July 1994 |
| | 6. Performing Organization Code |

| 7. Author(s) | 8. Performing Organization Report No. |
|---|---|
| L. Harrison, P. Saunders, J. Janowitz | |

| 9. Performing Organization Name and Address | 10. Work Unit No. (TRAIS) |
|---|---|
| Galaxy Scientific Corporation<br>2500 English Creek Avenue<br>Building 11<br>Pleasantville, NJ 08232 | 11. Contract or Grant No.<br>DTFAO3-89-C-00043 |

| 12. Sponsoring Agency Name and Address | 13. Type of Report and Period Covered |
|---|---|
| U.S. Department of Transportation<br>Federal Aviation Administration<br>Technical Center<br>Atlantic City International Airport, NJ 08405 | Tutorial<br>Handbook Chapter 20 |
| | 14. Sponsoring Agency Code<br>ACD-230 |

15. Supplementary Notes

Pete Saraceni, FAA Technical Center, Program Manager, (609) 485-5577
(Note: This tutorial is a condensed version of FAA Technical Center final report DOT/FAA/CT-94/41.)

16. Abstract

This chapter provides an overview of Artificial Intelligence (AI) technology, one of the more complex applications of digital systems. This chapter examines AI-based technology, focusing on three fields: neural networks, fuzzy logic, and Expert Systems. This chapter provides the reader with the background and a basic understanding of the fundamentals of these fields. Another section examines aspects of the AI development environment, including languages, tools, and AI-based hardware components.

Some of the proposed aviation-related applications for both civil and military aircraft, including pilot assistants and diagnostic aids, are surveyed. Additionally, certification issues, including regulations, guidelines, and verification and validation techniques are examined. Human factors issues relating to the use of this technology are identified and reviewed. In addition, this chapter identifies safety issues and concerns over the use of this technology in airborne systems.

| 17. Key Words | 18. Distribution Statement |
|---|---|
| Artificial Intelligence, avionics, blackboard system, certification, chaining, database, development tool, expert system, fuzzy logic, guidelines, human factors, inference engine, knowledge base, learning, neural network, rule base, search technique, shell, validation, verification. | Document is available to the public through the National Technical Information Service (NTIS), Springfield, Virginia 22161 |

| 19. Security Classif. (of this report) | 20. Security Classif. (of this page) | 21. No. of Pages | 22. Price |
|---|---|---|---|
| Unclassified | Unclassified | 135 | |

Form DOT F1700.7 (8-72) Reproduction of completed page authorized

TABLE OF CONTENTS


| Chapter | | Page |
|---|---|---|
| EXECUTIVE SUMMARY | | 20-xi |
| 1 | INTRODUCTION | 20-1 |
| 1.1 | Defining Artificial Intelligence | 20-1 |
| 1.2 | Artificial Intelligence and Aviation Applications | 20-1 |
| 2 | ARTIFICIAL INTELLIGENCE OVERVIEW | 20-3 |
| 2.1 | History | 20-5 |
| 2.2 | Expert Systems | 20-6 |
| 2.2.1 | Expert System Architecture | 20-7 |
| 2.2.2 | Expert System Implementation | 20-8 |
| 2.2.3 | Explanation of Reasoning | 20-9 |
| 2.2.4 | Knowledge Representation | 20-9 |
| 2.2.5 | Updating the Knowledge Base | 20-17 |
| 2.2.6 | Chaining | 20-17 |
| 2.2.7 | Knowledge Acquisition | 20-18 |
| 2.2.8 | Applications | 20-19 |
| 2.2.9 | Heuristic Reasoning | 20-20 |
| 2.2.10 | Searching | 20-20 |
| 2.2.11 | Expert System Drawbacks | 20-21 |
| 2.2.12 | Future of Expert Systems | 20-23 |
| 2.2.13 | Blackboard Systems | 20-23 |
| 2.3 | Fuzzy Logic | 20-26 |
| 2.3.1 | The Basic Theory | 20-26 |
| 2.3.2 | Crisp Logic | 20-28 |
| 2.3.3 | Fuzzy Control | 20-29 |
| 2.3.4 | Control Systems | 20-31 |
| 2.3.5 | Probability | 20-33 |
| 2.3.6 | Parallel Processing | 20-33 |
| 2.3.7 | Hardware | 20-33 |
| 2.3.8 | Designing a Fuzzy System | 20-33 |
| 2.3.9 | Fuzzy Logic Advantages | 20-34 |
| 2.3.10 | Fuzzy Logic Disadvantages | 20-35 |
| 2.3.11 | Applications | 20-35 |
| 2.3.12 | Future of Fuzzy Logic | 20-36 |
| 2.4 | Neural Networks | 20-36 |
| 2.4.1 | Learning | 20-37 |
| 2.4.2 | Transfer Functions | 20-38 |
| 2.4.3 | Construction | 20-38 |

TABLE OF CONTENTS
(Continued)


| Chapter | | | Page |
|---|---|---|---|
| | 2.4.4 | Designing a System | 20-40 |
| | 2.4.5 | Applications | 20-41 |
| | 2.4.6 | Future of Neural Networks | 20-41 |
| 2.5 | Integrated AI Systems | | 20-41 |
| | 2.5.1 | Fuzzy Logic Integrated with Neural Networks | 20-41 |
| | 2.5.2 | Expert Systems Integrated with Neural Networks | 20-42 |
| | 2.5.3 | Expert Systems Integrated with Fuzzy Logic | 20-43 |
| 3 | ARTIFICIAL INTELLIGENCE AND EXPERT SYSTEM DEVELOPMENT | | 20-45 |
| 3.1 | Software for Artificial Intelligence Applications | | 20-45 |
| | 3.1.1 | List Processor | 20-46 |
| | 3.1.2 | Programming in Logic | 20-46 |
| | 3.1.3 | Tools for the Expert System | 20-46 |
| | 3.1.4 | Fuzzy Languages for Development | 20-46 |
| | 3.1.5 | Other Languages and Tools | 20-47 |
| 3.2 | Artificial Intelligence and Expert Systems Hardware | | 20-48 |
| | 3.2.1 | Neural Processors | 20-49 |
| | 3.2.2 | Design Considerations for Fuzzy Logic | 20-51 |
| 3.3 | Artificial Intelligence Development Systems and Tools | | 20-52 |
| | 3.3.1 | Fuzzy Development Systems | 20-52 |
| | 3.3.2 | Fuzzy Logic for Digital Signal Processors | 20-54 |
| | 3.3.3 | Neural Network Development Tools | 20-54 |
| | 3.3.4 | Neural Network and Fuzzy Logic Combined System | 20-55 |
| 4 | ARTIFICIAL INTELLIGENCE AND EXPERT SYSTEM APPLICATIONS | | 20-57 |
| 4.1 | Design Considerations for Avionic Applications | | 20-57 |
| | 4.1.1 | Architecture Considerations | 20-57 |
| | 4.1.2 | Timing Considerations | 20-57 |
| 4.2 | Flight Management Expert System | | 20-58 |
| 4.3 | Navigation Systems | | 20-59 |
| 4.4 | Decision Support Systems | | 20-59 |
| | 4.4.1 | Cockpit Assistant System | 20-59 |
| | 4.4.2 | Diverter System | 20-60 |

TABLE OF CONTENTS
(Continued)


| Chapter | | | Page |
|---|---|---|---|
| | 4.4.3 | Flight-Plan Interactive Negotiation and Decision-Aiding System | 20-61 |
| | 4.4.4 | Pilot's Associate Program | 20-63 |
| | 4.4.5 | Rotorcraft Pilot's Associate Program | 20-63 |
| | 4.5 | Intelligent Monitoring and Diagnostic Systems | 20-65 |
| | 4.5.1 | Rotorcraft Transmission Health Monitoring System | 20-65 |
| | 4.5.2 | Technical Expert Aircraft Maintenance System | 20-66 |
| 5 | CERTIFICATION CONSIDERATIONS FOR ARTIFICIAL INTELLIGENCE | | 20-69 |
| | 5.1 | Certification and Federal Aviation Regulations | 20-69 |
| | 5.2 | Certification Guidelines | 20-70 |
| | 5.2.1 | Hardware Guidelines | 20-70 |
| | 5.2.2 | Software Guidelines | 20-71 |
| | 5.2.3 | Software Guidelines and Artificial Intelligence | 20-72 |
| | 5.3 | Artificial Intelligence Standards | 20-73 |
| | 5.4 | Conventional Software Verification and Validation | 20-73 |
| | 5.4.1 | Verification and Validation | 20-73 |
| | 5.4.2 | Verification and Validation Testing Phases | 20-77 |
| | 5.5 | Expert System Software Verification and Validation | 20-78 |
| | 5.5.1 | Contrasting Expert Systems and Conventional Software | 20-79 |
| | 5.5.2 | Specification and Development Differences | 20-83 |
| | 5.5.3 | Implementation Differences | 20-84 |
| | 5.5.4 | Verification and Validation Techniques for Expert Systems | 20-85 |
| | 5.5.5 | Supportability Issues for Expert Systems | 20-90 |
| | 5.5.6 | Configuration Management Issues for Expert Systems | 20-90 |
| | 5.5.7 | Reliability Requirements | 20-91 |
| | 5.5.8 | Testability Issues for Expert Systems | 20-92 |
| 6 | HUMAN FACTORS AND ARTIFICIAL INTELLIGENCE | | 20-93 |
| | 6.1 | Expert System Development Stages | 20-93 |
| | 6.2 | National Plan and Artificial Intelligence | 20-94 |
| | 6.3 | Keeping the Human as a Factor | 20-94 |
| | 6.4 | Automation and Other Human Factors Concerns | 20-96 |

TABLE OF CONTENTS
(Continued)


| Chapter | | Page |
|---|---|---|
| 7 | CONCLUSION | 20-97 |
| | 7.1 Learning Systems | 20-97 |
| | 7.2 Software Level Determination | 20-97 |
| | 7.3 Enhancing Safety | 20-98 |
| | 7.4 Hardware and Software Failures | 20-98 |
| | 7.5 Ground-Based Avionics? | 20-99 |
| | 7.6 Verification and Validation Issues | 20-99 |
| | 7.7 Tool Qualification | 20-100 |
| | 7.8 Design Guidance | 20-100 |
| | 7.9 Human Factors Issues for Expert Systems | 20-100 |
| BIBLIOGRAPHY | | 20-101 |
| GLOSSARY | | 20-123 |
| ACRONYMS AND ABBREVIATIONS | | 20-131 |
| INDEX | | 20-133 |

LIST OF ILLUSTRATIONS


| Figure | Page |
|---|---|
| 2.2-1 EXPERT SYSTEM ARCHITECTURE | 20-7 |
| 2.2-2 SEMANTIC NETWORK STRUCTURE | 20-16 |
| 2.2-3 BREADTH FIRST SEARCH TREE | 20-21 |
| 2.2-4 DEPTH FIRST SEARCH TREE | 20-22 |
| 2.2-5 BLACKBOARD SYSTEM ARCHITECTURE | 20-24 |
| 2.3-1 FUZZY MEMBERSHIP SETS | 20-27 |
| 2.3-2 CRISP VERSUS FUZZY LOGIC | 20-28 |
| 2.3-3 OBTAINING THE LOGICAL PRODUCT | 20-30 |
| 2.3-4 THE LOGICAL SUM | 20-32 |
| 2.4-1 NEURAL NETWORK | 20-39 |
| 2.4-2 BIOLOGICAL AND ARTIFICIAL NEURONS | 20-40 |
| 3.2-1 OPTICAL NEURAL COMPUTING INTEGRATED CIRCUIT | 20-50 |
| 4.4-1 DIVERSION PROPOSAL GENERATION | 20-62 |
| 4.4-2 ROTORCRAFT PILOT'S ASSOCIATE DEMONSTRATOR CONCEPT | 20-64 |
| 6.1-1 IMPROVED HUMAN PERFORMANCE GENERATOR | 20-95 |

LIST OF TABLES


| Table | | Page |
|---|---|---|
| 2.2-1 | FRAME STRUCTURE | 20-16 |
| 2.4-1 | FUTURE EXPECTATIONS FOR NEURAL NETWORKS | 20-42 |
| 3.1-1 | COMPARISON OF EXPERT SYSTEM AND PROGRAMMING LANGUAGE DEVELOPMENT METHODS | 20-47 |
| 5.5-1 | EXPERT SYSTEM AND CONVENTIONAL SOFTWARE DIFFERENCE SUMMARY | 20-81 |
| 5.5-3 | EXPERT SYSTEM VERIFICATION AND VALIDATION TECHNIQUES | 20-86 |

# EXECUTIVE SUMMARY

This tutorial was initiated by the Federal Aviation Administration Technical Center's Directorate for Aircraft Safety, Flight Safety Research Branch. The purpose of the tutorial is to provide an overview of Artificial Intelligence (AI) technology, highlighting selected fields. This tutorial is based on the technical report DOT/FAA/CT-94/41 with the same title.

One of the more complex applications of digital systems is that of AI. This report examines AI-based technology, the development environment, and proposed aviation-related applications of this technology. In addition, the report identifies safety issues and concerns over the use of this technology in airborne systems. There are seven sections of the report:

- Section 1 - Introduction
- Section 2 - Artificial Intelligence Overview
- Section 3 - Artificial Intelligence and Expert System Development
- Section 4 - Artificial Intelligence and Expert System Applications
- Section 5 - Certification Considerations for Artificial Intelligence
- Section 6 - Human Factors and Artificial Intelligence
- Section 7 - Conclusion

While AI-based technology may never be able to replace genuine intelligence, it can provide users with many benefits. Developing human-centered automation and designing advanced technology that will capitalize on the relative strengths of humans and machines are key to the success and usefulness of AI. For example, Expert Systems (ESs) can consistently provide expert advice in a timely manner. ESs are not influenced by factors which impair human decision making, such as stress or emotions.

Neural Network systems are helpful for applications such as pattern recognition. Recognizing pre-failure signatures in airborne machinery is an excellent application for this particular field of AI. Fuzzy logic systems are useful for applications that normally require human intuition, are difficult to control with conventional techniques, or are difficult to model.

One of the primary activities that requires addressing by developers and certification specialists concerns Verification and Validation (V&V) methods for AI-based systems. Numerous V&V techniques that have been successfully applied to ESs are identified in this tutorial. Identifying a set of tests that can be demonstrated to satisfy safety requirements will be a challenge.

AI technology is not efficient at solving all types of problems. For a number of applications, it can, however, assist in managing problem complexity. AI-based systems for commercial aircraft are being researched and tested. Ultimately, such systems can offer economic advantages and contribute to flight safety.

# 1. INTRODUCTION

## 1.1 Defining Artificial Intelligence

Because of ongoing debate over what comprises Artificial Intelligence (AI) technology and what the research goals are, it is difficult to provide an unequivocal definition of AI. Should some fields of AI technology, such as Expert Systems (ESs) or fuzzy logic, be considered simply another software application? Should a system without learning capability be considered an AI-based system, as some believe? Others contend that due to fundamental differences between computers and people, the ultimate goal of the autonomous "thinking machine" will never be realized. Human qualities, such as intuition, can not be reproduced in the confines of silicon and software (Dreyfus and Dreyfus 1986).

Debate over these issues will continue. Numerous definitions of what comprises AI technology exist. However, for the purpose of this report, AI is defined as the subfield of computer science that attempts to use computers to emulate the way humans think and reason when solving problems. This agrees with Wickens (1992):

> The study of AI is devoted to developing computer programs that will mimic the product of intelligent human problem solving, perception, and thought.

Defined in this manner, AI includes systems that are capable of "learning," as well as those that have static rules and databases. The part of the definition that is stressed, and that most agree with, is the mimicking, or emulating, of human techniques.

In this report, particular emphasis is placed on three fields of AI technology: ESs, fuzzy logic, and Neural Networks (NNs). Aviation- and avionics-related applications, or proposed applications, have been numerous for these disciplines, but especially for the ES. Therefore, sections 4, on AI applications, and 5, on certification issues, focus on these fields of AI technology.

## 1.2 Artificial Intelligence and Aviation Applications

AI technology has existed for a number of years. During the early years, the promises made concerning this technology did not match the products delivered. With this in mind, Bouchard (1991) said:

> The user community has been stung with high development costs and false expectations, although that is changing with respect to expert systems.

Some of the early applications of AI-based technology have included image and speech recognition systems, natural language systems, and handwriting analysis. While research in these fields continues, AI-based technology is being used in a number of other applications. AI-based systems are now being used for control and monitoring systems, financial analysis, medical prognosis, manufacturing, training, sorting through large amounts of data in databases, and scheduling. A great amount of research is being performed and sponsored by the National Aeronautics and Space Administration (NASA) for space applications and for potential flightdeck applications for both the commercial and military communities. Among the Department of Defense (DOD) community, this technology is referred to as "Machine Intelligence." AI-based systems are also being used by the Nuclear Power Industry (Naser 1991). Some of the benefits of such systems include consistency of reasoning in stressful situations, reduction in time to perform certain tasks, and prevention of equipment failures using predictive diagnostics.

AI-based products that save the user time and money and outperform conventional software-based systems are currently being manufactured. According to Curran (1992), it is likely that AI and ES technology will be used in the near future in at least two avionic areas. For the aviation community, the possible benefits of using AI-based systems would include the following:

- Optimizing the use of airspace
- Reducing the cost of flying
- Meeting Air Traffic Control (ATC) requirements
- Aiding the decision making process of the flight crew
- Aiding maintenance activity
- Assisting data management

AI-based pilot aid systems are being designed by both military and commercial manufacturers. In Germany, the Cockpit Assistant System (CASSY), an AI-based pilot aid system, is undergoing flight testing, with the expectation that it will be operational as early as 1997 (Nordwall 1992). This system may use AI-based technologies, such as NNs, ESs, and fuzzy logic (Prevot, Onken, and Dudek 1991). Certification authorities in Europe and the United States should be cognizant and concerned over such activity. Above all, they should be prepared for the presentation of such an AI-based system for certification by an avionics or airframe manufacturer.

AI is a broad discipline and this report is limited in scope and depth of coverage. For further information, the reader is urged to make use of the bibliography located after section 7.

## 2. ARTIFICIAL INTELLIGENCE OVERVIEW

As stated in the introduction, AI refers to the subfield of computer science that studies the process of programming machines to perform in a manner resembling human behavior. It is the computer-based solution to complex problems using the application of processes that are analogous to the human reasoning process (Rolston 1988).

Many disciplines contribute ideas to the field of AI. Basic ideas in fields such as mathematics, psychology, linguistics, philosophy, computer science, and engineering find frequent application in AI. Intelligence requires many strengths and many of the problems faced in established disciplines intersect the natural concerns of AI-related problems.

Two of the objectives of AI are to make computers more useful and to understand the principles that make intelligence possible in living beings. As the world grows more complex, AI-based computers can assist with many tasks. Resources such as energy, food, and brain power must be used as wisely as possible. High quality help from computers in ordinary computing, as well as computing that exhibits intelligence, can assist in this area.

The ultimate goal of AI research is to build a "person." This person will have various modules that will be integrated to produce the final system. These modules involve a number of different applications relevant to AI research. The input to the system includes vision and language interfaces with the external domain. The output of the system entails robotics, speech, and screen interaction with the external domain. The internal systems include deduction, search, planning, explanation, and learning (Charniak and McDermott 1985).

Intelligence requires more than the ability to reason; it also requires a great deal of knowledge about the world. The knowledge about the world needs to be encoded in a way that the computer system will be able to interpret. There are some properties inherent in knowledge that cause difficulty for AI-related purposes. Knowledge is difficult to characterize accurately; it is constantly changing; it is virtually unbounded; and it generally needs to be organized according to the way it will be used (Rich and Knight 1991).

Robotics is a significant subarea of AI. There are many possible uses for robots. Tasks such as welding, general house cleaning, construction, and making deliveries would be appropriate for a robot. Robotics is a complex area involving aspects other than AI, such as mechanical and electrical design. Successful implementation of a fully operational, independent robot has not been achieved. Tasks that are performed almost unconsciously by humans require many of the same abilities used in solving more intellectually demanding problems. Complex robot control problems require planning at high levels of abstraction, ignoring details, and then planning at lower and lower levels where details

become important. Research into the area of robotics has helped to develop many other AI areas.

Interfacing with the user requires capabilities such as voice and visual interface. Further investigation into these areas is required for them to be useful to an AI system. An area of AI that requires more research and development is natural language comprehension. This is the ability of an AI system to interpret written or spoken words. Human communication employs complex and little understood processes almost effortlessly. The primary use of language is to transmit a piece of mental structure from one brain to another under circumstances in which each brain possesses large, highly similar surrounding mental structures that serve as a common context. A computer system capable of understanding a message in natural language would require both the contextual knowledge and the processes for making the inferences assumed by the message generator (Nilsson 1980).

Common sense reasoning involves reasoning about physical objects and their relationships to each other, as well as reasoning about actions and their consequences. For example, an object can only be in one place at a time, and if an object is dropped it will fall to the floor and may break. This reasoning is difficult to capture because it is taken for granted. People do not need to think about actions related to common sense reasoning; they are virtually instinctive. At this point in AI development, the domains that require only specialized expertise without the assistance of common sense knowledge are the AI systems that are flourishing as useful systems.

Another area of AI research is intelligent retrieval from databases. Database systems are computer systems that store a large body of facts about some subject in such a way that the data can be used to answer questions about that subject. Many techniques have been developed to enable the efficient representation, storage, and retrieval of large numbers of facts. These techniques become related to AI when the task requires retrieving answers that entail deductive reasoning from the facts in the database. The designer of an intelligent database retrieval system is faced with several problems, including understanding queries that are stated in natural language, deducing answers from stored facts, and acquiring and utilizing common knowledge.

Expert consulting systems are systems that provide users with expert conclusions about specialized subject areas. These systems include medical diagnostic systems, which, given the symptoms of the patient, may suggest tests to perform and, possibly, speculate what the disease may be (Charniak and McDermott 1985). The key problem in developing expert consulting systems is how to represent and use the knowledge that experts possess and use. This knowledge is often imprecise or uncertain.

The task of a computer generating a computer program is related to both theorem proving and robotics. The system takes in a description of what the program is supposed to do and then writes the code. Program verification and debugging are a few of the capabilities that such a system must achieve.

Specifying optimal schedules and combinations is another class of problems that can be solved using AI techniques. The effort is in making the time-versus-

problem-size curve grow as slowly as possible. Several methods have been developed for delaying and moderating the inevitable combinatorial explosion. Knowledge about the problem domain is the key to more efficient solution methods.

Research is being done in the area of a computer's ability to perceive surroundings. It would be useful for computers to be capable of understanding their surroundings by hearing and seeing them. Understanding surroundings requires a large database of knowledge about the things being perceived. The ultimate goal is to represent the scene by a natural language description. The process of hypothesis formation requires a large amount of knowledge about the expected scenes. For example, when a robot enters a room, there will be some information in the AI system about what normally can be expected to be found in a room, such as windows, doors, and furniture.

A task that may be appropriate for an AI system is proving or disproving a theorem. This requires intuitive skills, such as guessing which theorems should be proved first to prove the main theorem and the ability to make deductions from hypotheses. There are many informal tasks that can be formalized as theorem proving problems. Therefore, this is an important area in the study of AI methods (Nilsson 1980).

There are AI programs that can perform a variety of tasks, such as solving mathematical problems, predicting the presence of mineral deposits, verifying the designs of electronic components, and using television cameras to see the world and identify what is there. AI programs are also capable of playing backgammon, controlling manufacturing processes, diagnosing computer faults, designing computers, underwriting insurance, playing chess, and much more (Charniak and McDermott 1985).

Three subfields of AI are examined in detail in the following sections. They are ESs, fuzzy logic, and NNs. ESs are programmed to use the knowledge of experts applied to specific task areas. ESs can achieve high levels of system performance in limited domains. Fuzzy logic is used to model systems where information may be imprecise or incomplete. Fuzzy logic is useful for systems that normally require human intuition, are difficult to control with conventional techniques, or are difficult to model. NNs are a simplified model of the brain's structure. They are used to solve problems that require input to output data mapping.

## 2.1 History

Beginning in the 1960s, there was much interest in developing intelligent systems. Many promises were made regarding the future capabilities of computers. However, by the early 1970s, most of the promises that were made had not been met. AI research lost most of its funding and was regarded as an eccentric branch of computer science. People became skeptical of AI because of unrealistic claims and expectations. Interest and activity in the area of AI research grew again, beginning in the late 1970s and peaking in the mid-1980s. Fuzzy logic was developed in 1965 as a method of capturing human expertise and incorporating it into ESs. There is still a significant amount of research being done in the field today.

Initially, the aim of AI was to develop universal models for general problem solving. However, soon it became apparent that this would rule out too many specific types of problems (Adeli 1990). Compared to other branches of engineering and computer science, AI is a very young field. Scientists and engineers are still struggling to find its rightful place in the scientific world. Although AI may not live up to early expectations, it can still benefit many applications.

## 2.2 Expert Systems

An ES is a computer-based system designed to emulate the problem solving behavior of a human who is an expert in a narrow domain. ESs are based on AI techniques that use knowledge about a particular domain and reasoning techniques to perform activities normally done by human experts. They are created by first capturing the domain expert's knowledge and then translating and storing this knowledge in a computer readable format, discussed in more detail below. ESs emulate the reasoning process of a human expert, sometimes reasoning from imprecise and uncertain information. They solve real-world, complex problems, using a computer model of expert human reasoning. These systems provide a way to make the knowledge of experts available to many.

Research in ESs began in the middle 1960s. There is an extensive body of knowledge covering a number of different approaches to ESs. Research and development for these systems is continuing at a rapid pace. There are many ESs, either in use or under development, for providing assistance in diverse applications such as:

- Performance monitoring and diagnosis of telecommunication systems
- Soil analysis
- Decision aiding for military maneuvers
- Data quality assurance in the Consumer Price Index
- Computer hardware configuration for large computer systems

A number of other applications related to avionics are examined in section 4.

Research in the area of languages to support symbolic reasoning has been done in conjunction with AI and ES research. LISt Processor (LISP), which is currently the most widely used AI language, was developed in 1958. Another widely used AI language is PROLOG (PROgramming in LOGic). Additional languages used in AI system development can be found in section 3.

Although there are other methods of encoding knowledge, ESs generally encode knowledge through a set of rules that are in the form of IF THEN statements. ESs compute a sequence of strings representing the steps in the solution to a problem. To simulate the human reasoning process, ESs apply specific knowledge and inferences. They are able to deal with a wide range of problems, as would a human expert.

High performance of ESs can only be achieved on powerful computing platforms because a great number of rules are needed in the knowledge base.

### 2.2.1 Expert System Architecture

The main sections of an ES are a knowledge base, a database, and an inference mechanism. An interface, whether human or with another system, must also be included. Figure 2.2-1 shows the ES architecture. The database contains the current state of the ES as it works toward a solution. The knowledge base contains domain knowledge which is applied to the problem at hand. Information in the knowledge base may be in the form of statements, such as the (attribute) of (object) is (value), or it may be in the form of an IF THEN statement (rule). Attributes are terms such as age, distance, or temperature, and they may not be known precisely (Togai and Watanabe 1992). The inference mechanism, or inference engine, controls the use of the knowledge base and database in solving the problem.

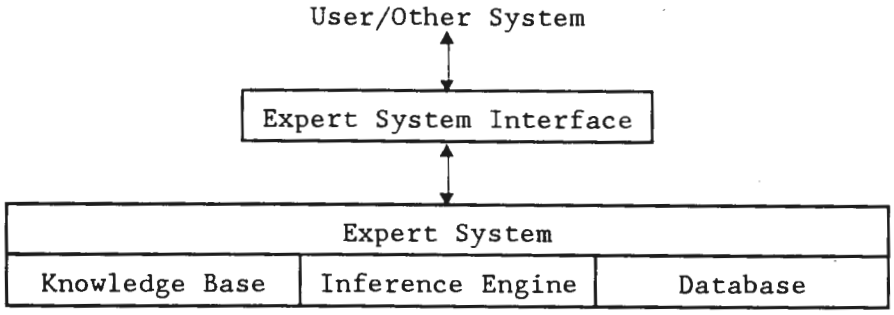



FIGURE 2.2-1. EXPERT SYSTEM ARCHITECTURE

#### 2.2.1.1 Knowledge Base

The knowledge base contains codified knowledge about a specific domain. It is a collection of facts and heuristics. The power of an ES depends more on the quality of its knowledge base than on the nature of its inference engine. Large knowledge bases can be difficult to develop and maintain.

#### 2.2.1.2 Database

The database contains an organized collection of specific data concerned with the problem to be solved. It represents what the computer believes to be either a partial or full solution to the problem. The inference engine acts as a gatekeeper between the database and the program using it. The inference engine is responsible for adding and deleting beliefs and performing certain classes of inference, such as adding facts or handling requests for information (Charniak and McDermott 1985).

#### 2.2.1.3 Inference Engine

The function of the inference engine is to use knowledge in the knowledge base, along with acquired knowledge about the specific problem, to form an expert solution. To achieve defined goals, the inference engine controls and executes reasoning using the knowledge contained in the database and the knowledge base.

The system's general problem solving knowledge is contained in the inference engine.

The inference engine must be able to determine when it is appropriate to ask the user for information and when it should consult the knowledge base and database. The inference engine needs to be flexible to deal with varying situations. It also may be required to handle imprecise and ambiguous information.

A well designed inference engine will be able to make use of many different patterns of evidence and use these to narrow the scope of possible conclusions. Various search techniques are used by the inference engine and applied to the knowledge base and database to reach a solution. Search techniques are examined later in this report.

### 2.2.2 Expert System Implementation

ESs that are most accepted by users and are most successful have a number of common qualities (Hall and Kandel 1992):

- Problem domain has limited focus.
- User interaction is minimal.
- Problem domains are well defined and manageable - the more limited and focused the domain, the less knowledge is needed and the less information that must be verified and tracked.
- Experts are available for consultation.
- Solution verification is straightforward and noncontroversial.
- The systems are not consultation systems, i.e., systems that claim to give indisputable expert advice.

ESs that have not gained wide acceptance by users typically do not have a good user interface, are slow in responding, and often ask too many questions of the user. Many unsuccessful systems are consultation or diagnostic ESs that do not explain their problem solving processes to the user's satisfaction.

Rolston (1988) categorizes current ESs into three general classes:

- Assistant - Performs a technically limited subset of an expert's task; however, generally it is an economically worthwhile system.
- Colleague - Performs a significant subset of an expert's task.
- Expert - Performs at an expert's level within a given domain. This type of system requires a large knowledge base and powerful development tools.

Most systems fall into the assistant or colleague class. It is difficult to design a system that is capable of replacing an expert in a complex domain.

### 2.2.3 Explanation of Reasoning

The ability of an ES to justify to the user exactly how it came to a particular conclusion is important. The user will gain trust in the system more quickly if there is an understanding of how a conclusion was reached. Also, errors, inconsistencies, and omissions are more easily detected in the debugging process if the path of reasoning can be traced.

### 2.2.4 Knowledge Representation

Knowledge is the fundamental part of an ES and is where the system achieves its power to solve problems (Rolston 1988). The effective representation of domain knowledge is considered the key to the success of AI programs. When choosing a form of knowledge representation, an important consideration for the designer should be the ease with which knowledge can be changed and updated. Flexibility is essential to handle input from human experts who often may change their minds. It is necessary to model a knowledge domain accurately.

Knowledge representation language is the method used to represent facts. Knowledge is defined in terms that can be manipulated by programs. Two common representation languages are natural language (English language sentences) and symbolic (representations of objects at the knowledge level defined in terms of symbols). Basic criteria for a knowledge representation language are as follows (Fikes and Kehler 1985):

- Expressive power - Experts must be able to communicate their knowledge effectively to the system.
- Understandability - Experts must be able to understand what the system knows.
- Accessibility - The system must be able to use the information it has been given.

The knowledge representation must facilitate inferencing, which converts the explicit set of beliefs about a problem domain to a larger effective set of beliefs. Three main ways of representing knowledge in an ES are rules, semantic networks, and frames.

Rules are a formal method of representing strategies, recommendations, and directives. Years of problem solving experience in a particular domain result in empirical associations that are suitable for representation by use of rules. An inference engine sorts through the rules and available data to reach a conclusion.

Semantic networks originally were developed to be psychological human memory models. Their use is now commonplace in the AI field. The network represents knowledge by using nodes and arcs. Nodes can represent objects, concepts, or events, while arcs are used to represent relationships or hierarchies between the nodes.

Frames are similar to semantic networks in that both nodes and arcs are used for representing knowledge. For frames, however, the nodes can take on a collection of attributes with their associated values. Each attribute can have attached procedures that execute when a value is changed.

2.2.4.1 Rule-Based Systems

Rules are a subset of predicate calculus with additional components to signify the way information in the rules should be used for reasoning (Fikes and Kehler 1985). Predicate calculus represents knowledge about real-world facts as statements.

Most current AI technology is rule-based. Rule-based systems constitute the best presently available means for codifying the problem solving methods of human experts. They are the most commercially successful AI products.

Rules are the most popular and effective representational form for behavioral knowledge in knowledge systems. Rules are fairly simple to work with and are relatively independent of each other. This independence allows incremental construction of AI systems. Rules also are able to represent different types of knowledge. Experts find it easy to state their knowledge in the form of rules. Rules lend themselves to expressing the heuristic knowledge that comes from experience (Friedland 1985).

Experts generally express most of their problem solving methods in terms of situation-action rules. Rule-based systems can incorporate rules that imitate the way experts reason. Rules must allow one to encode an expert's knowledge as thoroughly and efficiently as possible, and the rule-based system must be able to convey its knowledge and reasoning schemes to people of varying skill levels. Other types of systems lack the reasoning schemes used by experts and, therefore, cannot solve practical problems efficiently.

All rule-based systems have several properties in common (Hayes-Roth 1985):

- They use IF THEN rules to incorporate human knowledge.
- They use larger knowledge bases to perform at a higher skill level.
- They have the ability to solve a wide range of problems by combining rules appropriately.
- They determine the best sequence of rules to pursue.
- They can explain their reasoning by retracing their steps.

The rule base contains an expert's knowledge that is used to solve a problem. A rule-based controller models the behavior of the human expert, as opposed to modeling the problem solving methodology. This is a flexible approach to codifying knowledge and is well-suited for representing the knowledge used to control a system.

The usefulness of a rule-based system is dependent upon the fidelity of knowledge and problem representation. Problem solving using a rule-based system goes through many cycles of identifying the rules that have bearing on a particular problem and then applying those rules to solve the problem or achieve the end goal.

Rules can express the following (Hayes-Roth 1985):

- Deductive knowledge - Logical relationships, inference, verification, and evaluation.
- Goal-oriented knowledge - Used in seeking problem solutions.
- Causal relationships - Used to determine possible causes for specified events.

Control rules represent the expert's problem solving strategy. These rules are a way to achieve control over the system while maintaining clarity. There are two types of control rules: those that represent knowledge and those that represent the problem solving strategy. The rules that represent the problem solving strategy are a special type of ES knowledge called meta-knowledge. Basically, meta-knowledge is knowledge about the system knowledge. However, meta-knowledge often can worsen system transparency by complicating testing and making maintenance difficult. System transparency means that the system's knowledge is dependant neither on its location in the system nor on any of the other knowledge found in the system. Further research is needed so that ways to better represent meta-knowledge can be identified. For applications that are more than trivial, transparency and simplicity of the code seem to diminish (Coats 1991). Separating declarative and procedural knowledge makes the system easier to code, debug, and maintain.

If rules are transparent, their meaning is completely independent of their location in the rule base. However, this means the order in which the rules are processed is difficult to determine by examining the rules themselves. Conventional programming has low transparency but high behavioral visibility. Conventional programming controls the way programs process information. Controlling the way programs process information is detrimental to rule-based programming since it creates knowledge bases that are difficult to read and maintain. ESs are programmed to tell the system what to know, while conventional programming instructs the system what to do.

Sometimes rules may include procedural content. They represent knowledge of procedures and methods to process data. Generally, there is little information in the knowledge base about when rules should fire (execute). In theory, using rules to represent procedural knowledge can be risky because it can cause knowledge to become intertwined. When this occurs, it becomes difficult to follow the steps taken to solve the problem. The rule base and the inference engine which processes the rule base should be separate. Global control strategies should be context free. The selected control strategies may be applied regardless of the situation.

When procedural content is included in the rules, they become less independent of each other and their position begins to play a crucial role in producing correct results. Proceduralizing among the rules as opposed to within the rules also should be avoided. Rules should not be grouped together to imply procedures as a group. Rules should reflect knowledge, not procedures, and avoid implicit or hidden meanings. A particular problem may be more suited to traditional programming techniques when procedural content within the rules cannot be avoided.

When debugging a conventional program, normally the challenge for the programmer is in determining which statements are out of order. The sequence of events is followed easily. In an ES, the challenge is to determine how the inference engine is processing the rules, ensuring that the rules are correct, and adjusting the inference engine's method of processing the rules.

Conventional computer programming requires programmers to think procedurally, while rule-based programming requires programmers to think more analytically. Rule programming is similar to conventional programming because it requires mental modeling of state changes, syntactic and semantic checking of rule conditions, and heuristic methods for validating and verifying a proposed system (Hayes-Roth 1985).

As with conventional computer programming, structure is critical. A well-structured rule base is easier to test and maintain and better represents knowledge. In classic rule-based ESs, an inference engine processes a knowledge base composed of rules. The separation of the knowledge base and the processor of the knowledge makes the rule-based ES easier to develop, test, and maintain. Because of this, rules can be manipulated without affecting processing or other rules.

Automating expertise in specialized tasks generally requires a few hundred to a few thousand heuristic rules (Hayes-Roth 1985).

A rule-based system consists of three elements:

- A working memory (database) - Contains data structures that represent the current solution state of the system or intermediate results. Contents change until the final solution is found.

- A set of rules (knowledge base) - Contains IF THEN statements from rules and facts. This is a permanent, static structure.

- An inference engine - Contains hardware and software that matches the conditions in the rules to the entries in working memory and fires the rules if there is a match.

Two useful characteristic features of rule-based systems are incremental development, which allows knowledge to be refined and added, and explanation of reasoning, which aids in understanding computer systems.

Guidelines for a well-structured knowledge base are given below (Pederson 1991):

- Keep conclusions simple - Rules should only update one attribute at a time; the results are clear and easy to follow; and it is easy to tell which rule is responsible for which action.

- Keep rules free of procedural content.

- Minimize the use of ELSE constructs.

A poorly structured rule-based ES would be analogous to excessive use of GOTO statements in conventional programming. It is difficult to follow the path that the program is taking, and, therefore, more difficult to debug. Rules with multiple conclusions or the use of ELSE statements result in a process that is confusing to follow.

To be successful, a rule-based ES should have the following characteristics (Weiss and Kulikowski 1984):

- The conclusions must be generated by the system from a finite set of discrete and prespecifiable elements.

- The evidence about the problem must be obtained reliably by the user of the system or the system itself.

- The initial assumptions must limit the problem to a highly specialized area.

- A knowledge base must be available to link evidence about the problem to conclusions.

- A reasoning control strategy must be designed to guide the reasoning of the system and make its output correspond to that of a human expert.

The types of knowledge that can be included in rule-based systems follow (Hayes-Roth 1985):

- Inferences that follow from observations
- Generalizations of data
- Conditions for achieving a goal
- Best places to find relevant information
- Best strategies for eliminating uncertainty and minimizing risks
- Probable causes of symptoms
- Likely outcomes of hypothetical situations

Future goals for rule-based systems include the following:

- Increasing the size of the rule base to 10,000 or more rules
- Increasing the processing speed
- Using more inference techniques
- Improving reasoning with uncertainty
- Simplifying the process of creating and extending knowledge bases
- Sharing knowledge bases

- Improving architectures
- Generating more efficient ES development tools

Rule-based systems have the ability to capture, represent, store, distribute, reason with, and apply human knowledge. They are a practical solution to building automated experts and are especially applicable in tasks that require consistency and practical experience. Although AI developers have generated other approaches, the rule-based method is the only one that consistently produces accurate results. Rule-based systems offer advantages that cannot be found in traditional programming techniques (Hayes-Roth 1985):

- Modularity
- Knowledge bases that store rules and facts and make decisions
- Explanations of results
- Easily understood beliefs and problem solving techniques
- Inference chains (the path of reasoning and searching that the inference engine follows to achieve a goal) that are efficient

2.2.4.1.1 Rule Architecture

Rules basically are independent pieces of knowledge. As the rule base gets larger, rule components (antecedent, conclusion) must be flexible and independent enough so that extension and maintenance of the knowledge base will not adversely affect them (Hayes-Roth 1985). The rules are in the form of IF THEN statements and may contain the knowledge of one or more experts. The inference engine manages the rule base by using search expressions or algorithms.

The left side of the rule is the situation recognition part, known as the antecedent or premise. The right side is the action part, referred to as the conclusion or consequent. The antecedent expresses some condition in the state of the database and describes if and when it is satisfied. Usually it consists of a sequence of clauses connected by AND or OR, which serve as minimum and maximum operators. The action part specifies what changes are to be made to the database when a rule is satisfied. The inference engine monitors the facts in the database and when an antecedent is satisfied, it executes the corresponding action.

The current facts, along with the inference engine, make up the computing environment. Together they serve to interpret the current state, understand the meaning of the rules, and apply the rules appropriately.

When the consequent defines an action, scheduling the action for execution satisfies the antecedent. When the consequent defines a conclusion, inferring the conclusion will satisfy the antecedent.

There are several functions that the rules perform (Hayes-Roth 1985):

- Simplify auditing and explanation by having the ability to trace every result to its antecedent and intermediate inferences.
- Simulate reasoning by expressing logical relationships.
- Simulate human understanding.
- Simulate human decision making by using conditional rules to express heuristics.

The rules must cover all possible combinations. If no antecedents are satisfied, then no action will be taken.

2.2.4.1.1.1 Reasoning with Inexact Concepts

Certainty Factors (CFs) are values that represent the level of belief associated with a fact or a rule. They are a means of providing a judgement about the certainty of a conclusion, which is determined by combining degrees of belief and disbelief. The CF is based on the level of certainty known about conditions in the premise. Every time a new rule is considered, a new CF is calculated. The use of CFs is a technique for supplementing existing reasoning processes with information regarding uncertainty. Normally, an interval between 0 and 1 is used to represent the CF, with 1 being absolute certainty.

2.2.4.2 Semantic Networks

A semantic or associative network is a way of representing knowledge. Nodes represent objects, concepts, or situations in the domain. Arcs (links) represent relationships between the nodes. A network is a representation of the relations between elements in a domain using interweaving and crisscrossing arcs (Rolston 1988). A semantic network is a graphical representation of a network. Figure 2.2-2 shows the semantic network architecture. Arcs can also have weights, which indicate the strength of the relationship between the nodes.

Arc labels indicate the basis of the relationship between the two arcs. An arc can be viewed as something that is asserted to be true about one element relative to another (Rolston 1988). For example, "DC-10" is linked to "aircraft."

Semantic networks are easy to read. They are based on an object-oriented illustration, meaning the representation is based on objects that are treated as independent pieces of knowledge in the computer. Semantic networks are represented in a computer by using memory addresses to depict the arcs that go from one node to another. This facilitates access to all parts of the semantic network.

2.2.4.3 Frames

The organization of frames is similar to that of semantic networks. Frames consist of a set of slots that contain data, procedures, or pointers to other frames. The pointers can create nested frames. When slot relations warrant some

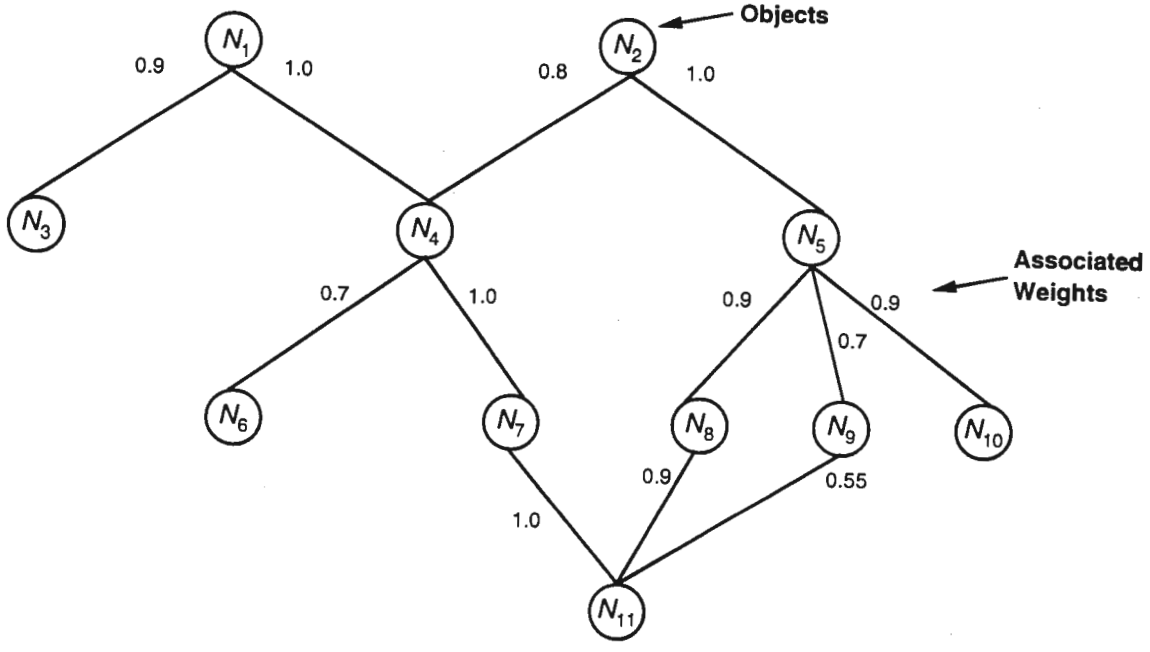



FIGURE 2.2-2. SEMANTIC NETWORK STRUCTURE

action, the procedures will define the type of action to be taken (Hall and Kandel 1992). Frames provide structured representation of an object or class of objects. Table 2.2-1 illustrates the frame architecture. Frames are well-suited for applications in which complex descriptions are needed to depict the domain accurately. They provide the knowledge base builder with a means of describing the types of specialized objects that the system must model.

TABLE 2.2-1. FRAME STRUCTURE

| Frame Label | Facette 1 | Facette 2 | Facette 3 |
|---|---|---|---|
| $slot_1$ | $value_{11}$ | $value_{12}$ | $value_{13}$ |
| $slot_2$ | $value_{21}$ | $value_{22}$ | $value_{23}$ |
| . . . | . . . | . . . | . . . |
| . . . | . . . | . . . | . . . |
| $slot_n$ | $value_{n1}$ | $value_{n2}$ | $value_{n3}$ |

Frames provide the ability to interpret new situations on the basis of knowledge gained from similar situations. Often, in the beginning of frame development, a best fit frame must be chosen for the current situation because there may not

be an exact frame that fits at that time (Rolston 1988). As development continues, frames may be added and/or changed.

Frames are appropriate for representing prototypical knowledge and the properties of objects to be modeled. Frames identify specific pieces of knowledge that belong together in a group. Frames are made up of pieces of knowledge that are typical characteristics of the ideas associated with that particular frame label. The pieces of knowledge in the frame are called facettes and they take on values that uniquely identify the characteristics of a specific object.

For example, a particular frame may be labeled Aircraft Accident. The facettes that may be contained in that frame include components normally associated with any aircraft accident:

- Cause
- Number of passengers
- Number of crew members
- Number of injuries
- Location
- Date
- Time

For an aircraft accident, these facettes are filled with values that can identify a specific accident. When the values are entered into the facette, the result is a particular instance of the general concept Aircraft Accidents represented as a frame (Adeli 1990).

Frames are an excellent way for a knowledge system to organize and direct its reasoning activities. As with semantic networks, frames are an object-oriented form of representation. The use of frames, along with rules, may be a promising area for improvements in knowledge representation. When representing knowledge, rules and frames integrated in a single unified representation scheme work best.

### 2.2.5 Updating the Knowledge Base

Machine learning is a process by which a system generates and updates its knowledge base. The system learns from past experience. This is the ultimate goal for updating a knowledge base. This area, however, has not been perfected and requires more research.

Knowledge also can be updated manually. This process is performed by a knowledge engineer who interprets and encodes knowledge from a human expert. Another form of updating knowledge is for the expert to enter knowledge directly into the system with the assistance of a tool designed to perform the knowledge engineer's tasks. The knowledge engineer is eliminated.

### 2.2.6 Chaining

Chaining refers to the path of rules that the inference engine fires to achieve a solution. The inference engine can process information in one of two ways: forward or backward chaining. Some ESs, such as blackboard systems, use a combination of the two methods.

#### 2.2.6.1 Forward Chaining

Forward chaining uses bottom up or event driven reasoning. A rule is triggered when changes in working memory data produce a situation that matches its antecedent component. This method begins with some initial data and moves down the inference chain until it reaches a solution to the problem. Working memory is used to store the new facts. Working memory also stores initial observations, findings, and conclusions.

#### 2.2.6.2 Backward Chaining

Backward chaining is the process of working backward from a known conclusion to find a path of reasoning to justify the conclusion. The goal is assumed to be true and the inference engine searches for evidence to support that conclusion. The rule-based system begins with a goal and successively examines any rules with matching consequent components. The unmet conditions of the antecedent are taken from each applicable rule and then defined as new goals (Hayes-Roth 1985). In backward chaining, working memory is used to note unresolved subgoals. This is also known as goal directed, or top down, reasoning.

### 2.2.7 Knowledge Acquisition

Knowledge acquisition is a lengthy process in which the knowledge engineer must meet with a known expert who is willing to provide information. It is a tedious process and creates a bottleneck in ES development. It is one of the most difficult phases of ES building. The knowledge acquisition process is not well understood or well defined (Rolston 1988).

Knowledge can be represented as procedural or declarative. Most declarative schemes are knowledge represented as static facts, along with a limited amount of information that describes how the information should be used.

An ES shell is an ES with an empty knowledge base. ES shells require knowledge engineering that extracts a collection of highly detailed facts from a human expert and programs them into a database. This process requires repeated testing of the program to verify and account for all possible situations.

Both procedural and declarative knowledge are needed for every ES. However, they should be kept as separate as possible. Generally, procedural knowledge is found in the inference engine and declarative knowledge is found in the rule base.

The inference engine utilizes meta-knowledge. Meta-knowledge includes information on how to utilize available knowledge, in what order the knowledge should be used, where the knowledge can be obtained, and the point at which processing is complete.

Knowledge engineering can be an expensive, time consuming, and difficult task for the following reasons (Adeli 1990):

- Vocabulary - The knowledge engineer must comprehend the basic vocabulary of the domain.

- Completeness - The knowledge engineer must understand the domain sufficiently to identify pieces missing from the knowledge base.

- Integration - New information must be added to the knowledge base in a way that will not adversely affect existing knowledge.

- Analysis - The knowledge engineer must have the ability to evaluate the expert's methods for arriving at conclusions.

- Transparency - The knowledge engineer must incorporate the expert's problem solving behavior into the knowledge base.

Knowledge engineers need to have a working knowledge of the domain that is being modeled. They need to become proficient in the field to be able to comprehend the expert's information and to determine how to handle situations. For example, the knowledge engineer must determine how to handle contradictions and inconsistencies between experts. The knowledge engineer may need to resolve conflicting, redundant, subsuming, or missing blocks of knowledge.

Several tools have been developed to assist knowledge engineers with prototype development. Some tools may replace the knowledge engineer with an automated acquisition system. Any system intended to replace the knowledge engineer must be able to perform all knowledge engineering tasks.

### 2.2.8 Applications

Applications suitable for ESs are presented below:

- Applications where a proven expert is available.

- Systems that entail robot activities.

- Limited domains with well defined expertise.

- Systems that do not require extensive user interaction.

- Problems that do not have a large amount of uncertain information. (An ES will no longer be appropriate when a specific level of uncertainty is reached.)

The use of an ES may be justified when experts are rare or soon will be unavailable, identical expertise is needed in many locations, or the expertise is needed in a hostile environment.

If the problem to be solved can be described by direct definitions and algorithms, it may be solved better by traditional software methods. In addition, if the problem is extremely difficult to define or requires intense judgement, it may be too complex for an ES and will need to be solved by traditional methods (Rolston 1988).

A number of characteristics are common to ESs in use today:

- There are proven experts available.
- The task requires a well defined domain without changing rules.
- The problem requires more cognition than physical attributes.
- The problem takes an expert a few minutes to a few hours to solve.
- The experts teach their skills and are able to explain their reasoning.
- Solving the problem will produce considerable benefits.
- The problem solution requires no common sense.
- There is low risk if a bad decision is made.

An ES attempts to find an acceptable, but not necessarily the best solution, in a reasonable amount of time. It deals with a specific problem domain, as opposed to attempting to mimic human behavior in all domains (Rolston 1988).

### 2.2.9 Heuristic Reasoning

An heuristic is any rule of thumb or strategy that is used to limit the time required to search for solutions in large problem areas (Coats 1991). Heuristics is the study of the processes involved in solving a problem. Heuristic reasoning is reasoning in the form of general rules, hunches, or rules of thumb that are approximate and generally have been acquired through years of experience. Heuristics contribute to an ES's power and flexibility and set them apart from traditional software.

Heuristic reasoning can provide direction to a search process, thereby reducing the area in the knowledge base that must be searched. It can assist in processes that include eliminating entire branches of a search tree, selecting a general path to follow, and selecting the next node to expand.

Heuristic reasoning is used in the absence of more precise control mechanisms because it uses rules of thumb that are meant to produce acceptable results in most cases. A drawback of heuristic reasoning is that it could lead a search down the wrong path.

The technique of programming used by most ES programmers is referred to as heuristic programming. The exact behavior of the system may not be precisely specified or predictable by the programmer (Denning 1986). These are techniques that can only be acquired from years of experience in a specialist domain. This type of knowledge tends to be inexact, uncertain, and incomplete. Most of the time, heuristics offer reasonable but not necessarily the best solutions.

### 2.2.10 Searching

Occasionally, more than one rule may be eligible to fire at the same time. A methodology to resolve these conflicts must be established. The most systematic ways of resolving conflicts are breadth first and depth first searching. They can be used with either forward or backward chaining. Both of these search techniques can be used with rule-based systems.

### 2.2.10.1 Breadth First

In a breadth first search, all possible subgoals on one level are considered before the next level is examined (Adeli 1990). Increasingly broad segments of the solution space are created and each level is checked for a goal state. All possible successor nodes are created from each state by applying applicable operators to the existing nodes. This ensures that the nodes at a given level are checked before the next level is addressed. Figure 2.2-3 shows the breadth first search strategy.

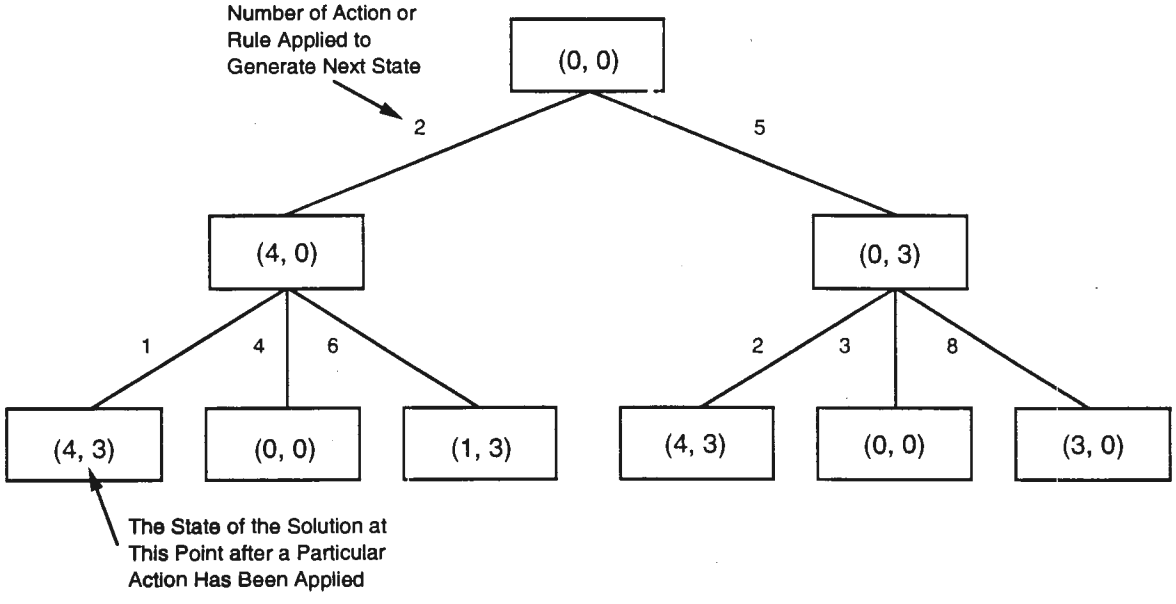



FIGURE 2.2-3. BREADTH FIRST SEARCH TREE (Rich and Knight 1991)

### 2.2.10.2 Depth First

Depth first searching entails pursuing each branch of a tree completely before considering another branch. When a goal or a subgoal matches the left side of a rule, all premises of that rule are examined before another subgoal is considered. This technique takes one idea and pursues it until a goal is reached or the limit of that branch of the tree is reached (Adeli 1990). The choice of the node used in continuing the path may be based on random selection. Figure 2.2-4 shows the depth first search strategy.

## 2.2.11 Expert System Drawbacks

The purpose of an ES is to perform a difficult task or resolve a substantial problem. It is rare for an ES to outperform a human expert and, if it does, it is because the ES forgets less rather than remembers more. Problems that are suited for ESs may require long development and testing times and can be costly.

A number of problems exist with ES methodology, code, knowledge acquisition, and validation (Coats 1991):

- Methodology - The use of IF THEN statements can be clumsy and unsuitable to express knowledge effectively.

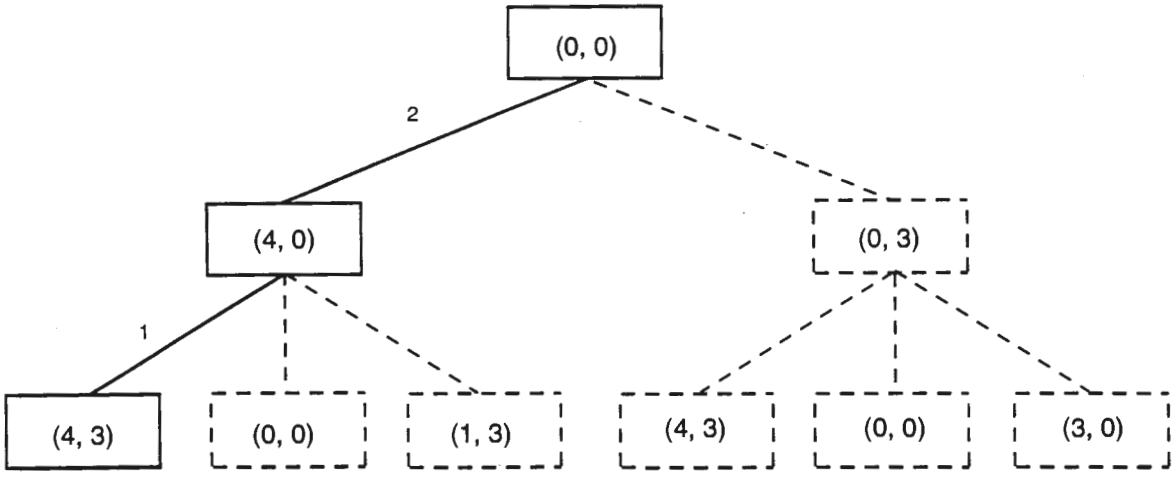



FIGURE 2.2-4. DEPTH FIRST SEARCH TREE (Rich and Knight 1991)

- Code - Can be difficult to understand, debug, or maintain, and does not always provide adequate user interface.

- Knowledge acquisition - It is not always clear how to resolve conflicting views among acknowledged experts.

- Validation - Methods for validation are still being researched. Perhaps the basic question should be "does the user of the ES make better decisions by using it?"

One main concern about ESs is reliability. Their effectiveness ultimately relies on the system developers. Very large databases are hard to maintain and it is difficult to account for all possible situations.

Unless ESs are designed to provide reliability and maintainability, they will not be capable of handling complex, real life situations. They need to be adaptable to deal with situations that lack complete information. Care must be taken to determine whether an ES is suited to the problem.

There are many influences in an ES domain that may introduce uncertainty into a problem (Hall and Kandel 1992):

- Characteristics of the solution to the problem
- Questions asked to determine the solution
- Knowledge acquisition process
- Reasoning process of an expert

- Knowledge representation language

### 2.2.12 Future of Expert Systems

Currently, interest in ESs is high and their applications are increasing. Relatively low cost computers are being developed with large memories, ample disk space, and faster processing capabilities. These capabilities will enhance the possibilities for developing ESs with broader applications.

ESs are most accepted if they critique conclusions or act as advisors. The goal of using an ES is to improve productivity and the quality of decision making. In the future, powerful software and hardware may be all that is necessary to develop a system. The ultimate goal will be to eliminate the role of the knowledge engineer and enable the experts to encode their own knowledge directly on the computer. Attention should be focused on methods of extracting and representing knowledge (Weiss and Kulikowski 1984).

### 2.2.13 Blackboard Systems

A blackboard system is a type of ES with a slightly different architecture. It is a multiple knowledge source system in which data are shared between sources. A knowledge source system is a program whose knowledge is in a separate database that can be manipulated by the user. The shared data provide communication and cooperation between the experts. The rules have access to all parts of the blackboard. The blackboard serves as the input to the rules and as a record of the outcome of the rules. The blackboard can display the best interpretation of a specific problem at any given time.

The blackboard is the controller for the entire structure. It records and organizes solutions that are generated during the problem solving process. Knowledge bases can communicate only through the blackboard and they can generate input to the incremental solution on the blackboard whenever they have relevant information. There is no priority or hierarchy established for knowledge bases to contribute to the blackboard, and the system is not committed to forward or backward chaining. It can implement whichever it sees fit.

The objective of each knowledge source is to contribute information that will lead to the solution of the problem. Each knowledge source must recognize when conditions are appropriate for its contribution to the solution. Each knowledge source has a separate inference engine. Some architectures contain a scheduler, which chooses the next knowledge source to contribute data.

Blackboard systems solve problems by dividing them into several smaller subproblems to be solved separately. The problem dividing method influences how well the problem is solved.

Advantages of a blackboard system include the following (Engelmore and Morgan 1988):

- Many knowledge sources participate in forming a solution.

- Each knowledge source has continual access to the current state of the solution and can contribute when needed.
- The solution is built incrementally.
- Both forward and backward chaining can be used.

The classic blackboard system architecture, as shown in figure 2.2-5, consists of the following (Cohen 1985):

- Several knowledge sources - Collections of production rules
- A database or blackboard
- A scheduler (optional) - An independent knowledge source that determines which knowledge source should be activated next to determine a solution

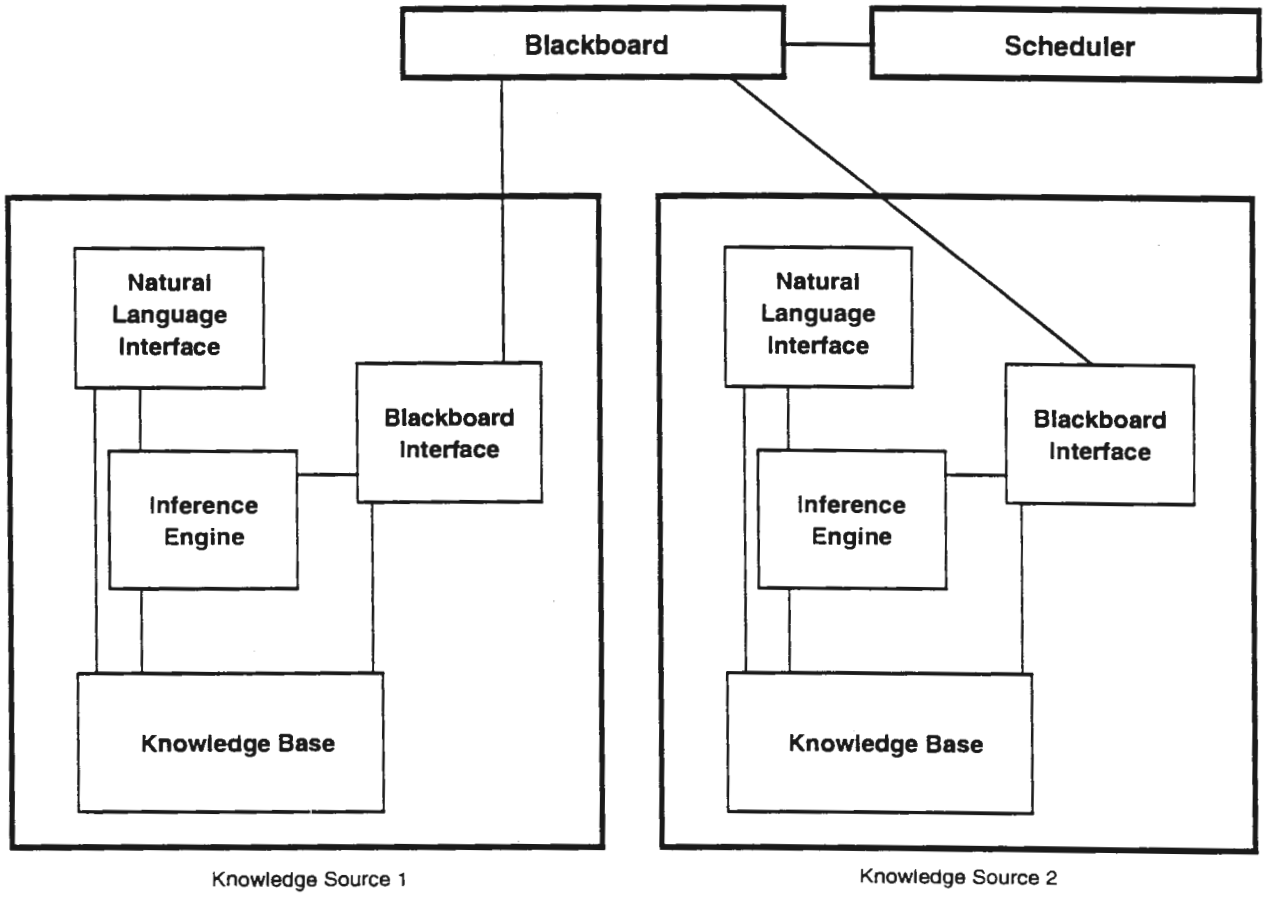



FIGURE 2.2-5. BLACKBOARD SYSTEM ARCHITECTURE

#### 2.2.13.1 Knowledge Sources

Each knowledge source is an ES in its own right, with an internal structure which can be independent from other knowledge sources. For example, the structure can be rule- or procedure-based, or can be homogeneous or heterogeneous. The form of the rules and the way knowledge is represented can vary from source to source (Engelmore and Morgan 1988).

#### 2.2.13.2 Blackboard System Control

Blackboard systems determine their own cognitive behavior to solve problems involving the control of another system. The system must determine the problems it will attempt to solve and the knowledge and methods it will use to solve the problems. The system goal can be choosing an area of the blackboard on which to concentrate next, choosing a particular knowledge source to utilize next, or some action between the two. Whichever method is chosen, it must be the most efficient for the particular application. Selecting irrelevant knowledge sources must be minimized. The control can be integrated into the system design or it can be a separate ES.

The blackboard is a record of the hypotheses reached and the way they interconnect. Attached to each hypothesis on the blackboard is a measure of likelihood, which is a measure of the system's confidence level in the hypothesis. The likelihoods of the hypotheses are updated continually. Changes made to the blackboard are called events. These include creating or deleting objects, changing values, and creating or destroying links between objects.

Blackboard architecture is a control structure design. The knowledge sources can communicate only through the blackboard, so the events are tracked easily. Blackboard models contain a great amount of information. Knowledge sources are not activated in any particular order. They contribute hypotheses in response to appropriate situations on the blackboard, or the scheduler may decide when they should contribute in response to the situation. They represent an opportunistic approach to solving problems that allows the knowledge bases to contribute only when appropriate (Cohen 1985).

Blackboard systems do not search for solutions; rather, they build an emerging solution from all independent knowledge sources. The presence of many different, independent knowledge sources leads to parallel processing architecture. Knowledge sources indicate to the blackboard when the problem solving process is terminated, either because an acceptable solution was found or the knowledge sources are depleted.

#### 2.2.13.3 Applications

Following are problem types that are well suited to being solved by a blackboard system:

- Problems requiring many different and specialized types of knowledge.
- Problems with a natural hierarchy of the problem solving strategy.

- Problems that will benefit from solutions developing incrementally.
- Problems where limited data are available or for which only a partial solution is possible.

2.2.13.4 Future of Blackboard Systems

More research is required to determine the types of applications that are suitable for blackboard systems. Also, further study of the characteristics of particular types of design choices for the systems is needed.

## 2.3 Fuzzy Logic

Traditionally, vague or imprecise concepts, such as tall, big, and warm, could not be represented on computers. Expressions such as these are used in our language constantly and the ability to represent these types of concepts is helpful when dealing with set membership.

The strength of fuzzy systems, and their reason for being developed, was their ability to manage imprecision. Fuzzy logic provides the opportunity to model conditions that inherently are imprecisely defined and deal with statements that are obscure or subject to different interpretations. It introduces a method of representing the degree to which an object is a member of a particular set (Miyamoto 1990).

There are many complex industrial processes that cannot be controlled satisfactorily by conventional computing methods. The use of rule-based control established on approximate reasoning provides an attractive alternative. Fuzzy logic was developed for complex control systems in which mathematical models were difficult or impossible to create. Fuzzy logic is an alternative to traditional set memberships. It was developed to express inexact concepts such as "normal height and weight." These concepts are used frequently in everyday language. However, they become a problem when attempts are made to express them using binary distinctions (Cohen 1985).

Fuzzy models provide a more flexible way to acquire accurate solutions to problems than the rigidness of crisp set solutions (Bezdek 1992). Data are represented symbolically, but they are processed numerically. Fuzzy logic is a precise and accurate method of reaching valid conclusions in control systems. Correctly implemented, it will sacrifice only unnecessary precision. Greater precision can be achieved by increasing the number of input and output values and increasing the number of rules (Brubaker[2] 1992).

### 2.3.1 The Basic Theory

There are few situations that are strictly true or false. Fuzzy logic proposes that membership functions operate over a range between 0 and 1, where 0.0 is total non-membership and 1.0 is total membership. This grade of membership is particularly helpful in expressing inexact concepts inherent in natural language. It is a method of representing analog processes on a digital computer.

The membership functions usually are graphed by using overlapping triangles; however, other geometric shapes are possible. Membership functions identify regions that will control the system. The purpose of overlapping shapes is to allow for partial set membership. The transition between being fully a member of the set and not a member at all is gradual. It is possible for an object to be a partial member of more than one set at the same time. Figure 2.3-1 illustrates a set of membership functions related to throttle movement. If the rotational speed of the throttle is -20 radians/second, it is a member of the set "Small Negative." The throttle's degree of membership within that set is the point where a vertical line intersects with the side of the membership triangle. That would be approximately .6 for a speed of -20 radians/second.

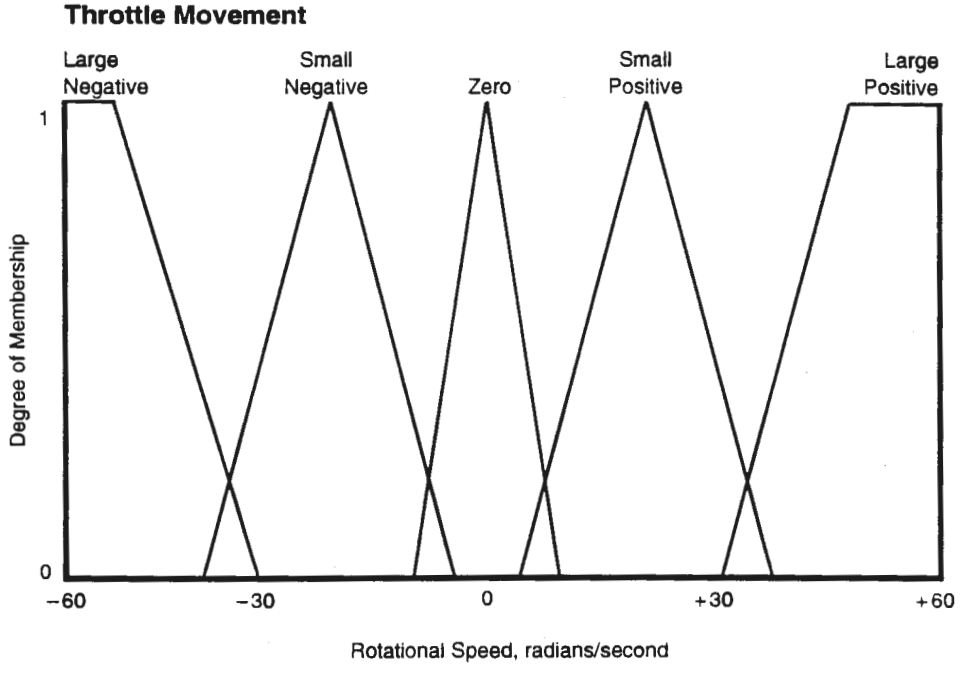



FIGURE 2.3-1. FUZZY MEMBERSHIP SETS

Fuzzy systems are based on the concept of parallel processing. Parallel processing allows large, complicated problems to be solved by breaking them down into smaller ones. The results of all rule processes are unified and expressed as a single logical sum. Compared to conventional control systems, fuzzy systems require fewer rules and reduce development time dramatically.

Rules are in the form of IF THEN statements that are linked by ORs. They use labels to describe the problem in words and are worded in a way that is similar to the human thought process. For example, a rule for controlling a flying aircraft may read "If nose is low and speed is high, then pull back on yoke."

One of the main advantages of using fuzzy logic is that modifications are made easily without the need to modify code. Rules and membership functions can be added without radical revisions. Therefore, designers can concentrate on the

elements of the problem, as opposed to the details of a language such as FORTRAN or C (Williams 1992).

Fuzzy controllers are more flexible than conventional controllers. If some element, such as information, is missing, a fuzzy system will adjust itself accordingly.

### 2.3.2 Crisp Logic

Traditional or crisp logic, such as Boolean, is based on predefined thresholds. For instance, in Boolean logic the output and input are always on or off; there are no other states. There are many situations that do not conform to Boolean logic. Fuzzy logic was developed for these situations (Lea 1989). The difference between crisp and fuzzy logic can be seen in figure 2.3-2.

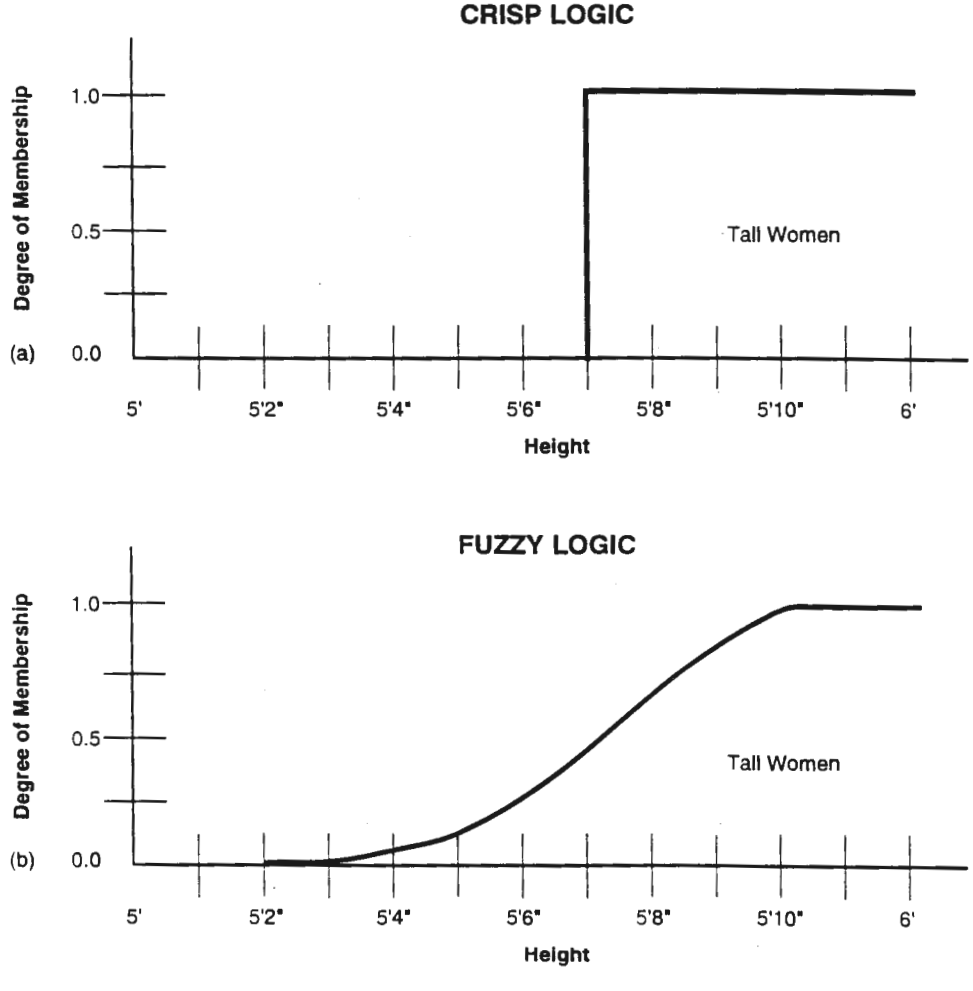



FIGURE 2.3-2. CRISP VERSUS FUZZY LOGIC

Thresholds in fuzzy logic are not clearly defined. Boundaries overlap each other and it is possible for an object to be partially a member of more than one set.

### 2.3.3 Fuzzy Control

There are three steps followed by a fuzzy controller. They are fuzzification, inference, and defuzzification.

#### 2.3.3.1 Fuzzification

A fuzzy system takes inputs and fuzzifies them. Numeric inputs are converted to fuzzy values, such as small, fast, or warm. This is done using degree of membership functions, which are methods of determining the extent to which an object is a member of a particular group. For example, a particular aircraft may be considered both a cargo and a passenger aircraft, which would qualify it as a partial member of both sets. It could be more a member of one than the other, but nonetheless a member of both to some degree. The resulting degree of membership then becomes an input to the inference mechanism.

#### 2.3.3.2 Inference Mechanism

Inference is the process of inferring fuzzy actions by applying fuzzy inputs to a rule base (Legg 1992). Fuzzy actions include "pull back on the yoke," effecting a climb rate of 100 feet per minute. Rule-based systems are the most popular type of fuzzy system at this time. The inference process proceeds from the conditions to the conclusion (logical product) and then to the logical sum (Omron 1991). The logical sum is a combination of the results of all the rules. Figure 2.3-3 illustrates how a fuzzy system determines the control actions to be taken for a particular example. The object is to balance a stick in the upright position using seven rules:

- If the stick is inclined moderately to the left and nearly still, move to the left quickly.
- If the stick is inclined slightly to the left and falling slowly, move to the left somewhat quickly.
- If the stick is inclined slightly to the left and rising slowly, do not move much.
- If the stick is inclined moderately to the right and nearly still, move to the right quickly.
- If the stick is inclined slightly to the right and falling slowly, move to the right somewhat quickly.
- If the stick is inclined slightly to the right and rising slowly, do not move much.
- If the stick is slightly inclined and nearly still, do not move much.

There are seven possible sets in each of the two position antecedent blocks. In the left/right position blocks, negative corresponds to leaning towards the right and positive corresponds to leaning towards the left. There are 3 degrees of membership in either direction: small, medium, and large. In the falling/rising

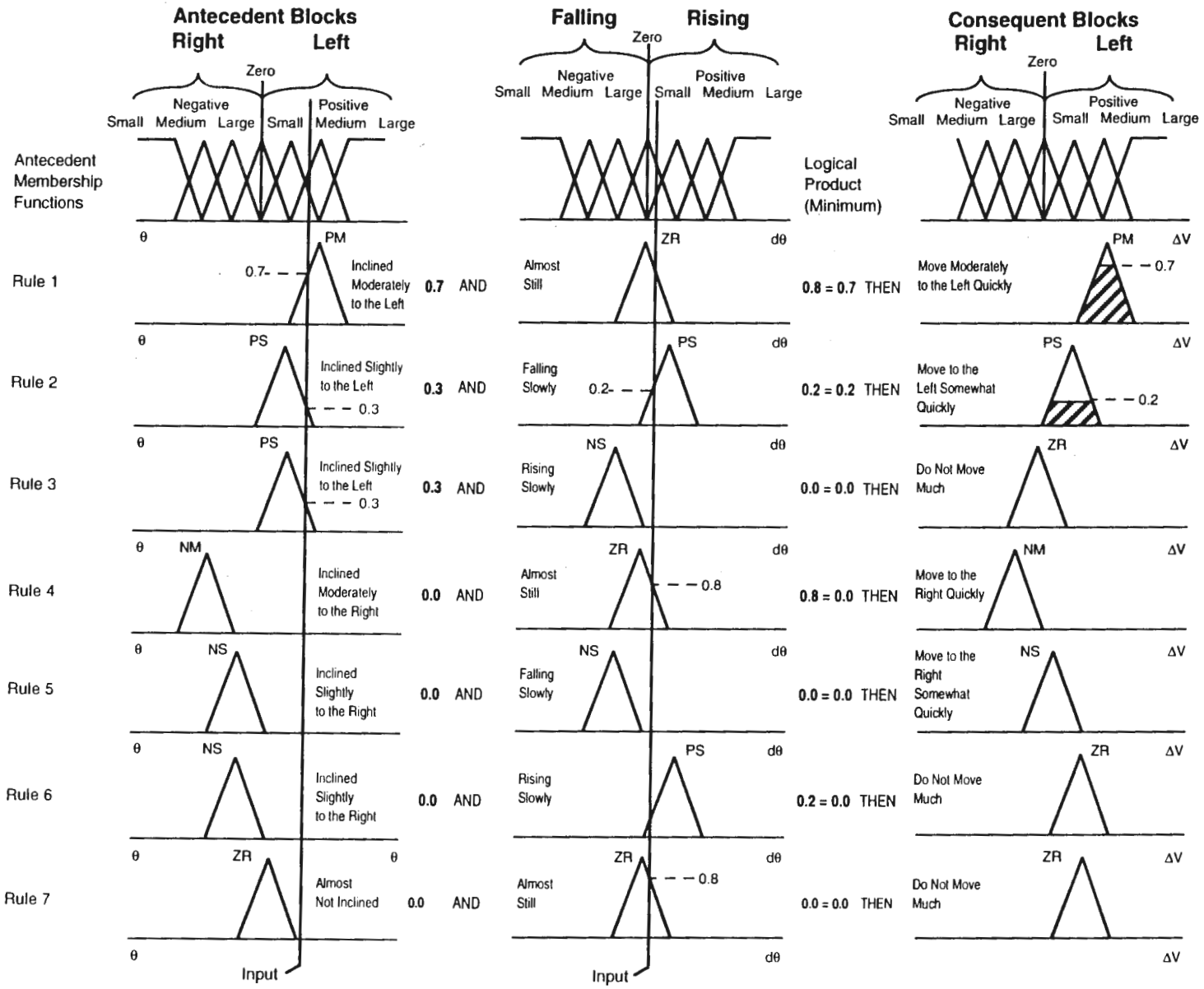



FIGURE 2.3-3. OBTAINING THE LOGICAL PRODUCT

position blocks, negative corresponds to falling and positive to rising. There are also 3 degrees of membership in either direction: small, medium, and large. Both position blocks have a zero where the stick is balanced.

The line labeled "input" is one instance when the stick is at a certain point and the system is attempting to determine appropriate action. The point is a member of several sets. Therefore, it must be determined which rules to apply to the input. This can be resolved by evaluating of which sets the input is a member. As far as left/right position, at this point the stick is mostly a member of the set "inclined moderately to the left." It is also partially a member of the set "inclined slightly to the left." As far as falling/rising position, at this point the stick is mostly a member of the set "almost still" and partially a member of the set "falling slowly."

By performing an "AND" operation on the membership functions, the consequent blocks for each individual rule can be determined. These values are known as the logical product. Figure 2.3-4 shows the logical sum, which is the total result of summing the consequent blocks. This total is defuzzified by determining its center of gravity, which, in turn, becomes the output of the system. The output is generally applied to another system to initiate an action.

The inference process determines the degree of truth of a rule. The corresponding action correlates to this degree of truth. The inference system is a collection of fuzzy IF THEN rules. The inference mechanism uses a method of approximate reasoning by which an imprecise conclusion is deduced from a collection of imprecise premises (Lee 1991).

#### 2.3.3.3 Defuzzification

Defuzzification is the process of translating fuzzy or non-crisp values into values that are crisp and clearly defined. A defuzzifier converts inference process values to a fixed value. Generally, a non-fuzzy control action, such as a fixed voltage applied to another system to cause an action, is needed as a result of the inference process. Normally, more than one rule applies to any given set of inputs, so the fuzzy system must combine the results of several rules.

### 2.3.4 Control Systems

Conventional control systems do not have the ability to self-adapt to changes in inputs. Adaptive fuzzy control systems can learn, explain their reasoning, and perform self-modifications, when necessary, to produce more accurate results. They can change supporting system controls, modify the characteristics of the rules, modify the topology of the fuzzy sets, and change the method of defuzzification. Their construction is similar to NN systems.

Fuzzy logic systems make excellent control systems. Controlling with IF THEN constructs is simpler than creating complex mathematical models. It is generally easier to create and understand fuzzy control rules than conventional control rules.

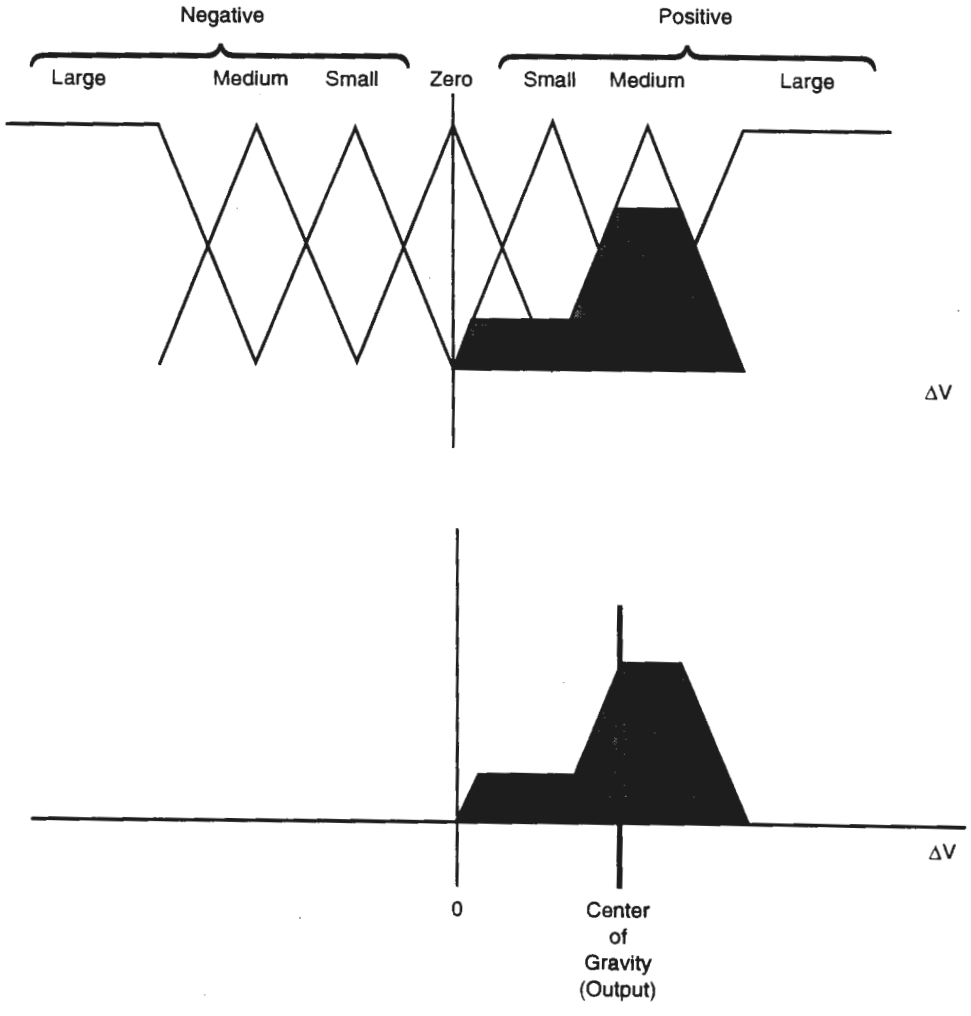



FIGURE 2.3-4. THE LOGICAL SUM
(as applied to the system in figure 2.3-3)

Fuzzy logic systems are designed to handle complex requirements using controllers that are simple, inexpensive, and easy to maintain. Because fuzzy systems use intuitive terms, complex systems are modeled easily and rapidly. A system's operational and control laws are expressed linguistically.

Fuzzy control systems can be cheaper, easier to develop, and more accurate than traditional control systems. Also, they can respond faster and consume less power than traditional digital methods. These factors provide the opportunity to bring products to market faster and more cost effectively.

### 2.3.5 Probability

Many people confuse fuzzy theory with probability. Both operate over the same numeric range, where 0.0 represents false and 1.0 represents true. A statement is probabilistic if it contains some likelihood or degree of certainty, i.e., the likelihood that an event will happen. Probability theory deals with the uncertainty that results from random behavior. The difference between probability and fuzzy theory is that probability states the chances of membership, while fuzzy logic states the degree of membership or the degree to which an event has happened. Fuzzy theory deals with the uncertainty associated with boundary conditions (Brubaker[1] 1992).

### 2.3.6 Parallel Processing

Fuzzy systems can use parallel processing, which allows multiple rules to be processed simultaneously. High-speed processing can be attained with hardware controllers because the rules are processed independently of each other. Errors will not drastically affect the final result in a parallel processing architecture.

### 2.3.7 Hardware

Fuzzy systems do not require massive computing power and many applications do not require special fuzzy Integrated Circuits (ICs). Normally, fuzzy processors work along with a general purpose microprocessor, which is needed for functions such as preprocessing inputs, keyboard interaction, and support in other input and output areas.

Simple fuzzy controllers require less than 10 rules, while complicated ones may require 40 or more. Some fuzzy processors allow arbitrary shapes for the membership functions. Some limit the number of shapes and others allow triangles only.

### 2.3.8 Designing a Fuzzy System

Fuzzy logic does not require that a designer be able to model a control problem mathematically. All that is required is a firm understanding of the problem, a basic understanding of physics, and a concept of how the system should behave. The designer is not restricted to what can be described with equations (Conner 1993).

The designer of a fuzzy system needs to identify a number of elements:

- The input, output, and operations to be performed on the data.
- The control and solution variables, fuzzy regions, and labels associated with those variables.
- The rules connecting inputs to outputs.
- The method of defuzzification.

Control variables appear in the premise of a rule (e.g., density, speed) and are linked by AND. Determining these variables and creating the rule base are the most difficult part of designing a fuzzy system.

Labels are names that are attached to fuzzy sets or membership functions (e.g., cold, warm, hot). The shapes and slopes of membership functions can be changed at any time to meet a design criterion. Overlapping input functions guarantees that at least two rules will fire for any given input and ensures a smooth output curve. Generally there are an odd number (between five and nine) of labels (e.g., slow, medium, fast) associated with each variable (Cox 1992). Membership function sets should be grouped closely at points where the most activity in the system being monitored occurs.

Rules in a fuzzy system normally are determined by an expert. They are stored in a rule base that is similar to the thought process that an expert would use to solve a problem. Because fuzzy rules are abstract, they do not need to cover every combination of cases that is possible. The rules are defined by the relation between antecedent and conclusion.

### 2.3.9 Fuzzy Logic Advantages

There are a number of misconceptions associated with fuzzy logic. Most of these are addressed in the following list of advantages (Cox 1992):

- Fuzzy logic allows partial or gradual degrees of membership. The concept that fuzzy logic provides imprecise answers is a common misconception.

- Fuzzy logic is different from probability. When dealing with probability, the event is clearly defined. Fuzzy logic is concerned with the ambiguity in the description of the event.

- Fuzzy sets are easy to conceptualize and they are modeled in the same fashion as an expert's thought process.

- Fuzzy systems are stable, easily tuned, and can be conventionally validated. There are fewer rules in fuzzy systems than in conventional control systems so tuning is simpler.

- Fuzzy systems are different from, but complementary to, NNs. Fuzzy systems are similar to NN classifiers. However, in a fuzzy system the classification processes performed on the data are more flexible for the developer.

- Fuzzy logic is more than process control. Fuzzy logic provides a way of representing and analyzing information, independent of particular applications.

- Fuzzy logic is a representation and reasoning process, not the answer to all AI problems. Fuzzy logic provides the opportunity to manage concepts that are outside of the realm of conventional Boolean logic.

### 2.3.10 Fuzzy Logic Disadvantages

Although fuzzy logic is a promising alternative to traditional control systems, there are a number of factors that need to be addressed:

- There is not a clear understanding of what applications are appropriate for fuzzy systems and how they compare to traditional designs.
- Fuzzy systems, when optimized for speed, use an amount of microprocessor memory comparable to traditional control methods.
- There is no formal design method or system modification and tuning method that will eliminate all unnecessary rules to find the optimum rule set. The systems are not analytic and system stability cannot be proven.
- Fuzzy systems can be designed more quickly than conventional control systems. However, they require more simulation and tuning to optimize their performance. Fuzzy systems may require the same amount of development time as traditional systems.
- Fuzzy systems may not be the best alternative for all control system applications because they cannot prove closed-loop system stability. This requires much simulation and testing. If traditional control methods are adequate for a particular system, there is no need to change methods. In general, if a system can be defined using conventional control equations, fuzzy logic is not necessary.

### 2.3.11 Applications

Types of systems that can benefit from fuzzy logic control include the following (Omron 1991):

- Non-linear systems
- Systems with insufficient or unclear data input
- Systems that are difficult to control
- Systems that benefit from or are normally controlled by human intuition
- Systems that require adaptive signal processing because of dynamic environmental conditions
- Systems with multiple inputs or conflicting constraints
- Systems for which models are unclear or difficult to define

Fuzzy logic controllers have been used successfully in a wide arena of applications. Items such as car cruise controls, washing machines, elevator controllers, vacuum cleaners, subway engine controllers, and security investment systems have benefitted from fuzzy systems (Schwartz and Klir 1992).

Fuzzy logic systems are well-suited to model very large systems, natural or man-made, such as weather, oceans, economy, and stock markets (Brubaker 1993).

### 2.3.12 Future of Fuzzy Logic

Designing with fuzzy logic is regarded with skepticism, in part due to its close association with AI. Another reason for skepticism may be distrust of a new technology that is perceived as revolutionary. Even its unusual name has added to the skepticism.

However, when compared to traditional system development, fuzzy systems can be designed more easily than systems using a linear control approach. Also, the basis of fuzzy design is easier to learn than linear methods.

Japan and Europe already utilize fuzzy technology in practical systems. In the United States, companies are beginning to turn to fuzzy logic as a viable option to solving many types of problems. Competition will encourage more involvement in this new technology.

Replacing traditional control methods with fuzzy systems may expand a system's capabilities. Just as the microprocessor revolutionized the electronics industry, fuzzy logic has the potential to do the same for the control industry.

## 2.4 Neural Networks

NNs are information processing systems that can be trained to solve problems. Artificial NNs are constructed to resemble the biological neurons of the brain. Hundreds of thousands of simulated neurons are linked together by synaptic connections. The structure of an NN is shaped and reshaped, depending on associations among facts.

NNs have been described as the second best way of solving a wide range of problems. They are capable of solving virtually any problem that involves mapping input data to output data. They produce relationships, rather than insight. The network maps the array of inputs to the array of outputs. NNs can find acceptable answers, although not necessarily the best answers, in a reasonable amount of time and for reasonable cost. NNs can spare tedious and expensive software design when compared to other methods of solving problems. NNs use information in large databases to extract a set of rules (Intel[2] 1990).

Rapid progress is being made in the field of NN technology. Scientists are trying to duplicate the way the brain makes inferences from incomplete or confusing information (Tazelaar 1989). NNs can classify, store, recall, and associate information. They can provide several recommendations for a particular situation in a few seconds. A machine that emulates the way a brain operates must work in a parallel fashion. There is no centralized site of computing power in the brain. NNs do not have a centralized computing site either. Rather, each individual computing site works independently.

NNs learn by association or examples, rather than by rules or mathematical formulas (Lawrence 1991). Neurocomputers are not programmed, they are trained. One of the most important objectives when developing an NN is to determine the

problem and the types of data that are available. Once this is accomplished, the next step is converting the information to a form that the NN can understand.

Several factors make NNs an attractive system choice for appropriate applications. These include using incomplete data to produce approximate results; managing large amounts of data quickly and efficiently; and using parallelism, speed, and trainability for fault tolerance (Obermeier 1989). However, appropriate applications for NNs still are being researched and clarified. The main application at this point in development is pattern recognition. NNs are best suited for problems in which overall accuracy may be less important than estimation and flexibility.

Two significant elements in NNs are learning and recall. Learning is the adjustment of weights (the strength of the relationship between elements in the NN that are computing outputs) to store information. Recall is the ability to retrieve the information stored in the weights.

### 2.4.1 Learning

NNs are data driven. While specific algorithms and problem definitions do not have to be formulated, the data must be accurate. If the data contain errors, the network will be trained according to those errors and will not be capable of producing accurate results. Although the success of an NN depends on accurate data, a clear physical understanding of the problem is not required to train the NN.

The most popular learning method is back propagation. To solve a problem with a back propagation network, the system requires many sample inputs with desired outputs. The network learns by adjusting weights. Back propagation compares actual prediction with a correct prediction and uses the error to change adaptive weights in a direction that is error reducing (Carpenter and Grossberg 1992). Known information is presented at the input, weighted values are assigned to the connections, and then the network is run repeatedly until the output is accurate. Running the network will readjust the connection weights. Data associations that are frequent will strengthen useful neural couplings, weaken negative couplings or connection weights, and disconnect unrelated ones. The network must be trained on preselected inputs and outputs and then it can be run on new information.

#### 2.4.1.1 Supervised Learning

In supervised learning, trial and error inputs are used to teach the network correct and incorrect responses. Learning is accomplished through direct comparison of the network output with known correct answers. When an error occurs, an error signal is sent back through the network and connection weights are changed to prevent the same error from recurring. This type of learning requires a priori knowledge of the results. Supervised learning uses class membership information. It detects errors because of pattern misclassifications. A trainer supervises what is output in reference to the input, and the network establishes a general representation of the regularities of the data.

#### 2.4.1.2 Unsupervised Learning

Unsupervised learning involves no trainer. Rather, the network organizes itself by establishing its own classification of inputs. This is accomplished by exposure to a number of inputs. Information used by the system is not categorized. Therefore, unsupervised learning cannot detect errors as well as supervised learning. Although this method is less computationally complex, it usually is less accurate.

#### 2.4.1.3 Self-Supervised Learning

Self-supervised learning takes place without an external monitor. There is a feedback device that detects errors and adjusts weights accordingly. This is a combination of supervised and unsupervised learning techniques.

### 2.4.2 Transfer Functions

The transfer function defines the value of the output signal (Lawrence 1991). The transfer function is a nonlinear function performed on the output of an NN. Different types of NNs use different nonlinearities, depending on the type of output desired. They are methods of limiting the output. There are five transfer functions regularly employed by NNs: linear, step, ramp, sigmoid, and gaussian. The differences among them have to do with the shape of the function. An NN system designer will choose the shape most appropriate for the particular application.

### 2.4.3 Construction

A biological neuron consists of axons (output), dendrites (input), and synapses (pathways for transmission of impulses from one neuron to another). In an artificial NN, wires are used for axons and dendrites, and resistors with weighted values are used for the synapses (Obermeier 1989). NNs consist of layers of Processing Elements (PEs), weighted connections, and threshold functions. Figure 2.4-1 shows the NN architecture.

The brain is an extremely complex system and artificial NNs are comparatively simple. In designing an NN, scientists are attempting to duplicate many of the desirable features found in the brain, such as fault tolerance, flexibility, and parallel processing (Hertz, Krogh, and Palmer 1991). Figure 2.4-2 shows the difference between artificial and biological neurons.

The PE, or neuron, is the part of the NN where computing is performed (Simpson 1991). NNs operate efficiently in parallel because each PE operates independently of the others. All PEs operate in parallel, as opposed to traditional digital processors, which perform sequential arithmetic and/or symbolic information processing (Krogmann 1991). A large number of interconnected neurons work toward a common goal. A PE depends on adjoining inputs from abutting connections to produce its outputs.

A network achieves stability, or converges, when the weight values associated with the PEs have finished changing. Networks normally converge, with training, to a stable solution.

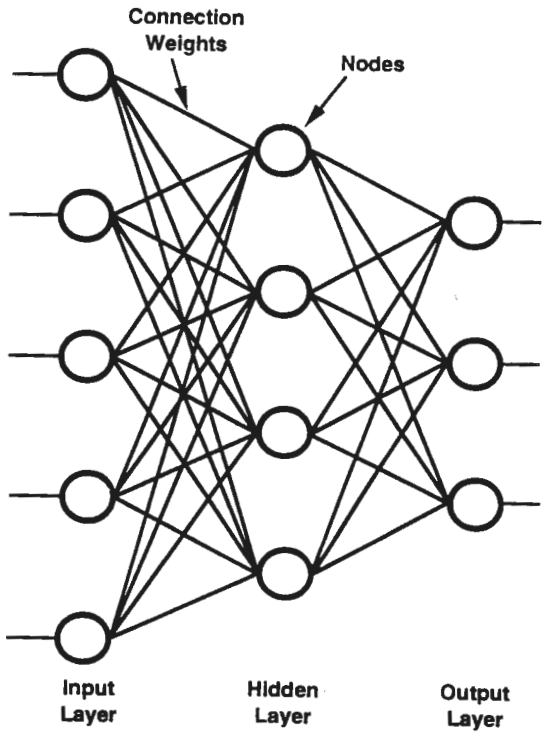



FIGURE 2.4-1. NEURAL NETWORK

2.4.3.1 Network Connections

The way in which neurons are connected determines the type of processing that will occur. A connection is a line of communication that goes from a sending neuron to a receiving neuron. Connections may go between the layers of neurons in an interlayer or intralayer fashion. The connections modulate the information flow between the PEs. The strength of the connections between neurons is a weight. A weight controls the strength of the incoming signal to the neuron. The degree of cell stimulation is determined by the frequency of the signals coming into the cell (Krogmann 1991). The collection of weights for the entire network is called a weight matrix. The knowledge of the network is distributed across the interconnections of neurons. Weights can be positive, negative, or zero. Feedback occurs when the output of one layer goes to the input of the previous or same layer.

The network interconnections can be changed by generating new connections, losing existing interconnections, and changing existing interconnection weights.

Detailed past information is stored in the connection weights of the network and is represented by a matrix.

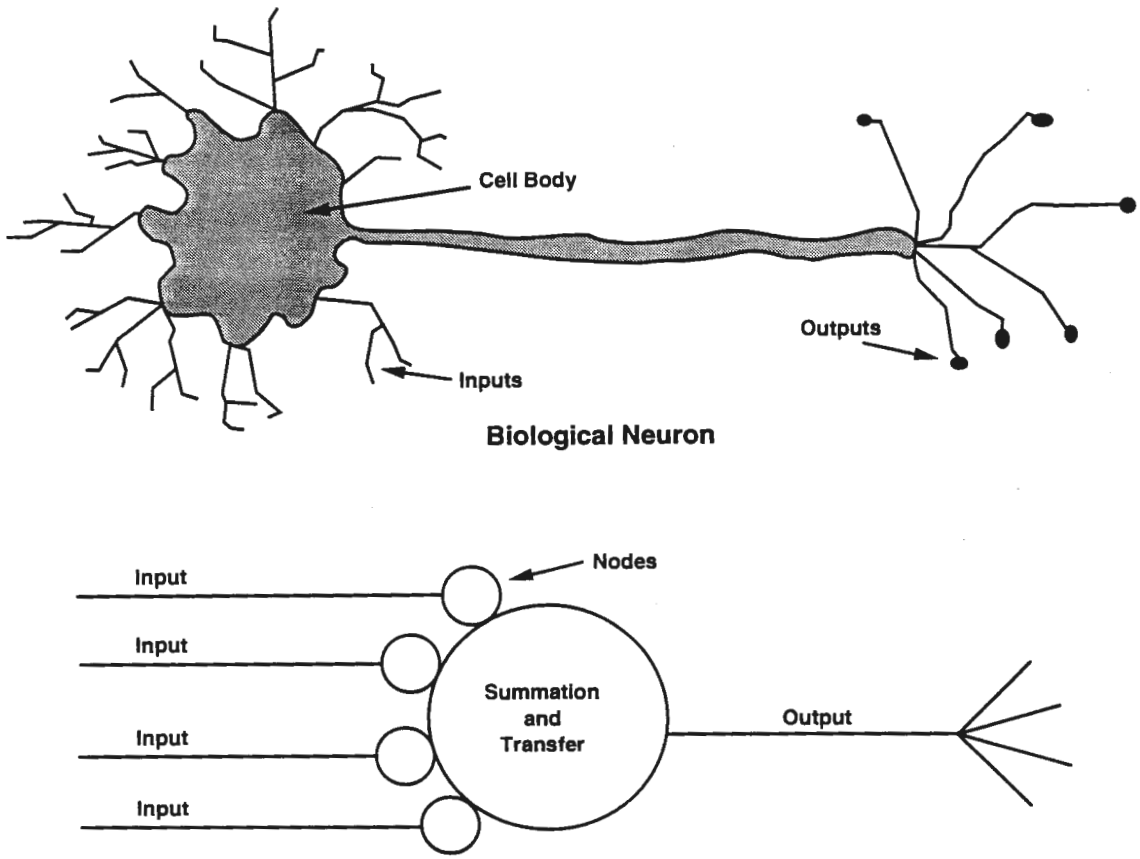



FIGURE 2.4-2. BIOLOGICAL AND ARTIFICIAL NEURONS

2.4.3.2 Neural Network Layers

A layer is a collection of neurons in an NN that is separate from other collections. Neurons are located in input, output, or hidden layers. Hidden layer neurons connect only to other neurons. Input and output layer neurons connect to the environment outside of the NN. Three layers are sufficient for any arbitrary non-linear mapping.

The input layer does not process data; it passes the inputs to the hidden layer. The hidden and output layers are the processing layers.

2.4.4 Designing a System

Following are some issues that must be resolved before an NN can be successfully implemented (Hertz, Krogh, and Palmer 1991):

- Architecture - How many units and layers are necessary? How should they be organized and what type of updating should be used?

- Training - How many examples should be used? How many times should the network cycle through the examples? Should learning be supervised or unsupervised, real-time or non real-time?

- Network type - How much can the network learn? How fast and robust is the network? How well can it generalize?

- Network hardware - What are the advantages and disadvantages of network hardware, and how does the hardware compare to software simulation.

### 2.4.5 Applications

NNs are used to solve problems that digital methods cannot handle efficiently. They can solve nonlinear problems because the neurons are highly interconnected and provide a close nonlinear relation between the input and output. Also, they can handle large amounts of data. Problems that involve complex mathematics and/or require quick, nonexact answers are suited for NNs. Also well-suited for NNs are problems that include many examples, but no rules or formulas. An advantage of NNs over rule-based programming is that if the process being analyzed changes, all that needs to be done is to collect new examples and retrain the NN. There is no need to determine new formulas and rewrite software.

Some of the operations that NNs perform include classification, pattern matching, pattern completion, and signal noise removal. The main application is pattern recognition. NNs can recognize patterns even when the data are noisy, ambiguous, distorted, or variable.

Following are characteristics of appropriate problems for NNs:

- The algorithm to solve the problem is unknown or expensive to develop.
- Rules to solve the problem are unknown or difficult to define.
- Large amounts of data are available for the problem at hand.

NNs, however, are not suitable for handling all types of problems. They are not suited for number processing, therefore, they should not replace data-based or knowledge-based processing. Traditional digital computers are better suited for problems that require accuracy and computational power. NNs are not an extension of sequential computers. Problems that are linear should use linear problem solving techniques.

### 2.4.6 Future of Neural Networks

Table 2.4-1 illustrates some of the needs and future applications of NNs.

## 2.5 Integrated AI Systems

### 2.5.1 Fuzzy Logic Integrated with Neural Networks

Integrating NNs with other technologies may have considerable potential. The difference between fuzzy and neural systems is in the way they represent knowledge. By adding fuzziness to an NN, the designer can describe boundaries more generally, describe input more vaguely, and maintain better control. Few

systems are capable of creating fuzzy decision rules and modifying them on the basis of past experiences. The synthesis of fuzzy logic and NNs has control and image processing applications.

### 2.5.2 Expert Systems Integrated with Neural Networks

ESs are suited for problems that are well-defined and specific to a certain domain, while NNs can deal with a broader and more general range of problems. ES implementation tends to be a long process, while NNs can provide answers quickly and with acceptable accuracy.

The goal of integrating these two methods is to extract features from NNs and ESs to create a system that can provide the advantages of both. The advantages include reasoning for collecting data and fast, general answers or more accurate, slower ones. Also, ESs can analyze and validate an NN's output.

TABLE 2.4-1. FUTURE EXPECTATIONS FOR NEURAL NETWORKS

| Needs |
|---|
| • More powerful serial digital computers<br>• Better understanding of brain operation<br>• Developments in hardware that will support NN speed and storage requirements |
| **Applications** |
| • NNs integrated with ESs<br>• Space exploration<br>• Nuclear power<br>• Military<br>• Applications where an NN's graceful degradation would allow operations to continue under circumstances in which conventional computers would cease operations<br>• Any application that would benefit from the unique pattern recognition and identification capabilities of NNs |

There are three steps involved in an ES/NN problem solving process: data collection, data evaluation, and conclusion analysis. Data collection and conclusion analysis are tasks best suited for the ES component. ESs are designed for user interaction, such as collecting inputs and producing outputs. Data

evaluation is a task well-suited for the NN part of the system, because it deals with imprecision and learning.

### 2.5.3 Expert Systems Integrated with Fuzzy Logic

A fuzzy ES is an ES that incorporates fuzzy sets and fuzzy logic into its reasoning process and knowledge representation. This allows a wide range of problems to be solved with the benefits of ES techniques and fuzzy theory. Fuzzy ESs can be user friendly, efficient, and compact.

In some areas, fuzzy methods can replace traditional methods altogether. In some applications, fuzzy techniques can augment other methods.

Inference procedures that can handle imprecise or vague knowledge are beginning to be recognized as useful problem solving methods. When developing the rule base in an ES, experts often express their knowledge in vague terms. An ES can handle imprecision in controlled amounts. When problems have uncertainty, fuzzy logic is the desired choice of methods. It has been employed successfully in several ESs. Fuzzy ESs use human-like behavior to handle concepts and reasoning.

Fuzzy logic is especially helpful for managing the rule base of an ES. It is nearly impossible to account for all possible combinations of antecedents and conclusions for any particular problem, especially large, complex problems. The logic needs to be traced, step by step, from all starting points to possible solutions. This task can be nearly impossible when dealing with even medium-sized knowledge bases. Fuzzy concepts, however, allow all combinations to be covered using fewer rules. Some action always will be taken, even if an exact match of the antecedent is achieved (Togai and Watanabe 1992).

## 3. ARTIFICIAL INTELLIGENCE AND EXPERT SYSTEM DEVELOPMENT

This section examines the software, hardware, and development environment related to the AI field. The software and hardware development tools and hardware devices discussed in this section are mentioned as a sampling of available tools, systems, and devices, and not intended to be a complete listing of what is available.

Complete solutions for AI-based problems are not available exclusively in silicon at this time. However, in the future this could change as technology continues to drive IC densities to greater levels of complexity and AI technology continues to mature. Whether it is a microprocessor with an AI-based native language, or an NN IC, the device is generally viewed as part of a larger system. Current applications for AI-based devices are proposed for automating and assisting human or machine performance. AI-based devices typically function as embedded processors.

Whether one is concerned only with hardware devices, or only with developing an AI-based software program, tools are required. Many issues should be examined when developers are considering a tool for building an AI-based system. There are many tools available, but the suitability of a tool should be examined carefully. Tools may be chosen by financial motivation, the developer's familiarity with the tool, or the tool's compatibility with the developer's hardware. In some cases, tools are developed and applications are then fabricated to test them (Waterman 1986).

### 3.1 Software for Artificial Intelligence Applications

There are many computer languages available for the program developer. Programming large applications in assembly language (a low level language using mnemonics to represent the processor's machine code) is inefficient, costly, and difficult to debug since there is a large body of code to review. Higher level languages and other development tools have been created to allow the programmer to develop and debug code rapidly.

Along with the creation of languages and tools came increases in the complexity of digital systems and the corresponding complexity of programs required to control them. Languages created specifically for the purpose of programming AI-based systems include LISt Processor (LISP) and PROgramming in LOGic (PROLOG). Other high level languages have been used as well.

Shaw[2] (1993) provides a list of AI language products and tools available, including LISP, Scheme, Rete, Logo, and PROLOG products. Some of the common AI-specific programming languages are presented briefly in the following sections.

### 3.1.1 List Processor

LISP is called the native language of AI (Charniak and McDermott 1985). Many different versions of LISP have been developed. COMMON LISP was developed in an attempt to focus these versions in a common direction and enhance the language portability.

Users can change the syntax of the LISP language to suit themselves. It is symbol, instead of number, oriented. Data structures are easily constructed in LISP. LISP is also procedure oriented. Nested subroutines are used to organize and control program execution.

LISP was designed for manipulating symbols, as opposed to numeric data. The ability to manipulate symbols has made LISP popular among the research community. Programming in LISP provides certain benefits:

- An interactive programming environment
- Incremental compilation
- Customization based on the application
- Automatic memory management features

### 3.1.2 Programming in Logic

PROLOG is a programming language based on mathematical logic. Finding a solution is a matter of deducing it from the facts. PROLOG has been successful in natural language processing applications and has been an important language base for the study of parallel programming techniques (Murphy 1993).

PROLOG software packages come in interpreted and compiled versions, and allow for the use of recursion. PROLOG is not suited for number crunching or interfacing to real-time data collection systems (Murphy 1993). Programming is often performed using two languages to handle the required functionality efficiently. For instance, C may be combined with PROLOG to be effective in a program with a mixture of numeric computation and logic.

### 3.1.3 Tools for the Expert System

ES shells are used to develop knowledge base systems. An ES shell provides the human interface for programming an ES. Shells limit development flexibility, but save development time by providing a development environment specifically suited to ES programming. Shells provide a method for representing and storing domain knowledge in the form of rules and provide an inference engine for making decisions based on that knowledge.

For ESs, programming languages and ES shells are used for development. Table 3.1-1 (Waterman 1986) summarizes some of the advantages and disadvantages of using an ES shell, as opposed to a programming language, to implement an ES.

### 3.1.4 Fuzzy Languages for Development

Language tools exist for generating assembly code for popular microprocessors based on a language translation from a predefined set of fuzzy instructions.

These tools operate on the fuzzy instruction set and convert each instruction into a sequence of assembly code mnemonics for the selected microprocessor. Tools also exist for fuzzy processors with a fuzzy-based native language. These tools define a particular fuzzy assembly language that requires no conversion process.

Instructions are converted from assembly code directly into executable machine code for the fuzzy controller.

TABLE 3.1-1. COMPARISON OF EXPERT SYSTEM AND PROGRAMMING LANGUAGE DEVELOPMENT METHODS

| | Expert Systems | Programming Languages |
|---|---|---|
| Advantages | Development is easier, faster, and, therefore, less costly | Greater flexibility |
| | More guidelines and mechanisms for representing and accessing the knowledge base | Product may more closely match the problem being solved |
| Disadvantages | Less flexibility | Developer will need to design the knowledge base and the inference engine |
| | May lack efficiency | Takes longer to develop application |

Some tools also generate C code from the translated fuzzy instruction set. The C code can be compiled and executed on a number of processors. These tools allow the developer to program using a fuzzy command set. A translation is then performed to create high-level C code. The C code can be compiled and executed on numerous processors and systems.

### 3.1.5 Other Languages and Tools

Procedures, standards, and development methodologies have been produced for the use of conventional programming languages in digital avionic systems, but do not exist for AI specific languages. Since this is so, some of the proposed avionic applications of AI-based systems have focused on the use of conventional software for the implementation. In the military community, the use of Ada has been mandated for future avionics and space applications.

A goal for large scale development of AI-based systems should be to automate as much of the process as possible by developing tools or by using tools that are available and suitable. Tools focused on automating the design process decrease the development cycle time. They also reduce the number of human errors.

#### 3.1.5.1 Boeing Advanced BlackBoard Ada Generation Environment

One of the tools being developed at Boeing is the Boeing Advanced BlackBoard Ada Generation Environment (BABBAGE). This tool is designed to be a complete work environment for the development of AI-based systems.

Leeper (1990) indicates that the need for such a tool set arose due to the following:

- Reluctance by AI programmers to deal with strongly "typed" languages, such as Ada
- Necessity to reduce syntactic and semantic coding errors
- Inflexibility of software interfaces to the AI engine
- Requirement to eliminate laxity in programming loosely typed languages

#### 3.1.5.2 CLIPS Expert System Shell

CLIPS is a knowledge base tool developed at NASA Johnson Space Flight Center. It was written in American National Standards Institute (ANSI) C. It is an ES shell that has a LISP-like syntax. Some of the features of CLIPS include (Daysh, et al. 1991):

- Use of variables for control of rules and facts
- Capability to perform LISP-like calculations within rules
- Ability to prioritize the rule firing sequence
- Provision of commands to aid in debugging

#### 3.1.5.3 FORTRAN Library for Expert Systems

The FORTRAN Library for EXpert systems (FLEX) was developed by the Royal Aerospace Establishment (RAE). It consists of a library of FORTRAN 77 subroutines that allows FORTRAN programs to interface with ESs. In addition to the subroutines, FLEX supports a separate knowledge base and provides explanation facilities (Daysh, et al. 1991). Butler and Corbin (1988) give further details on the features and operation of FLEX.

### 3.2 Artificial Intelligence and Expert Systems Hardware

Development of AI applications based on LISP began on an IBM 709. A high-performance application, MacLisp, appeared on the PDP-10. Other special purpose LISP machines began to appear, such as the Xerox PARC and Symbolics systems. Now, systems implementing AI-based applications are being developed using generic systems with compilers for the particular task at hand. Development environments exist for mainframes, workstations, and PCs.

Target environments for implementations vary widely. For instance, one AI implementation was performed using a MIL-STD-1750A processor, 64k memory, and the Ada programming language.

Recently, however, great strides have been made in the development of silicon-based devices that implement specific AI technologies. Most notable are high-speed and dense NNs and very fast fuzzy logic processors.

### 3.2.1 Neural Processors

NNs are implemented in both software and hardware. A software-implemented NN may be a standalone or an add-on software package which can run on a number of computer platforms. Inputs to the NN, as well as the weights, are stored in computer memory. The software performs processing based on a fixed or user-selectable algorithm. The output state is then determined based on the inputs, NN weights, and calculation method. Typically, users adjust the weights, choose the calculation method, and train the network. Software-based NNs may derive inputs from various external or internal sources, such as spreadsheets, databases, and custom input files. Where higher levels of performance are necessary, NNs are implemented using hardware accelerators or NN ICs.

Neural processors have been the focus of significant research recently. The following sections examine some of the silicon-based AI hardware that is either currently available or in some phase of research and development.

#### 3.2.1.1 80170NX Electrically Trainable Analog Neural Network

The 80170NX Electrically Trainable Analog Neural Network (ETANN) is an Electrically Erasable Programmable Read Only Memory (EEPROM)-based IC available in a 5-volt pin grid array (PGA) package with 208 pins. It is both Transistor-Transistor Logic (TTL) and Complimentary Metal-Oxide Semiconductor (CMOS) compatible at the interface connections (Intel[1] 1990).

The ETANN is internally configured with 64 neurons and 10,240 synapses. Inputs to the ETANN are both internal (feedback) and external. The device feedback inputs can also be used as device inputs, providing up to 128 total inputs. Performance claimed for this device is greater than $2 \times 10^9$ connections per second.

The use of EEPROM makes the device weights reprogrammable and nonvolatile. Individual weights and neurons are addressed using the device's address lines, and the weights are stored internally. A Learn Control pin allows the NN to operate in a learning mode, which causes the values of the synapse weights to change.

NNs are typically made for cascade operation. The outputs and inputs are voltage and current compatible so that the output from the first device connects directly to the input of the second device. Connections between ETANN devices can be made using either a bus interconnection scheme or direct pin-to-pin wiring (Intel[1] 1990).

#### 3.2.1.2 Nil000 Neural Network Integrated Circuit

One of the programs of the Advanced Research Projects Agency (ARPA [formerly DARPA]) is the Artificial Neural Network Technology program. One of the directions of this program is to fund development and applications of the Nil000.

Development is being performed by Intel and Nestor. Lockheed Missiles and Space Company is the first Beta test site for this device (Coleman 1993). The Nil000 design contains 1024 artificial neurons, implemented using 3.7 million transistors. Processing speed for the device is claimed to be 20 billion integer operations per second.

3.2.1.3 RN-200 Neurocomputer

This Neurocomputer IC was designed by Ricoh of Menlo Park, California. It is fabricated using CMOS technology as a gate array device with 200,000 gates, using .8-$\mu$m line widths. A total of 256 synapses are implemented on the IC. It consists of three layers: the input, processor, and output layers. The neuron update rate is claimed to be 1.5 billion per second when the device is running at a processing speed of 12 MHz (Normile 1992).

3.2.1.4 M1718 Digital Neural Network

The M1718 Digital Neural Network is produced by Hughes Semiconductor. This device utilizes static CMOS memory internally. It contains 1024 weights, which support 32 8-bit inputs connected to 32 internal nodes. The M1718 is TTL-compatible with an 8-bit bus interface. A throughput of 100,000 patterns per second is claimed by the manufacturer. The devices are gangable so that networks of varying sizes can be created.

3.2.1.5 Optical Neural Computing Integrated Circuit

A prototype optical neural computing IC has been produced (Mitsubishi Electric 1992). The design, shown in figure 3.2-1, uses eight long and narrow Light Emitting Diodes (LEDs) on top of an 8 x 8 array of photodetectors. Weighting is

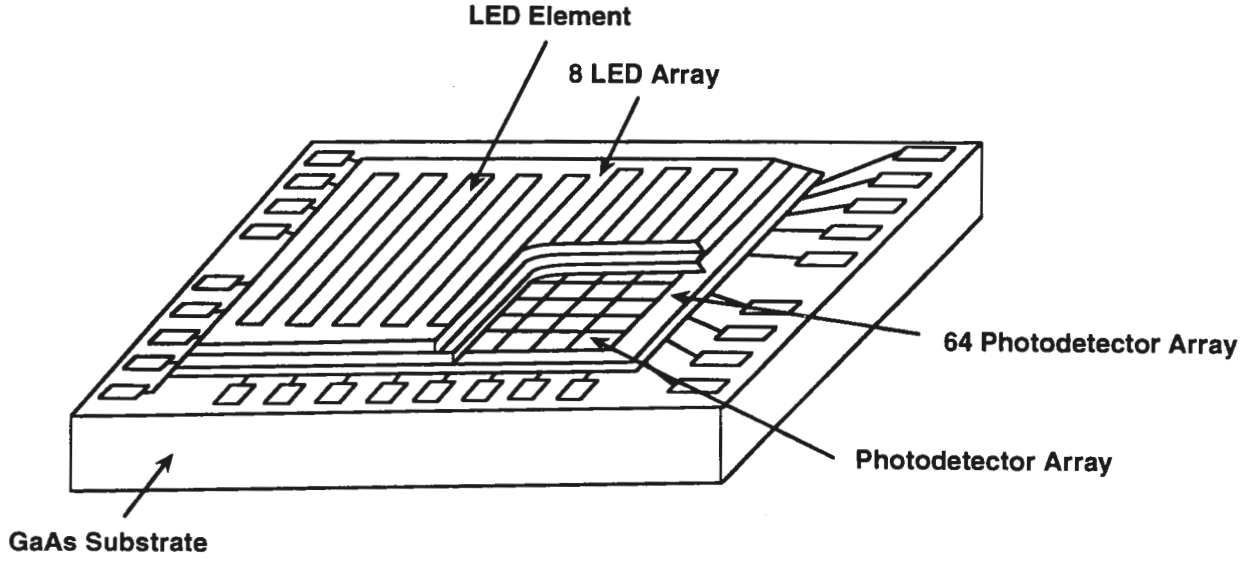



FIGURE 3.2-1. OPTICAL NEURAL COMPUTING INTEGRATED CIRCUIT
(Mitsubishi 1992)

performed by adjusting the sensitivity of the detectors. This IC is fabricated on a gallium arsenide (GaAs) substrate that measures 6 mm square. It learns at a rate of 600 million connection updates per second.

#### 3.2.1.6 Neural Accelerator Hardware

Neural accelerator hardware implements an NN on a printed circuit card that is hosted on a computer system such as a PC. Neural accelerators use high-speed Reduced Instruction Set Computer (RISC) processors, architecture, and other supporting hardware to attain fast connection rates, as compared to software-implemented or NN ICs.

### 3.2.2 Design Considerations for Fuzzy Logic

In cases where a fuzzy logic solution matches the intended application, designers can achieve significant advantages. Using fuzzy logic, system designers can realize lower development costs, superior features, and better end product performance. Products can be brought to market faster and more cost-effectively.

There are considerations to be made in the choice of an architecture for a fuzzy logic-based system. The proposed application can dictate the hardware required. Conventional Central Processing Units (CPUs) are sometimes used where execution speed is not a factor. Where speed is a factor, the choice of the CPU can have significant impact on system performance.

A CPU architecture containing a large register file is helpful. This is necessary if all terms are to exist within the CPU. A CPU containing a large register file greatly reduces the number of external memory accesses that are required during program execution.

Applications vary widely with respect to the number of rules required and the speed of the CPU which executes those rules. Some speed-critical applications may require only a few rules, but a very fast CPU to keep in time with the controlled system. A high-speed motor control application is one example where speed is essential, but the number of rules applied can be limited.

Automotive applications require high speed and can be complex. Antilock braking systems, climate control, and transmission control are examples of applications that may be suited to the use of fuzzy logic. High-speed applications will require the use of special fuzzy logic ICs. These ICs may be fast enough to control several tasks simultaneously. Fuzzy controllers execute rules 10 to 100 times faster than general purpose CPUs (Legg 1993).

#### 3.2.2.1 Fuzzy Coprocessors

Fuzzy Coprocessors and "engines" are also produced by a number of manufacturers. Fuzzy Coprocessors are intended to interface to various microprocessors so that fuzzy instructions can be performed rapidly, by a device tailored for them. They execute instructions specifically targeted for them out of the CPU instruction stream. Fuzzy engines are designed to be used as a peripheral device within a CPU, or as a basic building block for an Application Specific Integrated Circuit (ASIC). The ASIC elements can be combined to form various designs:

- ASICs with fuzzy capability and custom logic
- ASICs that combine fuzzy processing on the same IC as a conventional CPU
- ASICs that are designed to function as coprocessors

Some of the fuzzy engines and coprocessors that are under development or are in production are discussed in the following sections.

#### 3.2.2.1.1 FC110 Fuzzy Processor

The FC110 Fuzzy Processor was developed by Togai InfraLogic, Incorporated. The FC110 is called an "8-bit RISC processor optimized for fuzzy logic." It is designed for processing large rule bases and allows user-defined membership functions (see section 2.3) with arbitrary shapes (Legg 1993).

#### 3.2.2.1.2 SAE 81C99 Coprocessor

The SAE 81C99 coprocessor was developed by Siemens Corporation. Its features include an 8-bit interface to a microcontroller. Peak performance of this device is given at 7.9 million rules per second at a clock speed of 20 MHz. It has 256 inputs, 64 outputs, and handles up to 16,384 rules (Shear 1993).

#### 3.2.2.1.3 VY86C570 Fuzzy Computational Accelerator

The VY86C570 Fuzzy Computational Accelerator (FCA) was developed by Very Large Scale Integration (VLSI) Technology. It can execute in excess of 850,000 fuzzy rule evaluations per second when using two inputs, one output, and 40 rules. It is built around the Togai InfraLogic VY86C500 fuzzy processor core.

This device has a 12-bit data path, 256 x 12-bit data registers, and addresses up to 64k x 12 bits of external rule memory. It also can be used as a core design for custom ICs where fuzzy computational capabilities are desired. The VY86C570A has built-in self-test capability (Weiss 1993).

#### 3.2.2.1.4 NLX230 Fuzzy Microcontroller

The NLX230 fuzzy microcontroller is produced by American NeuraLogix. It is fabricated using 1.25-micron CMOS technology and can execute up to 30 million rules per second. The rule memory is 24 bits wide and can be programmed with up to 64 rules which are shared among the outputs (Ziemacki 1993).

## 3.3 Artificial Intelligence Development Systems and Tools

### 3.3.1 Fuzzy Development Systems

Systems developed using fuzzy logic concepts have used both conventional CPUs and special purpose fuzzy logic processors. For many applications, the general purpose CPU will suffice for the task at hand. For most current fuzzy logic applications, an 8-bit CPU is adequate. In more demanding applications, the use of a fuzzy logic processor is required (Legg 1993).

For developers of fuzzy systems, a number of tools are available:

- Assembly language code on a general purpose CPU
- Special fuzzy assembly code for fuzzy controllers
- C code generating tools
- Complete development environments

Speed requirements and system complexity dictate the hardware that is required. The use of a high level language, such as C, will slow down the execution speed when compared to a system developed using assembly code. For simpler applications, fuzzy controllers are not required and the system may be implemented by the use of look-up tables. Where rule execution speed is the primary requirement, high-speed fuzzy controllers are required. Between these two extremes, developers use general purpose CPUs with fuzzy logic software (Legg 1993).

The number of degrees of membership (see section 2.3) can be the determining factor when CPU resolution is considered. Complexity of the system (number of rules) is a determining factor when speed is considered.

As with any digital system development, tools should be a major consideration for the choice of IC used. Tools vary in capability, price, and user-friendliness. Tools are used not only in the development phase, but in the debug and testing phases as well. Some of the advantages tools can provide include (Legg 1993):

- Graphical interfaces to simplify editing and creation of membership functions

- Graphical display of controlled surface

- Tuning of a system while it is active

Tuning a fuzzy system empirically does not provide enough assurance of system stability. Research is addressing this issue.

Development environments are available for fuzzy logic-based systems and provide system engineers with necessary tools. Tools exist for each level of debugging, including data tracing, analysis, and simulation. If the inference unit does not perform as expected, these tools can assist the user in locating bugs in the source code.

Different target processors are supported by a number of development environments. Some development environments contain a full compliment of tools to take the design from inception to completion, and are combined into a single software product. These are referred to as Integrated Development Environments.

As growth in this field continues, standard fuzzy language code and portability among the various processors will become issues. Tools that provide users with an open architecture environment allow flexibility in their use. Tools can provide the user with data formats for membership functions, inference rules, inference units, and other fuzzy inference parameters.

#### 3.3.1.1 TILShell®

One development environment available to designers is a graphical, object-based environment called TILShell®. This shell allows the user to define the production rules and membership functions. Three compilers are provided, depending on the choice of target system. These compilers allow code to be produced as C source code, microprocessor specific code, or machine code for the FC110 Digital Fuzzy Processor (Conner 1993).

#### 3.3.1.2 Fuzzy Inference Development Environment

The Fuzzy Inference Development Environment (FIDE) is a tool based on Fuzzy Inference Language (FIL). FIL is a non-sequential language with English-like statements that incorporates inference methods and a variety of logic operators for different applications. FIDE includes a graphic editor for describing membership functions. The editor converts the graph into a segment of code. The FIDE compiler translates the source code into the standard data structure.

Debug tools, such as a trace facility, graphical analyzer, and simulation, are also provided. Once a processor has been selected for implementation, the corresponding real-time code generator can be used for that processor. If the unit is implemented in software, the run-time library, when linked into a C program, performs the fuzzy inference computation (Aptronix[1] 1992).

### 3.3.2 Fuzzy Logic for Digital Signal Processors

Systems can be designed using a fuzzy development package built around the Digital Signal Processor (DSP). This allows the features of the fuzzy system environment to be utilized in the DSP field for greater flexibility in particular applications. The development software combines fuzzy logic programming and allows DSP techniques to be programmed for the same hardware. Such a tool is available for the PC environment (Texas Instruments Literature 1994).

### 3.3.3 Neural Network Development Tools

NNs typically are not standalone devices. They generally are used as part of a larger system. A process passes data to the NN inputs. The NN then performs its operation and passes the output data back to the process. This is true for both software- and hardware-based NNs.

Designers need to keep in mind some potential drawbacks when using NNs.

There may not exist a satisfactory solution for a given problem if data are insufficient or if there is no learnable function. The problem must be definable. Training data need to be available and sufficient for the defined problem.

Proving consistency, completeness, and correctness for an NN may be difficult. The outputs of NNs depend on a high number of calculations. These calculations are based on input patterns and weights. The origin of the weights may be difficult to explain, since they are the result of a complex machine learning procedure.

NNs can be expensive to train. The training expense is due to the need to collect, study, and exercise the data. Additionally, the developer will need to manipulate parameters until the NN response is acceptable.

For NNs implemented using microprocessors, small increases in the number of nodes required cause large increases in execution time. As a rule of thumb, the execution time is roughly the square of the number of nodes. A microprocessor that works for a small number of nodes may not meet the required response time where the number of nodes is larger (Hammerstrom 1993).

Developers of NNs have the option of using a software-implemented NN or an NN hardware IC. Hardware devices are a relatively recent development and are generally expensive. However, if a software solution is too slow, a hardware NN may be required.

NN developers need to weigh the characteristics, requirements, and drawbacks of this technology carefully. Computational needs should be considered, including system throughput and the availability of training data. Sufficient data must be available to give a good representation of the behavior to be modeled.

If NNs are to be used in critical applications, other factors need to be considered. Hammerstrom (1993) sums up the design concerns effectively when he states:

> They are like statistics in their aggregate behavior, their usefulness, and their effectiveness in applications that lack other solutions and tolerate imprecision. They may be unsuitable when safety is critical or risks are to be avoided, unless they can be validated with all possible input values.

A number of manufacturers are marketing tools to assist in the design of NNs. NN development tools support software driven NNs and hardware NNs. Based on the design considerations, the developer chooses one of the two methods. Tools run on various systems, including DOS, Macintosh, Unix, VMS, and OS/2. Hammerstrom (1993) lists a variety of currently available development tools for both software- and hardware-based NN design.

### 3.3.4 Neural Network and Fuzzy Logic Combined System

An NN and fuzzy logic system have been combined in a product called NeuFuz4, which generates code for the COP8 $\mu$C (Shear 1993). The development package uses an NN to assist in building fuzzy logic applications, since generating membership functions and fuzzy rules can be difficult.

Input and output data and other application parameters are used to create the fuzzy logic rules and membership sets. NN weights are adjusted until the outputs of the network closely match the outputs of the training set, given the same set of inputs. The final weights represent behavior that can be translated into rules and membership functions (Legg 1993). The rules are then examined by a rule verifier and processed by an optimizer. When the developer is satisfied with the system simulation, machine code for the host processor can be generated.

The NeuFuz4 neural tool uses supervised learning. Also under development is a similar tool that uses unsupervised learning. It is anticipated that NN development tools used in training the fuzzy system will migrate into the fuzzy controller (Legg 1993).

# 4. ARTIFICIAL INTELLIGENCE AND EXPERT SYSTEM APPLICATIONS

Much research has been performed in the field of AI. Numerous studies have been funded or performed by government laboratories and agencies, as well as private industry and universities. This section examines some of the aviation-related studies, particularly systems proposed for airborne application. AI-based aircraft monitoring and diagnostic systems are also examined. This section will identify some of the design considerations for such applications.

## 4.1 Design Considerations for Avionic Applications

Designers of ESs must answer many questions during the course of development. These questions consider high level issues, such as system requirements, and black box issues, such as the type of system, inputs, processes, outputs, and interfaces. Many minute details must also be examined to implement the system.

### 4.1.1 Architecture Considerations

Avionic subsystems for advanced military aircraft will provide a host of complex functionality. The high number of sensors on modern aircraft requires that data management be given careful consideration. Assimilation of the wealth of data available presents a difficult task for any crew. Information needs to be compressed and sorted, presenting meaningful and necessary results to the crew. This function will be computationally intensive, requiring reliable and high-throughput processing. AI-based systems are targeted to help reduce the amount of information requiring the pilot's attention.

When designing an ES, one must choose an architecture consistent with the requirements. The influence of timing considerations on the architecture and hardware selected for the development task will need to be determined. The system timing requirements and architecture choice influence the implementation. Designers are faced with many decisions, including whether to choose a single CPU or a multi-processor configuration; conventional, RISC, or special purpose CPUs; message passing schemes; memory design; operating system; and numerous other related issues.

### 4.1.2 Timing Considerations

Timing is another example of a necessary, but important, detail when designing an ES for quick response or real-time operation. Following are some of the timing issues designers need to be aware of, or implement in their ESs, according to Ng (1988):

- Demonstrating that under any given scenario the system will provide data when desired, in a timely manner.

- Addressing response latency. Algorithms, such as those that perform LISP garbage collection, should be formulated with a time-sliced version to reduce response latency.

- Bounding problems so that the best reasonable solution under the circumstances can be identified.

- Eliminating unnecessary overhead from interactive development environments.

- Matching the processor architecture to the High-Level Language.

- Where higher speeds are required, using parallel algorithms, RISC processors, or other special hardware.

4.2 Flight Management Expert System

As aircraft systems become increasingly complex, the pilot's mental and physical workload will increase beyond realistic limits. Expert flight management systems can assist decision-making processes for the pilot. However, problems can arise in acquiring, structuring, and applying the knowledge base needed by an expert flight management system capable of operating in a dynamic domain.

The Emergency Procedures Expert System (EPES) is a decision aid system suited for flight management applications. The goal of the system is to provide pilot assistance during emergency situations while keeping the pilot in the loop. During normal operations, the EPES would remain unseen to the pilot, but would monitor the state of the aircraft constantly. If an emergency were to arise, the EPES would initiate steps to correct the emergency, inform the pilot of the action, and allow the pilot to override the action, if desired.

The EPES is comprised of an environmental simulation and an ES. The environmental simulation models the behavior of the aircraft and provides a display panel with realistic dials and gauges. The ES contains a special purpose inference engine and the knowledge base.

A rule/goal paradigm for managing emergencies forms the basis for the knowledge base. Data-driven, condition-action rules are triggered by changes in the state of the knowledge base. Corrective actions are defined by goals that become active when one or more emergencies arise.

The EPES is intended to represent the knowledge of an experienced pilot in handling multiple emergencies. A top-down approach to knowledge acquisition was used in the development process. The knowledge base was developed incrementally over time at increasing levels of detail. The knowledge base was developed as a collaboration between the knowledge engineer and pilot. The knowledge engineer gathered basic emergency information from the flight manual. However, to create a system that performed at an expert level, the knowledge of an experienced pilot was needed.

During knowledge base development for EPES, a number of lessons were learned:

- Although emergency procedures were described separately, often they were interrelated.

- Rules and goals were developed more effectively in parallel and in an iterative cycle.

- The structured text representational format allowed the knowledge engineer to isolate portions of the knowledge base that were incomplete. Specific questions could be generated to obtain the missing knowledge and to minimize interview time.

- Simulation exercises facilitated pilot feedback and aided in capturing knowledge that otherwise would have been difficult to obtain.

Although these processes were used in a flight management domain, this approach to knowledge base development could be applicable to similar domains with dynamic environments (Anderson, et al. 1985).

### 4.3 Navigation Systems

The Knowledge-Based AutoPilot (KBAP) is a NASA/RAE cooperative program established to develop and validate a real-time KBAP. The program investigates implementation and validation issues. The KBAP "provides a simple well defined yet real problem within which to explore, develop, and demonstrate real-time knowledge-based system concepts and validation and verification techniques for mission-critical systems" (Daysh, et al. 1991).

A prototype autopilot was developed using CLIPS, but was found to be too slow for real-time flight control applications. The RAE developed FLEX (a library of FORTRAN 77 subroutines that allow FORTRAN programs to interface with ESs) and has applied it to this application with much improved response time. One of the immediate goals for improving the response time is to use the rules produced under CLIPS and implement them under FLEX. The long-term goal is to apply the conventional software verification and validation (V&V) methodology used for flight critical control systems to the KBAP (Daysh, et al. 1991).

### 4.4 Decision Support Systems

Curran (1992) states that AI will likely be employed in the future to monitor and assist the pilot. Some of the proposed decision support systems are examined in this section.

#### 4.4.1 Cockpit Assistant System

The CASSY is a knowledge-based computer aid for flight-planning tasks. The purpose of this system is to support the pilot in complex planning and decision operations during situation assessment and flight replanning operations. This system has been implemented in flight simulators for single pilot and dual pilot testing. Research is being conducted by Dornier and the University of the German Armed Forces in Munich.

Many accidents in highly automated cockpits have been attributed to human error. The intent of CASSY is to use AI to reduce the number of accidents attributed to human error. CASSY functions like an expert copilot to recommend flight plan

revisions, to provide warning if the pilot deviates from ATC clearances, and to monitor aircraft systems.

CASSY monitors the aircraft systems to supply the air crew with warnings and advice. It performs situation assessment and planning activities using a knowledge base of ATC rules, airways, instrument approaches, and instrument procedures. CASSY uses voice recognition technology to communicate with the pilot and data link to communicate with ATC. Future plans include providing ATC clearances directly to the pilot and CASSY via data link.

When a pilot deviates from the approved flight plan, CASSY infers whether the deviation was intentional or accidental. The pilot is given verbal advice when a deviation is beyond the prescribed tolerance.

The results of simulator tests have been encouraging. It is the intent of CASSY to simplify the decision-making process, yet keep the pilot in the decision loop. During simulator trials, pilots using CASSY responded to ATC warnings within 5 seconds. Pilots without the system took from 15 to 95 seconds to evaluate the situation and respond.

System tests have been ongoing and future tests will be performed on aircraft. CASSY could be operational within 10 years (Nordwall 1992 and Prevot, Onken, and Dudek 1991).

### 4.4.2 Diverter System

The Diverter System is an AI application that provides pilots with information for making in-flight diversion decisions. In August 1990, a report describing the prototype system was prepared for NASA. The goal of the Diverter System is to use AI and algorithm-based decision aiding to replicate a pilot's information processing and application of logic principles during in-flight diversion planning. The Diverter System is designed to resemble closely the cognitive and information processing functions used by pilots during diversion functions.

The Diverter System is designed to be "invisible," although continuously active, until a situation arises that requires planning, display, and execution of a diversion. Background activity consists of constant database updates from onboard systems, as well as from ground-based systems through data link communications. Data are not displayed unless the system is activated.

Activation is accomplished either by the pilot or by the Diverter System. A menu option of the flight data Control Display Unit (CDU) allows the pilot to activate the Diverter System. Automatic activation occurs when the reason for diversion is due to a change in onboard aircraft system status or information from data link communications from ground-based sources. Activation annunciation takes place through the aircraft's fault/system status annunciation system.

A functional flow analysis was developed by querying pilots (domain experts) to obtain the methods and logic used during diversion planning. This information was augmented with information from sources such as Federal Aviation Regulations (FARs) and the Airmen's Information Manual (AIM).

The Diverter prototype is capable of combining available knowledge and using built-in knowledge to determine the best combination of runway, airfield, and route. The resultant route is then suggested to the pilot. In an actual system, runway, airfield, and aircraft attributes would be stored onboard, while information, such as weather and navigation aid status, would be obtained through data link or pilot input. The current Diverter System prototype did not attempt to define how this information transfer would take place. Also, the form in which information would be presented was not defined.

### 4.4.3 Flight-Plan Interactive Negotiation and Decision-Aiding System

The Flight-Plan Interactive Negotiation and Decision-Aiding System for Enroute Rerouting (FINDER) is being developed by Sextant Avionics and will be tested by Air France pilots when completed. FINDER is an interactive system that uses ES techniques, as well as conventional code written in C and C++.

FINDER's purpose is to suggest a limited number of satisfactory solutions for crew members facing a diversion situation (FINDER 1) or, more generally, an enroute replanning situation (FINDER 2). The system focuses on managing existing data and making decisions that aid the pilot. Following is information that FINDER will use to provide pilot assistance (Bittermann, et al. 1994):

- Fuel capacity
- Weather forecasts
- Ground facilities
- Regulations
- Passenger-based constraints

The problem is clearly stated here (Bittermann, et al. 1992):

> ...when pilots are confronted with diversion situations, they must find a solution which theoretically integrates all the parameters within a limited time and is based solely on their personal experience. Their attention is mainly concentrated on the critical considerations that may affect the aircraft's survival (fuel, weather).

The system will also, upon request, give reasons for a particular selection and allow modification by pilot/system dialog.

It is assumed that FINDER will have data link connections available with the airline owner, ATC, and possible destination airports. In addition, information is necessary from the Central Maintenance System and the Electronic Library System. FINDER will also investigate the impact of the evolving negotiation process between the crew and ATC.

A diversion is generated by FINDER using a two-phase decision process. Figure 4.4-1 shows how the process works.

The first phase, Context Analysis, examines the environment as completely as possible. This includes the condition of the aircraft and facts about the possible destination airports.

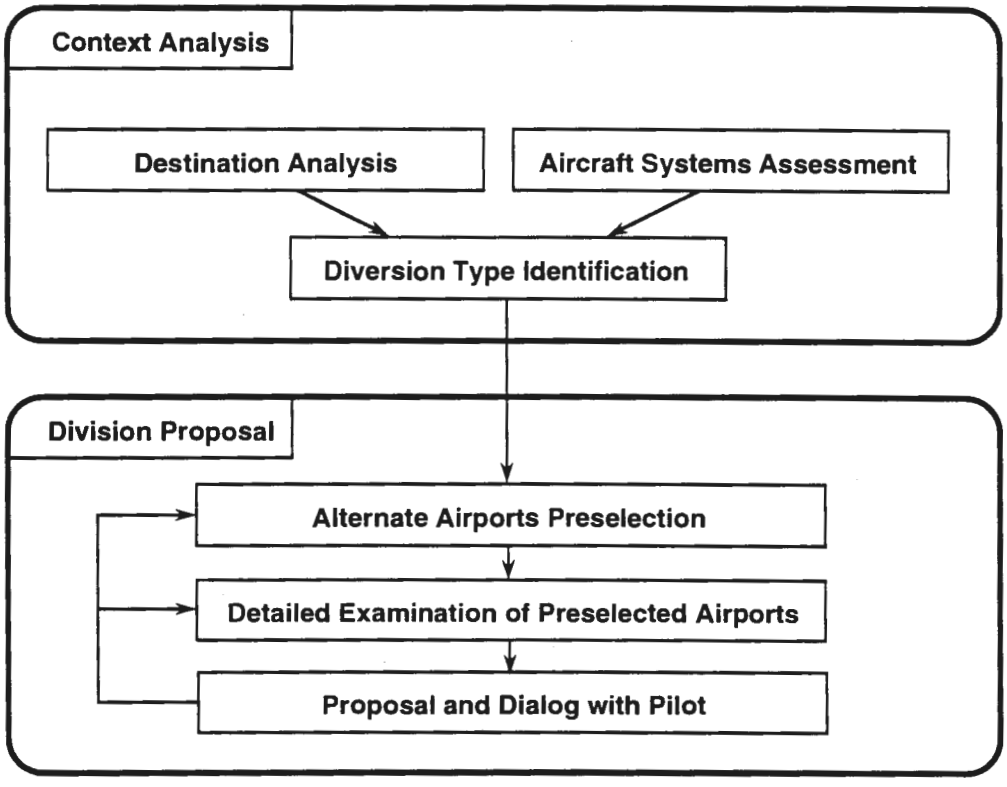



FIGURE 4.4-1. DIVERSION PROPOSAL GENERATION
(Bittermann, et al. 1992)

The second phase, Diversion Proposal, starts with the type of action determined by the Context Analysis phase and seeks to arrive at a satisfactory solution in light of all known facts. The first stage of the Diversion Proposal selects a number of possible destination airports, based on the destination analysis information for those airports. Using optimization heuristics, the optimum route is selected for the possible destination airports.

The last stage of this phase is the destination airport proposal and dialog with the pilot. A Graphical User Interface (GUI) is used for the pilot-system interface. A list of proposed airports is displayed by the GUI. If the pilot selects one of the proposed airports, the system will handle the necessary interface with ATC via the data link.

If the pilot does not select one of the airports, he or she can modify the parameters used in decision making. This will cause the system to regenerate the list of proposed airports in light of these new parameters. This new list will then be displayed for selection or further modification by the pilot.

The concept of diversion assistance during the FINDER 1 project led to the FINDER 2 project. FINDER 2 is devoted to developing an enhanced real-time demonstrator with extended replanning features (avoidance and optimization) and an enriched simulation environment (weather events, predictions and reports, regulated areas, and ATC and operation constraints).

The situation assessment module has been extended to overall environment and aircraft monitoring. It identifies potential problems, proposes corrective actions, and provides the crew with a comprehensive picture of the context evolution.

Each solution generated is maintained in association with the context and aircraft evolution (i.e., if an unaccessible airport becomes available while in diversion mode, the previously proposed solutions are automatically updated).

The pilot-system interface is hosted by an enhanced Electronic Library System Pilot Access Terminal. The logic of the dialog has been defined according to cognitive modelling procedures. The software architecture takes advantage of concurrency between the modules of the system's kernel (mainly situation assessment, replanning, and dialog). Human-like explanations are continually provided by the system on the pilot's request.

### 4.4.4 Pilot's Associate Program

A significant amount of aviation-related AI research in recent years has originated from programs funded by the U.S. Air Force (USAF) Aeronautical Systems Division. One such program, Pave Pace, will investigate and validate many areas of new technology, including AI algorithms, to assist in mission decision making and NN parallel-processing capability for adaptive learning systems (Curran 1992).

The USAF and the ARPA awarded a contract to a team from Lockheed and a team from McDonnell Douglas Corporation. This program, the Pilot's Associate (PA), has the goals of defining, designing, and demonstrating the application of machine intelligence to the assistance of advanced fighter aircraft pilots (Leavitt and Smith 1989). The Lockheed team was awarded a contract to develop an ES that tailored its response to the pilot's particular style of flying.

This particular system had technically difficult constraints. It had to react to inputs and communicate with the pilot rapidly. In addition, the knowledge base was designed to receive inputs from a variety of systems. The construction of the knowledge base relied on inputs from human factors experts, psychologists, engineers, and pilots.

One goal of the PA program is to have all of the software running on an avionic processor using a real-time procedural language, such as C or Ada (Rouse, Geddes, and Hammer 1990).

### 4.4.5 Rotorcraft Pilot's Associate Program

The Rotorcraft Pilot's Associate (RPA), a program sponsored by the U.S. Army, has similar goals to the PA program. The Army requested proposals from industry to develop a system that would alleviate the workload of rotorcraft combat pilots. The RPA proposes the use of a knowledge-based system to draw upon the knowledge of experts with prior experience in this discipline.

During the Vietnam era, a number of lessons were learned about combat avionics. It was found that when fighter aircraft and rotorcraft approached a combat zone,

pilots were shutting down some systems. The systems remaining "on" were those that worked automatically and reliably, yet kept the pilot fully aware of system status. These systems were advisory in nature and the pilot always was in command of the system.

The RPA program follows this design trend. The system monitors the rotorcraft and provides situation awareness. If an unknown target is encountered, the system suggests the correct response. The pilot can choose to accept the response or reject it. If the response is accepted, the system implements it automatically. The purpose of this system is to reduce cockpit workload and free the pilot to concentrate on tasks necessary for combat survival. Figure 4.4-2 identifies the basic functions included in the RPA.

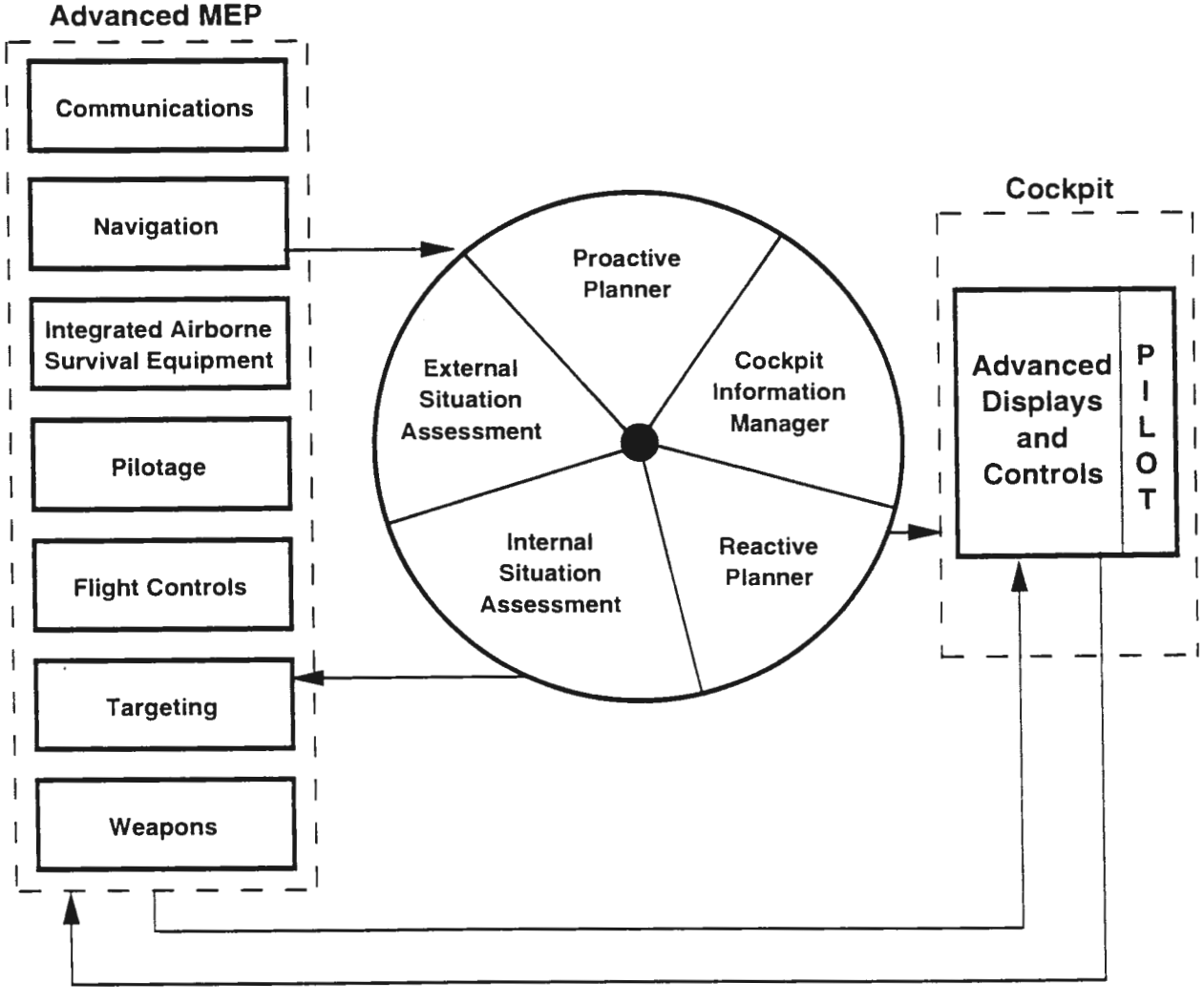



FIGURE 4.4-2. ROTORCRAFT PILOT'S ASSOCIATE DEMONSTRATOR CONCEPT (Harvey 1993)

The knowledge base will require detailed study and definition of the decisions that a pilot must make in combat aviation. In addition, difficult issues, such

as deciding which functions to automate and which are best left to the pilot, will need to be faced. The growth of AI technology makes the RPA project viable for future combat rotorcraft (Harvey 1993).

### 4.5 Intelligent Monitoring and Diagnostic Systems

As aircraft systems become more complex, difficulties have increased for system maintenance tasks. False alarms consume as much as 50 percent of maintenance resources, while troubleshooting actions take as much as 50 percent of total labor hours spent for repair. Accurate and efficient diagnostic systems are needed.

Developing rule bases for complex system diagnostics is difficult. Few experts possess ample understanding of new systems to provide adequate diagnostic rules. If multiple experts are used, it is difficult to gain consensus about the rules. These problems may be addressed by using rules based on causal relationships between failures and test information and by developing rules from examples of actual or simulated failures. However, providing sufficient examples to produce adequate fault trees may be impractical.

ESs are being developed for a number of aspects of avionics diagnostic testing. ESs monitor the results of Built-In Test (BIT) applications to provide better localization capability. Performance monitoring and fault locating software determine the cause of anomalies as they occur. In the future, software will be integrated into onboard maintenance systems to allow online diagnosis, reconfiguration, and repair.

Another application of ESs is production of Interactive Electronic Technical Manuals (IETMs) and Portable Maintenance Aids (PMAs). IETMs and PMAs combine online technical information with maintenance facilities.

ESs are beginning to be used in avionics testing as test executives for intelligent Automatic Test Equipment (ATE). The ATE systems' capabilities are as follows:

- Learn from experience
- Devise and execute test strategies dynamically
- Correct errors
- Explain and validate test choices

#### 4.5.1 Rotorcraft Transmission Health Monitoring System

As commercial and military aircraft age, there is increased emphasis on maintenance and safety issues. One important problem in this area is predicting impending faults in helicopter transmissions. In new aircraft, advanced diagnostics in which continuously monitored volumes of data must be reduced to provide meaningful information to pilots or ground crews also are important.

The Office of Naval Research (ONR) has initiated an aircraft health monitoring program that is investigating the use of NNs in mechanical diagnostic systems. Reduced complexity, speed, and cost were cited as advantages of using NNs in diagnostic systems. Reduced complexity would increase the reliability of a

diagnostic system. NNs will allow real-time diagnostics to be performed onboard the aircraft. These diagnostics will fuse data from the power train, drive train, structure, rotor system, and oil systems. As ICs to support this technology become mass produced, the cost of this technology will be increasingly affordable.

In June 1992, ONR contractors began work on an NN application to detect helicopter gearbox faults. Phase 1 has concentrated on detection and feature extraction. Subsequent phases of the ONR program will focus on classification, network design, and potential migration to a commercial product.

Preliminary ONR reports indicate a need for more and higher quality data with both fault and normal conditions. In the future, implementation of a data fusion approach for health monitoring will involve ESs, blackboards, NNs, and other AI technologies. Research still is needed in areas such as smart sensors, improved system-state recording, advanced algorithms, and faster data sampling and processing. Programs such as the ONR research in transmission diagnosis can enhance safety by using AI to improve health and usage monitoring (Rock, Malkoff, and Stewart 1993).

### 4.5.2 Technical Expert Aircraft Maintenance System

The complexity of modern combat aircraft avionic systems makes diagnosis and repair of these systems a difficult process. Knowledge of the aircraft, as well as a broad base of repair experience, is needed. Faults that occur during initial testing of newly manufactured aircraft are especially difficult to isolate, since the assumption that a failed system once worked correctly cannot be made. Typically, after each test flight, a team of mechanics fixes any problems that were discovered. The team of mechanics is supported by a staff of engineers who are specialists for each specific system.

The Technical Expert Aircraft Maintenance System (TEAMS), developed by McDonnell Aircraft Company, is an interactive ES that supports problem diagnosis in new aircraft. Using an ES in fault diagnosis supports the mechanic's diagnostic decision-making process and reduces the need for engineering support.

Development of ES technology applied to maintenance systems occurred in several stages. The first stage involved an evaluation of the technology and its expected impact on the aircraft delivery process. An ES was developed for the Communication, Navigation, and Identification (CNI) suite of systems. During a 3-month study period designed to assess the impact of the ES, it was determined that the CNI ES could result in a significant reduction in diagnostic time. The study also showed a reduction in the number of test flights required to sell an aircraft and a reduction in the number of spares required to be kept in inventory.

During stage 2, additional ESs were developed based on the results of stage 1. Flight controls and radar were selected for stage 2 because they produce the greatest number of flight faults in the avionic systems and isolation of failures in these systems requires considerable engineering support. This ES was developed using CLIPS. Stage 3 involved expanding the ES concept to other programs.

During stage 2, additional ESs were developed based on the results of stage 1. Flight controls and radar were selected for stage 2 because they produce the greatest number of flight faults in the avionic systems and isolation of failures in these systems requires considerable engineering support. This ES was developed using CLIPS. Stage 3 involved expanding the ES concept to other programs.

A number of approaches were used to involve the end user in the project and to encourage user acceptance of TEAMS. A training course was developed as soon as the CNI system was complete. The course focused on basic operation of the system and using the system in a simulated environment to find flight faults. A comment feature was added to TEAMS. This feature allowed users to enter a remark at any time during their use of the ES. The comment feature provided a valuable source of feedback. When comments were made, the designers were contacted for clarification (Lischke and Meyer 1992).

# 5. CERTIFICATION CONSIDERATIONS FOR ARTIFICIAL INTELLIGENCE

Depending on the intended use of a software-based system, incorrect software can result in significant safety or economic impacts. When safety issues are a primary consideration, a software-based system may be required to undergo certification corresponding to its application and criticality. Critical applications of software require rigorous attention to methods for reducing the probability of failure and ensuring the safety of those potentially affected by the failure of such software. This is no less true when applied to AI-based software. A "bug" in the AI-based software in a research lab may have minimal effects. However, a bug in the ES used to control stock market buying and selling or in the ES used to diagnose and monitor nuclear power generation systems may have far-reaching consequences.

For airborne software, certification is the process of obtaining FAA approval for the design, manufacture, and/or sale of a part, subsystem, system, or aircraft by establishing that it complies with applicable government regulations. The purpose of certification is to demonstrate and record that the total aircraft is suitable and safe for use (Elwell, et al. 1991). Key to certification is proving the software to be reliable and high in quality by a thorough examination of the methods used in designing, coding, testing, and life cycle planning.

Much research has taken place related to verification, validation, and testing of AI-based ESs. This section presents some of the methods and considerations relating to this discipline for AI-based software. A background of V&V techniques for conventional software is provided. Additionally, this section highlights some of the concerns relating to the proposed use of AI technology in the cockpit.

## 5.1 Certification and Federal Aviation Regulations

When a major design effort is required to develop a system, the integrity of the aircraft into which it will be installed is in question. Thus, one of two type certification processes must be followed to receive a certificate. For totally new designs, or changes that are so extensive as to require a complete reinvestigation of the design, the developer must follow the process required to obtain a Type Certificate (TC) for the aircraft. For major changes (as defined in FAR Part 21, section 93) to a system previously approved under a TC, the developer can follow a simpler process to obtain a Supplemental Type Certificate (STC).

The Technical Standing Order (TSO) authorization method of approving components was developed to allow manufacturers to substitute equivalent components "off-the-shelf" without jeopardizing the existing TC. Since a TSO authorization request must be processed within 30 days and does not require integration testing, manufacturers use this method rather than Type Certification whenever possible.

In the days of simpler aircraft design, TSO authorizations were adequate. Now, however, digital avionic systems are more complex and require elaborate testing procedures. To improve the TSO approval process, Aircraft Certification Offices (ACOs) are becoming more involved in approving new digital systems. They are reviewing V&V plans for TSO packages and are working more closely with the manufacturers. The ACOs are suggesting that system integration test plans be required for substitutions in integrated digital systems.

As avionic systems become more complex, the manufacturer must ensure that the system will function as intended within its operating environment. Manufacturers' validation facilities will play a major role in establishing the requirements for integration testing, since the functions of digital avionic systems will be simulated there. The certification requirements will expand for such systems to reflect the FAA's concern that the systems safely perform their functions once they are installed in an aircraft.

The system to be certificated can be simple or complex. The FARs stipulate which process must be followed in each case. Although the manufacturer may refer to the FARs to decide which approval should be sought, often a Certification Engineer (CE) recommends which application the manufacturer should submit. The authority for determining whether a change constitutes a modification or a redesign, and whether a redesign is minor or major, rests with the ACO.

## 5.2 Certification Guidelines

In general, to show compliance with the FARs, a system or component is subjected to environmental tests, software tests, failure analyses, and other testing, as deemed necessary. The FAA relies on the manufacturer to conduct the testing. If the FAA approves new tests, the manufacturer must comply with them. These tests are identified in guidelines adopted by the FAA. Organizations such as Aeronautical Radio, Incorporated (ARINC), Engineering Society for Advancing Mobility Land Sea Air and Space (SAE), and Requirements and Technical Concepts for Aviation (RTCA) produce such guidelines. Some guidelines which may be applied to AI-based systems are identified below. Further details of the certification process can be found in DOT/FAA/CT-91/19, "Avionic Data Bus Integration Technology."

### 5.2.1 Hardware Guidelines

In the past, to certificate an airplane, inspectors and engineers had to understand avionics based on analog electronic systems driving mechanical, pneumatic, and/or hydraulic systems. One of the currently used guidelines for the certification of avionic hardware is RTCA/DO-160C, "Environmental Conditions and Test Procedures for Airborne Equipment." RTCA/DO-160C addresses the effects of magnetic fields, voltage spikes, and induced voltages on electronic components. Any airborne hardware is subject to its requirements. There are also a number of other issues relating to hardware that influence the safety of flight.

Another guideline is the SAE's Aerospace Recommended Practice (ARP) 1834. It defines Fault and Failure Analysis (F/FA) techniques for digital hardware. Since digital systems are fault prone, the FAA has decided that fault analysis should

be employed during the certification process. FAR Parts 23, 25, 27, and 29, section 1309, and Advisory Circular (AC) 25.1309-1A express the need for fault analysis and ARP1834 has provided a means for conducting such an analysis.

Today's digital systems are more complex than those for which the tests were designed. Digital hardware complexity has increased to the point that traditional design methods can no longer be used. Automated tools are required for design, layout, and testing of complex ICs and circuit boards. According to Keller (1992):

> Some electronics systems themselves have attained a complexity level such that traditional lab and flight testing methods cannot realistically be extensive enough to show compliance to existing requirements.

New guidelines, however, may be adopted in the future as a result of RTCA's Special Committee (SC) 180, which is formulating design assurance guidance for complex electronic hardware used in airborne systems (RTCA Paper No. 548-93/SC180-18 1994).

### 5.2.2 Software Guidelines

This same explosive growth that occurred with digital hardware technology has occurred in the software domain as well, to the extent that "the probability of a hazardous error in software cannot be quantified" (Keller 1992).

Keller identifies three major issues relating to certification of software-based systems:

- Method of compliance demonstration
- Determination of the rigor required to ensure software integrity
- Determination of the software contribution to systems hazards

A method of compliance demonstration commonly used by manufacturers to meet FAA requirements is found in "Software Considerations in Airborne Systems and Equipment Certification" (RTCA/DO-178B 1992). This document was prepared by SC 167 of RTCA and was adopted as an official guideline in FAA AC 20-115C. Since software is generally too complex to test for errors, guidance is given in this document on acceptable processes used for developing airborne software. The software developer is required to be in compliance with this guideline from the initial stages of development and must have the appropriate artifacts to demonstrate the amount of development rigor (Keller 1992).

Determination of the rigor required to ensure software integrity is done in concert with the participating ACO. RTCA/DO-178B defines five categories of failure conditions, ranging from "Catastrophic" to "No Effect." Associated with each of these categories is a corresponding level, ranging from Level A through Level E. Level A, the most critical level, is defined as:

> Software whose anomalous behavior, as shown by the system safety assessment process, would cause or contribute to a failure of system

function resulting in a catastrophic failure condition for the aircraft.

Software that is determined to have no effect on the crew workload or the operational capability of the aircraft is defined as Level E. Once the software level determination is made, software at Level E is not subject to any additional guidelines of RTCA/DO-178B. The remaining three levels, from Level D through Level B, define software having an increasingly greater effect on flight safety.

RTCA/DO-178B addresses guidelines for producing airborne software that will meet airworthiness requirements. It addresses software life cycle processes, the activities associated with those process objectives, and the evidence that shows that the objectives have been met. It is a document that addresses all the critical stages of software development.

### 5.2.3 Software Guidelines and Artificial Intelligence

#### 5.2.3.1 RTCA/DO-178B

Existing certification guidelines were not designed with AI or knowledge-based software in view. The unique characteristics associated with the development and testing of AI-based software are not covered in RTCA/DO-178B, although AI has been identified as a technology that may receive attention in a future revision (DeWalt 1992). Manufacturers wishing to implement systems incorporating AI are, therefore, faced with certain difficulties when considering the certification process.

As an example, RTCA/DO-178B requires that algorithms be checked for accuracy if they are classified at level A, B, or C. While ESs may contain algorithms, they also contain knowledge bases which should be subjected to the same processes as conventional software. Just as algorithms are checked for accuracy, knowledge base accuracy also needs to be verified. For software criticality levels A or B, this verification should be satisfied by independent verification. This may be difficult for ESs, since domain experts are scarce and their knowledge can be contradictory. In addition, eliciting, capturing, and representing knowledge from domain experts can be a difficult task. Not even the expert can know if all the relevant information has been embodied in the knowledge base. Acceptable methods for knowledge base verification need to be addressed.

ESs are similar to advanced development tools, since there are few intermediate levels when ESs are viewed from the requirements level through the executable code level. While the number of products generated during the development of ESs are few as compared with conventional software, the trend in conventional software is to design systems at higher levels and reduce the number of steps in the development process by using automated design tools. Regardless of the number of development steps, tool qualification issues also need to be addressed, whether for conventional or ES-based systems. These and other related topics should receive coverage in a future revision of RTCA/DO-178.

#### 5.2.3.2 DOD-STD-2167A

DOD-STD-2167A establishes uniform requirements for software development over a system's life cycle. ES developers can face problems when compliance with DOD-STD-2167A is required. The software development cycle described by DOD-STD-2167A is not descriptive of the development process for ESs. The development process for ESs is a much more iterative process than that of conventional software. Also, the five phases of development are not independent and clear-cut, but describe a knowledge acquisition process. To accommodate knowledge-based systems, a new development cycle needs to be defined and incorporated into the appropriate standards (Carlton 1987).

## 5.3 Artificial Intelligence Standards

Since AI is a relatively new technology, a lack of standardization exists. Standards relating to AI are only beginning to emerge. This generates difficulties, especially from the developer's perspective. Adherence to standards can have many benefits for developers, including interoperability through common interfaces, as well as product portability. Additionally, lower overall life-cycle costs can be realized through increased reliability and maintainability. Areas where standards could benefit the technology include knowledge representation, ES shells, and AI language standardization.

## 5.4 Conventional Software Verification and Validation

This section examines techniques, methods, and issues relating to the V&V of conventional software-based systems. A review of these issues is necessary to provide the background and framework for the application of these techniques to AI-based systems. Once the issues are understood for conventional software, they can be applied to other types of systems and tailored for specific characteristics of those systems.

Although this section deals with software issues, it is important to note that the processes described in this section can and should be applied to hardware, software, and systems. New techniques may be necessary for ensuring safety goals for today's highly integrated and complex systems. Each component, as well as the total system, needs to be subjected to scrutiny that yields the required assurance level. Today's highly integrated avionics, using silicon devices with millions of transistors in a single package, require the application of V&V techniques specifically designed for this level of complexity, just as ESs require unique V&V techniques.

### 5.4.1 Verification and Validation

#### 5.4.1.1 Verification

There is no exclusively correct set of software statements for implementing solutions in software-based systems, since there are virtually an infinite number of different ways to represent solutions to system problems. Since software solutions are used in systems where failure can result in loss of life and property, methods are needed to demonstrate that the particular software solution

chosen to solve a system problem is correct and safe. This is one of the functions of the verification process.

According to the glossary of the Digital Systems Validation Handbook–Volume II, verification is defined as:

> The act of reviewing, inspecting, testing, checking, auditing, or otherwise establishing and documenting whether or not items, processes, services, or documents conform to specified requirements.

5.4.1.1.1 Verification and Documentation

One of the important ingredients in the process of verification is documentation. According to Naser (1991), "Documentation creates a traceable and systematic approach to the development of a product." Traceability is important since it demonstrates the flow and development of concepts and design elements from the requirements documents through the final product.

When a system is well documented, it can be understood by knowledgeable persons other than the developers. Persons other than the developers are often used in the verification process to analyze the product from a different perspective. This activity is referred to as Independent Verification and Validation (IV&V).

Problems in the development process often appear as the result of the transition from one step to the next. Part of the verification process deals with determining that a complete and accurate translation has been performed by the developers.

A requirements document describes the system at the highest level. It goes through many iterations of refinement during development. It also requires maintenance throughout the product life cycle.

Traceability depends on the completeness and accuracy of the requirements document. The requirements document is viewed as a set of external performance goals. Testing explicitly confirms that these product or system performance goals can be met. Testing against the requirements is both desirable and necessary (Naser 1991).

5.4.1.2 Validation

The Digital Systems Validation Handbook describes validation as:

> Demonstration and authentication that a final product operates in all modes and performs consistently and successfully under all actual operational and environmental conditions founded upon conformance to the applicable specifications.

Validation is performed on a system to ensure that the requirements and the end product, operating in its intended environment, have a direct correspondence. Validation is a process that is composed of a number of activities. The purpose is to demonstrate that a piece of software is designed correctly and that it is the correct or right design. The functionality, performance, and device

interface are given in the requirements document. If the product does not meet these criteria, then the deficiency can be traced back to the cause through each of the step-wise refinements of the development process.

One may deduce at this point that errors found in the validation phase are the most costly to correct. Since development proceeds in a step-wise manner, the correctness of each step is essential to avoiding mistakes that will ripple down into the end product, where they are consequently discovered.

5.4.1.3 Aspects of Verification and Validation

5.4.1.3.1 Correctness

There are several different aspects of correctness that require examination. These include functional, safety, user-interface, resource consumption, and utility correctness. Consistency, completeness, and termination are also aspects of correctness, but are examined separately.

5.4.1.3.1.1 Functional Correctness

Functional correctness is concerned with how a particular piece of software behaves, based on various system inputs. This behavior is defined by the system documentation. In general, it is not possible to test all input combinations completely. Therefore, a number of techniques and rationales exist for generating tests that are the most beneficial for demonstrating functional correctness of the subject software. Test methods to demonstrate functional correctness check to see that the program generates the correct system output for any input. Testing of all cases is generally not possible; therefore, only certain sets of test cases are examined. The preferred set of tests include those that are most likely to occur in normal operation.

There are several methods used for test case selection, some of which follow:

- Cause-effect graphing - Consists of mapping the input and output paths, where the conditions are expressed in Boolean form. Intermediate steps are also shown, yielding an accurate representation of the level of abstraction of the software.

- Boundary testing - A technique used to test input boundaries defined in the software. Testing the response using inputs near the boundary provides significant data on the correctness of the software.

- Attribute-based test case selection - Requires that test cases are selected based on software characteristics. For instance, a specific test case may be designed for a complex part of the software, while a simpler part may be passed over. Criticality, as well as program size, are attributes used for test case selection.

- Random testing - A technique for test case selection. As the name suggests, the method of selection for this technique is entirely random. Using this method, no rationale is required for selection of test cases.

#### 5.4.1.3.1.2 Safety Correctness

Safety correctness deals with showing that software never reaches a state that poses a danger to man or machine. With the increasing complexity of systems, this attribute may be easier to demonstrate than functional correctness.

Safety correctness testing is concerned with picking tests that are designed to generate unsafe states in the software. Knowledge of the controlled device is necessary to identify unsafe software states. Various combinations or sequences of inputs are exercised while the response is examined for unsafe conditions.

#### 5.4.1.3.1.3 User-Interface Correctness

User-interface correctness is concerned with the end-user and human factors aspects of the software interface. Some of the human factors aspects that are considered include ambiguity, correctness, and "user-friendliness."

#### 5.4.1.3.1.4 Resource Consumption Correctness

Resource consumption correctness demonstrates that the software operates efficiently within its environment. It focuses on performance issues, such as general hardware architecture, disk utilization, communication channel bandwidth, and meeting specified timing constraints. This method is essentially a performance test to check whether or not a system operates efficiently for its intended application.

#### 5.4.1.3.1.5 Utility Correctness

Utility correctness focuses on the usefulness of the system. It considers whether or not it is the right system, it does the right things, it does what the customer wants, and it meets the system requirements.

### 5.4.1.3.2 Consistency

Consistency demonstrates that the software follows a predictable path. Two forms of consistency are examined: external and internal. External consistency demonstrates that when a particular event occurs, an associated routine is exercised. External consistency also considers the user interface. The look and feel should be consistent. Internal consistency involves showing that all parts of an integrated software module are consistent.

### 5.4.1.3.3 Completeness

Completeness involves demonstrating that nothing is missing from the program. Completeness seeks to demonstrate properties, such as the following:

- The software accepts and acts upon all data.
- The software generates all required outputs.
- The software performs all required actions.

There are no algorithms that can be exercised to demonstrate completeness. Sufficient completeness should be demonstrated to show that the requirements are satisfied.

5.4.1.3.4 Termination

Another aspect of correctness is the demonstration of termination. This characteristic ensures that a program will always end. Demonstration of termination is done by showing that all looping structures within the program terminate (French and Hamilton 1992).

5.4.2 Verification and Validation Testing Phases

Testing is beneficial for several reasons:

- Both major and minor errors can be identified.
- Errors introduced into the system can be reduced.
- Confidence in the product can be increased.

This section discusses classes of testing performed on software, including static and dynamic testing. Testing methods are also discussed.

5.4.2.1 Static Testing

While dynamic testing involves running the software in the target environment, static testing involves analyzing the software by methods other than execution of the code. Static testing is done either by software review or by using tools to analyze the software. A detailed examination of the equations contained in the software is performed, or other checks are made against the specification.

The static testing method is the most cost effective since it occurs early in the development phase. As software development proceeds from the requirements phase to the system test phase, the cost of fixing errors increases drastically. Errors identified and corrected in the requirements phase have the smallest impact on the overall development.

5.4.2.2 Dynamic Testing

Dynamic testing is testing of the software in its intended operational (target) environment. Dynamic testing has two parts: system testing and unit/integration testing. This testing method is the least cost effective since it is performed later in the development phase. Errors identified and corrected in this phase of testing have a larger impact on the overall development cost and schedule. Detailed verification at the requirements phase can greatly reduce the amount of rework that may otherwise be performed in later stages of the development process.

5.4.2.2.1 Integration Testing

Integration testing is referred to as a "white box" testing technique (French and Hamilton 1992). It is dynamic and focuses on Input/Output (I/O) response classes. Integration testing can deal with multiple levels of abstraction,

whereas system testing deals with only one. This technique is concerned with the correctness of the unit.

Integration testing depends on modularity of the software. Modularity implies that the design and development process can be simplified by breaking a large and complex task into smaller but more manageable pieces. Each piece can be handled by a different person or group, and can, after thorough testing, be integrated with the other pieces. Modularity is a benefit to verification in that when changes are made, only the module that is changed needs to be verified.

#### 5.4.2.2.2 System Test

The system test is the last test to be performed. It is the final step in the development process. Demonstrating correctness at this stage does not require knowledge of the internals of the Unit Under Test (UUT). It is simply treated as a "black box." This black box is tested to see that the behavior it exhibits is correct. The responses of the black box are based on the original requirements that were established with the customer. The contents of the box are, at this point, irrelevant and not the subject of any investigation. Independent verification is desirable, since knowledge of the internals are irrelevant during system test and may be a stumbling block to testers.

It is generally accepted that complete testing of software is not possible. This is due to the fact that, for any given system, there are too many I/O responses to be tested. Complete testing of a system is not done, and, therefore, it is unknown whether or not the software in the black box is completely correct.

In order to devise practical limits on the number of I/O responses to be checked during testing, the software in the black box can be divided into categories based on the classification of the I/O responses. These individual categories can then be treated as separate units and tested as though each were its own separate box.

#### 5.4.2.3 General Testing Methods

The three general methods of testing are prototyping, regression testing, and IV&V. Prototyping involves building a scaled version of the real system. This is then used to evaluate the operation of the system. Regression testing involves testing older software that has undergone modification. It ensures that the modified portions of software do not affect other portions that were not changed. IV&V is a testing method that calls for an organization not involved in a product's development to perform the V&V. This method seeks to avoid prejudice towards that particular product on the part of the developers.

### 5.5 Expert System Software Verification and Validation

V&V is an essential activity for any system where software criticality is an issue. V&V is as important for ESs as it is for any software system. The adaptation of existing V&V methodologies to ESs requires an appreciation of the risks inherent in the particular implementation just as it does for conventional software. The variation in ES applications and implementation covers a wide range just as it does for conventional software. The V&V testing should be

planned accordingly. In many cases, an ES will require less effort to validate than a conventional software application of comparable complexity. Methodologies for V&V of ESs is an area of ongoing research.

Conventional V&V techniques can and should be used for parts of an ES, such as the inference engine; particularly if the inference engine is implemented with conventional software. However, the knowledge base is substantially different from conventional software implementations requiring the application of new V&V techniques.

ES applications that are limited in scope, having fixed and easily verifiable knowledge bases, should require significantly less V&V effort than those dealing with pilot intent interpretation. Software modules of the PA need to be subjected to more intense V&V because of the scope of the PA, the information the PA has as input (such as pilot intent), and the capability of the PA to exert control over other aircraft systems. A diverter ES, on the other hand, is much more limited in scope, deals with trustworthy data at the input (such as the AIM and FARs), and offers advice without exerting control over other systems.

In general, ES applications proposed for commercial aircraft have limited complexity. However, this report presents V&V techniques that address simple, as well as complex ESs. One of the tasks of the certification specialist is to determine the required level of testing based on the proposed application of the ES.

### 5.5.1 Contrasting Expert Systems and Conventional Software

The types of problems that ESs are used to solve are highly complex and too difficult to solve by conventional computer methods.

For these problems, a solution exists in the mind of the expert. This solution is derived by application of the expert's domain knowledge. This knowledge is then translated by the knowledge engineer into terms that a computer can understand. The manner by which problems are represented and solved constitutes the fundamental differences between ESs and conventional software.

#### 5.5.1.1 The Software Development Cycle

While many techniques that are used for V&V of conventional software can be applied to ESs, there remain areas where new techniques are required. Some of the differences are examined in the following sections. In spite of the differences between ES and conventional software development, ESs demonstrate a recognizable life cycle. There are well defined steps within this cycle to which V&V techniques can be applied.

Specifications and design documents are as necessary for ES development as they are for conventional software. Performance goals and functionality are measured against these documents. To implement any verification technique, an objective for a project must exist and be well documented. Without an objective, there is little help for the developer and no hope for the verification effort. Traceability requirements start with concrete measurable objectives. From this point, further refinements are made until the requirements are well-defined.

As the system is developed, document maintenance is required. ES requirements often emerge throughout the development cycle, necessitating incremental build and test cycles. According to Naser (1991):

> Testing the prototype reveals deficiencies in performance, suggests holes in the knowledge base, and stimulates another round of knowledge-building, coding, and testing.

5.5.1.1.1 Development Complexity

The power that an ES shell environment offers the developer and the user is often employed in developing applications far more complex than would ever be contemplated using a conventional software approach. This is usually manifested by a large number (many hundreds) of rules in an attempt to model a complex process with extremely high fidelity.

If the application is so difficult that only a powerful approach can hope to succeed, the problem of V&V comes from the scope of the application, not from the approach. A program such as the PA has very difficult goals, regardless of the development technique. An ES approach to the PA reduces the complexity of the overall task.

As with the verification of any critical and complex software application, the V&V approach must include an assessment of the risks to correct performance and direct attention to minimizing exposure to these risks. A major risk that is avoided with an ES approach is the integrity of the logic applied to the rules in the application. The V&V effort should also focus on the question of the integrity and fidelity of the rules relative to the problem domain. This effort is tenable, even when there are a large number of rules. This is because the rules are in the form of English-like statements, and, more importantly, the logic taken by the application on test cases can be examined minutely.

The complexity of the internal interactions of a software system depends not so much on the implementation technique, but on how the developer has applied the technique. Programs written in C can be as cryptic as the developer chooses. ES internal interactions that are complex need to receive greater attention in verification activity so that all consequences are known. It is possible to design ESs that have a more structured appearance and, therefore, facilitate testing.

In some cases, the inference engine may be implemented using LISP. When this is the case, the developer may be required to implement a garbage collection scheme.

Table 5.5-1 summarizes some of the major differences between ESs and conventional software.

5.5.1.1.2 Determination of Correctness

It is difficult to determine the correctness of a system that generates solutions based on the heuristics of an expert. The domain expert is the only person qualified to comment on the correctness of the ES. Knowledge base verification will be a key focal point of certification activity for critical software.

TABLE 5.5-1. EXPERT SYSTEM AND CONVENTIONAL SOFTWARE DIFFERENCE SUMMARY

| Expert Systems | Conventional Software |
|---|---|
| • Context driven, nonprocedural | • Sequential execution, procedural |
| • High-level, abstract data representation | • User-generated data representation |
| • More complex internal interactions | • More decisions |
| • May require garbage collection | • More control structures |
| • Rule interaction | • Program dependent internal interaction |
| • Intelligent control behavior | • Complexity |
| • Iterative development | • Stepwise development |

5.5.1.1.3 Expert System Shells

As is the case with conventional software compilers, ES shells may contain misleading documentation or exhibit faulty operation. Schultz and Geissman[1] (1991) point out that these potential problems may ultimately lead to incorrect operation of the developed system if they go unnoticed. To develop reliable systems, the knowledge base, database, and user-generated code, as well as the developer's tools need to be validated.

Paradigms (such as chaining) implemented in the tool need to be identified. The tool then needs to be examined to see if the paradigms are carried out correctly. The importance of a validated ES shell cannot be over-emphasized, since the attributes and capabilities of a system are influenced and limited by the weakest link of that system. All parameters of the tools that affect the operation or timing of the end product must be known and specified.

The easiest kinds of ESs to validate are those developed with an established commercial ES shell. These environments feature built-in rule editing and data management, graphics interfacing, and accessibility by direct incorporation of user written C code. They also support justification of answers by displaying on request the chain of rules by which they have come to a conclusion. Since the language that these ESs execute is in the form of English-like rules, the process of verifying the correctness of the program logic is greatly simplified. Both the accessibility of the language and the justification feature make it possible to achieve a level of confidence that an ES application developed and supported by such an ES shell does what was intended and with high reliability.

#### 5.5.1.1.4 Inference Engine Verification and Validation

In the past, ES users created their own ESs using conventional software. In so doing, they also programmed the inference engine. While the availability of commercial ES shells has greatly reduced the number of custom ESs, when this is the case, and the application's reliability is critical, the V&V effort should also extend to the inference engine. This is all the more true if the code and the knowledge base for the inference engine are to be maintained by the customer. Since ES shells contain inference engines, using a commercially available shell for system development can reduce, if not eliminate, the need for conventional programming. Also, the time to develop a system can be reduced by using a commercially available shell.

At a fundamental level, an inference engine matches antecedents of some rules with the consequences of others in what is interpreted as a logical sequence of reasoning. The testing of an inference engine involves developing a suite of rules whose behavior is well understood and encoding them into the form required by the inference engine. The test would be designed to show that, at a high level of confidence, all the possible connections between rules are explored in arriving at the conclusion to tests. The fact that the same answers are received by the inference engine under study as are received by the trusted inference engine should give high confidence that the inference engine is trustworthy.

#### 5.5.1.1.5 Flow Control

Knowledge-based systems are more context driven than systems developed with conventional software. Testing is more time consuming when context driven software is used due to an increase in the number of branches that require inspection. For verification purposes, it is easier if flow is determined at design time, as with conventional software.

#### 5.5.1.1.6 Implementation Language

Conventional software is developed using procedural languages such as Ada, C, Pascal, or FORTRAN. Knowledge-based systems use a combination of both non-procedural and procedural languages.

#### 5.5.1.1.7 Deterministic Behavior

A knowledge-based system is deterministic, unless learning is designed into the system. Hammer (1991) indicates that it would be difficult to verify that a system incorporating learning behaves safely at all times. Deterministic behavior must be demonstrated for ESs and conventional software systems.

#### 5.5.1.1.8 Use of Dynamic Memory

Computer programs often allocate memory from a "heap" when memory resources are required during the execution of various software procedures. For verified software, there should exist a statically determined upper bound, so that a routine requiring memory will never lack this resource. This memory allocation should be determined during the design phase.

#### 5.5.1.1.9 Domain Knowledge Review

In order for knowledge to be utilized by a computer, it is transformed extensively. A detailed review for correctness should be performed by the domain expert before translation. After undergoing transformation for use in a computer system, the original knowledge is in concealed form and the transformation process and product should be carefully examined.

#### 5.5.1.1.10 Verification of Knowledge-Based Requirements

Requirements specify system behavior. Understanding intelligence well enough to specify its interactions in a requirement document is difficult. Intelligent behavior consists of adaptivity, not simply complexity. Therefore, verification of a knowledge-based system should involve the testing of intelligent control behavior.

#### 5.5.1.1.11 Program Structure

It has been suggested that another difference between ESs and conventional software is that ESs are unstructured while conventional software is structured. While certain High-Order Languages (HOLs), such as Pascal and C, are referred to as structured languages, the degree to which a program reflects this characteristic is dependent upon the programmer exercising it. Software whose native language is not structured can be implemented in a structured manner by the programmer. What matters for ESs, as for conventional software, are the techniques used to implement a software application.

A related issue is the modularity of the system. With conventional software, tracing system development from the requirements to the code demands the use of a hierarchical structure. A technique called structured design is used. When this method is followed, procedures are generated in a top-down incremental fashion. In the past, rule-based ESs have lacked the hierarchical structure of conventional software. Rules were intermixed with each other and with control rules since no standard techniques or development tools existed to assist in achieving the goal of modularity. Modern design tools, however, allow rules to be categorized into classes by the designer. The ability to express the high levels of abstraction made possible with ESs allows designers to structure the knowledge base, along the most natural divisions that exist, as part of the knowledge.

### 5.5.2 Specification and Development Differences

Compared with conventional software, the number of design stages in an ES development project will be fewer due to the high level of expressiveness. The language of the ES operates at a much higher level than that of conventional software and may appear more like a requirements document than code. This is a benefit for ESs, since the verification process is concerned with the details of fewer intermediate steps. The number of associated documents will also be fewer than for conventional software.

#### 5.5.2.1 Vaguely Specified Systems

V&V of ESs is difficult when there is uncertainty that the rules in the knowledge base properly represent the best approach to a problem. This situation arises when the problem domain is not fully understood. When a knowledge base is developed by interviewing an expert who is unwilling or unable to define the processes used in solving a problem, it is up to a knowledge engineer to translate that knowledge into rules. Even with a cooperative and articulate expert, the process of refining a rule base can falter when the effort to make explicit expertise reaches the intuition level. It is frequently at the intuitive level that the performance of an expert is distinguished from that of a rule-follower.

Challenges to the V&V effort result when the requirements for the system, and specifications to which the system was developed, lack detail. The performance of the knowledge base must then be tested empirically, with the proof involving the approval of experts in the problem domain. The problems of verification are not insurmountable if the inference engine is verified independently and there is a large number of tests, perhaps developed by domain experts, for which the answers are independently known.

### 5.5.3 Implementation Differences

Conventional software and ESs are similar in that both consist of software which executes on a digital computer. There are associated inputs and outputs for any software system. An AI-based system or ES may even use a conventional HOL. Parts of the ES are typically implemented using conventional software.

Normally, the ES user develops ES applications through the use of tools. A common tool is the ES shell. The ES shell gives developers the capability to create systems without using procedural languages. Additionally, the problems that are solved using ESs have different characteristics. To be effective, the V&V approach and activity should be tailored to address these differences.

Procedural programs use a sequence of executed statements which are followed easily. The next statement to be executed in a sequence is based on the value of a defined set of variables. With non-procedural languages, the execution flow is not determined as easily. This is due to the characteristics of the inference engine used to control execution. Data and control are partitioned in ESs. The inference engine determines what statement is executed next by taking into account the basic state of the program together with the data. Partitioning for the knowledge base and built-in traceability tools for the shell assist developers in generating systems where the execution flow can be traced and explained easily.

Tool performance limitations may become a factor in the development process for ESs, as also may be the case for conventional software. There are performance variables associated with tools that are not apparent to the developers until actual real-time operation of the program.

All software is developed iteratively. ESs, however, require greater amounts of iteration in the development process since development involves extracting

information from the mind of domain experts. Many of the design assumptions made during early stages of development will undergo changes during the ES development cycle. Therefore, an incremental approach to development is viewed as necessary.

It must be said, however, that current conventional software development allows a far greater degree of iteration than in the past. New techniques and tools, such as Object-Oriented programming, applied to the development of conventional software systems have allowed rapid and iterative development cycles which former methods could not support.

### 5.5.4 Verification and Validation Techniques for Expert Systems

There are a number of techniques used for V&V. No single technique will be efficient for all applications. Different types of errors and characteristics are tested with each technique. Two general approaches exist. One examines characteristics of the ES implementation. The other examines whether or not the solution fits, or solves, the problem. This approach is implementation independent. The former method is called a "clear box" technique, as opposed to the black box technique. Both techniques are essential.

V&V methods are sometimes classified into two groups: qualitative and quantitative. A qualitative method applies "subjective comparisons of performance" to the ES. A quantitative method uses statistical techniques for comparison of ES performance against test cases or experts.

O'Keefe, Balci, and Smith (1991) divide the quantitative validation method into two categories: one uses a confidence interval and the other uses a hypothesis test. The confidence interval is compared subjectively with an acceptable performance range, whereas the hypothesis test is a formal test that compares measurements against a predetermined acceptable performance range.

A conventional software subroutine that manipulates variables passed to it based upon a mathematical procedure, will produce a solution for which only a single value or set of values is the correct result. Hence, correctness is easily tested. A difficulty often arises with ESs because there may be more than one correct solution to a problem. What constitutes an acceptable or unacceptable solution from the system should be defined clearly before verification begins (Schultz and Geissman[1] 1991).

Table 5.5-3 summarizes some V&V techniques for ESs (Naser 1991).

#### 5.5.4.1 Testing the Implementation

Testing the implementation involves checking interactions to see if they are intended and correct. In systems where there is a large amount of interaction, this can be a difficult task. Several techniques for implementation testing are examined in this section. These techniques do not identify errors, but provide an indication that an error may exist. It is then up to the programmer or domain expert to determine if the error actually exists.

TABLE 5.5-3. EXPERT SYSTEM VERIFICATION AND VALIDATION TECHNIQUES

| |
|---|
| • Formalize the requirements in a specification document. |
| • Verify that the requirements specification captures the requirements, including those for system usability. |
| • Verify that the requirements specification is implemented in the system design. |
| • Verify that the design is maintainable. |
| • Verify adequacy of knowledge and accuracy of knowledge presentation. |
| • Verify that the requirements are met for all interfaces of the system. |
| • Verify rule completeness and consistency. |
| • Conduct comprehensive shakedown testing, exercising as much of the system as possible. |
| • Verify system usability. |
| • Conduct tests for special cases and boundary conditions. |
| • Verify that the system is useful and fulfills its design purpose. |

Testing ESs can be similar to testing conventional software. Testing conventional software can be performed by exercising decision branches, checking looping conditions, and testing boundary conditions. Applied to ESs, these techniques could be mirrored by testing every rule or sequence of rules, if the rule base is not too large, and exercising rules based on the rule conditions. Additionally, just as conventional software is tested for quality of implementation, ESs should be checked for problem indicators. These include obscure constructs, overlapping rules, incongruous rules, incomplete rule coverage, and infinite rule loops (Hammer 1991).

5.5.4.1.1 Testing Rule Consistency

Rule consistency checks for the absence of undesirable characteristics in ES implementations. These characteristics include the following:

- Dead-ends - A rule that has no effect on any other rules.
- Unreachable rules - Rules that are not affected by any other rule.
- Circular rules - Rules that have the potential to repeat endlessly.

- Conflicting rules - Rules that contradict each other and perform actions which are in opposition.

- Redundant rules - Rules that contain the same information and perform some of the same actions as other rules. Although redundant rules may not actually produce errors, they should be identified and checked.

5.5.4.1.2 Testing Data Consistency

Data consistency involves checking that data or facts are used as defined. An operation that writes a 16-bit value into a variable defined as only 8 bits would be detected by such a check.

5.5.4.1.3 Sensitivity Analysis

Sensitivity analysis is an examination of how one variable may influence other variables. This type of analysis is useful in classifying problem solutions (French and Hamilton 1992).

5.5.4.1.4 Structural Testing

Structural testing is performed to ensure that a set of tests is comprehensive in coverage. This type of testing can be adapted to ESs by defining coverage to include rules, frames, and frame demons (procedures activated by changing or accessing frame data). When changes are made to the knowledge base, the coverage again must be determined and new tests created and executed.

5.5.4.1.5 Specification-Directed Analysis

Specification-directed analysis is a method for demonstrating that a program is traceable to the specification from which it was produced. This is performed by first arriving at a method which identifies and records specifications. These identified specification items are then checked against the program for correspondence.

In addition, the knowledge represented needs to be adequate for the intended use of the system. This requirement helps define test cases so that test coverage will be complete. When incorrect responses are found in the system, it should not be difficult to trace the incorrect response to its origin.

5.5.4.2 Test Bias Elimination

Biases in testing should be avoided, where possible. One particular bias is the use of test cases that were used during the development process. Presumably, the system was designed in such a way that it responds favorably to these tests. For completeness, some of these tests should be used to ensure that they still work. However, testing should focus on areas in which development testing has not been performed.

As with conventional software, validation and testing should be performed by a team independent of the development process to avoid prejudice.

5.5.4.3 Test Coverage

For conventional software, the sequence of execution corresponds to the high level code. In an ES, the exact sequence of execution may be difficult to determine by inspection. This becomes an issue since it adds to the difficulty of determining test coverage for verification. However, the sequence of execution is known after execution, since ESs can justify their reasoning by showing the chain of logic used to reach a conclusion.

5.5.4.4 Expert System Black Box Techniques

Black box techniques examine whether or not the solution of a particular problem fits, or solves, that problem. They are independent of any implementation language or design.

5.5.4.4.1 Knowledge Acquisition Checking

The earliest that errors in knowledge are introduced into the development process is in the acquisition of knowledge from the expert. Errors at this stage may be kept to a minimum by testing the expert knowledge for completeness. This can be accomplished by checking for the following (French and Hamilton 1992):

- Tasks which are defined but have no associated goal
- Goals that are defined with no associated task
- Undefined terms
- Conditions for which no goal or task is defined

5.5.4.4.2 Knowledge Consistency

Consistency of the supplied knowledge is also the focus of V&V techniques. This involves checking for contradictory or redundant statements.

5.5.4.4.3 Knowledge Representation

Representing knowledge in a form easily understood by both the expert and the knowledge engineer is essential to the verification process. Some techniques that can be used to simplify the representation of knowledge include state tables, concept maps, and task time lines (French and Hamilton 1992).

5.5.4.5 Formal Verification of Expert Systems

Formal methods, a technique designed for verification of conventional software, can also be used in verification of ESs. This technique, described as an alternative technique in RTCA/DO-178B, can be used to satisfy airborne software certification requirements. According to Rushby and Whitehurst (1989), this technique "can, in principle, guarantee the absence of faults." Its application involves demonstrating consistency between two different descriptions of a program. This can be done using two different levels of program specifications, or by using the program itself along with the program specification.

Consistency can be demonstrated by the working out of a formal mathematical proof, when the two parts are treated as formal mathematical texts. One way this

technique can be applied is by using a "stacked" application of verification steps. Conventional software breaks down into a number of steps during the development process. It is possible to represent each specification level formally and to demonstrate that each successive level demonstrates consistency with the prior level.

The use of formal methods is a promising technique, but requires further research. The formal methods technique can be time consuming and difficult, and, therefore, expensive to apply. Since this is so, formal methods are typically reserved for critical applications, where reliability and safety issues are primary considerations. This technology would benefit from guidelines or standards for expressing knowledge that needs to be cast into the formal verification mold. In addition, the development and use of suitable tools could assist ES implementers in producing systems with greater reliability.

#### 5.5.4.6 Other Verification Techniques

Other black box methods used in V&V of ESs include expert review, minimum competency testing, and disaster testing. Expert review is necessary for systems requiring V&V. By definition, the expert already has an understanding of the problem to be solved and may be able to work out solutions to test cases. Having the implementation in a form easily understood by the expert is essential to this phase of the review.

A minimum competency testing of an ES involves testing the system in much the same way as humans are tested for proficiency. A number of different scenarios may be generated for testing the response of the ES.

Systems are often tested for scenarios that may rarely, if ever, occur. Typical operation of the system under normal conditions often is treated with less rigor than some of the remotely possible conditions. These remotely possible conditions, however, may have serious consequences when they do occur. Testing for conditions such as these is referred to as disaster testing. This type of testing is also necessary for testing completeness of the system.

##### 5.5.4.6.1 Automating Knowledge Base Verification

Some tools are designed to assist the developer in automating the knowledge acquisition process. Others can keep the knowledge engineer in charge of the knowledge transfer process. As ESs become more commonplace and take on critical tasks, the need for tools to assist in the validation process increases. Methods to automate the validation process can provide many benefits. Tools can shorten the verification process, eliminate bias, and catch problems that may otherwise be overlooked. Following are the types of problems that may be found using automated tools:

- Syntactic errors
- Unused rules, facts, and questions
- Incorrectly used legal values

- Redundant constructs

- Rules that use illegal values, incorrect external declarations or references, and multiple methods for obtaining expression values

Jafar and Bahill (1993) describe a tool called Validator. This tool provides an interactive environment that is used to identify certain types of errors that may exist in knowledge-based systems. Syntax errors, misspelled words, values out of range, and unused values and rules are some of the errors detected by Validator. It has been exercised on a number of ES knowledge bases produced by university students, in addition to demonstration system knowledge bases from commercial software houses.

### 5.5.5 Supportability Issues for Expert Systems

In a systems engineering environment, supportability issues are not considered after a product is fielded. They are considered at the requirements level and influence the specification, design, and testing.

Following are several techniques used to enhance supportability of ESs (Carlton 1987):

- Keeping the inference engine separate from the knowledge base - Allows the knowledge base to be maintained without inference engine changes, allowing ease of use and flexibility.

- Representing knowledge uniformly - The higher the number of structures used to represent knowledge, the greater will be the effort when updates are required. It could be difficult to maintain a single mechanism since the problem type may dictate the most efficient representation.

- Avoiding complexity in the inference engine - For cases where the inference engine is developed by the user, a well-designed and simplified engine facilitates the generation of explanations.

- Utilizing redundancy - When developing the knowledge base, the use of multiple knowledge sources is recommended so that confidence in system performance is increased.

Another supportability issue focuses on the development environment used. If an ES shell is used, development can be faster and more reliable. Additionally, maintenance can be performed on this type of system by nonprogrammers. If an HOL is used, the developer has greater flexibility with knowledge representation and inferencing schemes. Additionally, if an HOL is used, other considerations should be weighted, such as ease of use, expressiveness, and flexibility of the HOL, since these factors influence supportability.

### 5.5.6 Configuration Management Issues for Expert Systems

Configuration Management (CM) is a process or discipline that manages and controls the modification of hardware and software components of a system. It defines functional and physical component characteristics and records and reports

changes in the status of these components. It should be applied rigorously to large-scale software and hardware development efforts. It has classically been applied to procedural software.

Knowledge can be represented both procedurally and declaratively. Procedural solutions are derived after a sequence of instructions is executed. An inference engine is an example of procedural knowledge. Conventional programs are totally procedural, while knowledge bases may contain a mix of procedural and declarative knowledge. Declarative knowledge consists of facts asserted to be true. An inference scheme is also associated with these facts (Carlton 1987).

CM techniques applied to procedural software are understood and can be applied easily. Since the inference engine is procedural, CM for it is well defined. Changes to the knowledge base have no influence on the inference engine. According to Carlton (1987), conventional CM for the knowledge base presents a problem. He states:

> While conventional configuration management procedures can be applied to the knowledge base, this will put unnecessary constraints on the flexibility of the knowledge-based system by limiting the ability of the user to make changes to the knowledge base to suit local conditions.

Changes to the knowledge base may become necessary when the ES operating environment changes. Local conditions are considered when information contained in the knowledge base is tailored to suit each operating environment. In this case, rapid changes to the knowledge base are required and the conventional CM process will be burdensome.

A solution to this problem is suggested. If the knowledge-based system can track its own configuration, much of the CM burden can be alleviated. This can be accomplished by having the system track changes and note them in a configuration data file. This file can be transmitted to the CM facility periodically. This file is then subject to central control and analysis (Carlton 1987).

### 5.5.7 Reliability Requirements

Metrics can be valuable sources of indicators for both development cost and failure intensity. Two common metrics applied to conventional software include complexity and size. The assumption is that larger programs and more complex programs will contain more errors and, therefore, be less reliable. A major difficulty is knowing which metrics to use and how to interpret the results (Rushby 1988). Measuring reliability is further complicated due to the differences between knowledge-based and conventional software. Further information on software metrics can be found in DOT/FAA/CT-91/1, "Software Quality Metrics."

Since ES technology is still rather young, there is a lack of experimental data for these systems. It also may be risky to assume that the relationships observed for conventional software will apply in the same way to ESs. Concerning this, Rushby (1988) states:

Empirical data will need to be gathered in order to suggest and validate useful relationships among metrics for AI software. It is unlikely that simple counts of the number of statements or rules in an AI system's knowledge base will yield useful information regarding its competence.

### 5.5.8 Testability Issues for Expert Systems

#### 5.5.8.1 Testing Techniques

With conventional software, coverage techniques exist for testing. They include branch, path, and condition coverage. Critical paths can be examined and traced through the software. The same is not true for knowledge-based software, since no critical paths exist to trace.

There are a large number of possible combinations and permutations of knowledge. This necessitates a change of testing methods for knowledge bases. Additionally, the number of test cases available to test the operation and performance of the ES will influence the testing coverage and the user's confidence in the system.

#### 5.5.8.2 Validity of Input Data

For conventional software, input data are tested for validity by verifying that they are within acceptable bounds as defined for the system. Methods for checking the validity of largely symbolic data need to be examined and developed for knowledge-based systems.

#### 5.5.8.3 Testing Level for Changes to the Knowledge Base

A change made to the knowledge base of an ES used in a critical system requires testing. The level of testing required is unclear. Working with partitioned knowledge bases may mitigate some of the difficulties encountered when testing changes (Carlton 1987).

# 6. HUMAN FACTORS AND ARTIFICIAL INTELLIGENCE

The amount of data that the flight crew must absorb has increased due to the expanding complexity and number of avionic systems in the cockpit. Reducing pilot and information overload can be achieved using AI. As with any technology, the final system interface (i.e., man-machine interface) needs to be an integral part of the design and development cycle.

Human factors concerns cover all aspects of technology, from design through implementation. To field automated intelligent systems and pilot assistants that add to the safety of flight, human factors technology should be applied to designs from the beginning. This will ensure that potential problems are identified and corrected before they have significant impact on system development.

## 6.1 Expert System Development Stages

There are four critical stages of development for ESs and their human factors-related concerns, as identified by Wickens (1992):

- Knowledge Acquisition - For complex ESs, the process of extracting knowledge from experts, and implementing that knowledge in the form of a rule base, can be a considerable challenge. Information supplied by the expert is subject to bias for which "debiasing" techniques may need to be applied. Additionally, not all expert knowledge can be verbalized. Methods other than verbalization may need to be explored.

- Knowledge Representation - How knowledge is organized is a concern. Is it understandable and can it be retrieved easily by users? Also, consideration is necessary for tailoring the stored knowledge to the capability level of the user.

- Knowledge Utilization - The design of the computer interface must consider sound human factors design techniques. This includes displays and controls, as well as the vocabulary used to communicate with the user. Another factor that needs to be addressed is how the confidence level of the ES's offered advice (which is a variable) is communicated to the user.

- Knowledge Maintenance - As is typical with any computer system, its data occasionally requires revision. Knowledge bases of ESs also require maintenance, either with new information being added or older knowledge being modified. How this is done, and when it is done, are concerns that certification specialists need to address.

## 6.2 National Plan and Artificial Intelligence

A valuable resource for identifying human factors issues and related safety concerns is the FAA's National Plan for Aviation Human Factors. This Plan represents the first step of an intensive effort to coordinate human factors-related research in aviation among government agencies, such as the FAA, NASA, and DOD.

In the Plan, a number of issues are raised that focus on safety concerns for intelligent cockpit devices. One such concern is how pilots will react to probabilistic AI. The Plan states:

> Automated flight deck systems will soon incorporate artificial intelligence with the capability to give pilots information or advice about complex matters with an associated probability that it is correct. Before such probabilistic AI systems come into extensive use in advanced technology aircraft, we need to know more about how crews will react to them, how to design them for optimum use, and how to train crews to interpret and use the probabilistic information.

In addition, other concerns, such as those listed below, are identified and expressed in the Plan:

- Intelligent automated systems should not be able to move the aircraft out of the safe operating envelope.
- The test and validation methodology for intelligent automated systems is still in its infancy and requires research and definition.
- The reaction of the flight crew to the type of information presented by an AI system requires investigation.

## 6.3 Keeping the Human as a Factor

One of the human factors problems identified for automated systems is the user's overconfidence in the decision making process. According to Wickens (1993), one of the methods proposed to:

> deal with bias, one that is emerging in the commercial flight deck and is already used in the military flight deck... [is the use of expert system technology]. Expert systems can, at least according to some scientists, replace some of the pilot decision making necessary in the cockpit, or at least can recommend judgements to the pilot...

Automated decision aiding has been studied for its ability to reduce or eliminate bias from decision making. There is some concern that these aids may serve to worsen, and not improve, decision performance (Wiener and Nagel 1988). In many of the cases where poor performance was noted, the human was used to gather data and implement decisions of the computer. In some cases, the combination led to worse performance than either the machine or human acting alone.

A method of improving the overall performance and interaction is shown in figure 6.1-1. This method takes into account the capabilities and strengths of the human and the machine. All of the possible solutions are generated by the machine. A more active role for the human is maintained using this method, and, hence, the human remains in the loop in a key role; not merely as a physical actuator for control input devices.

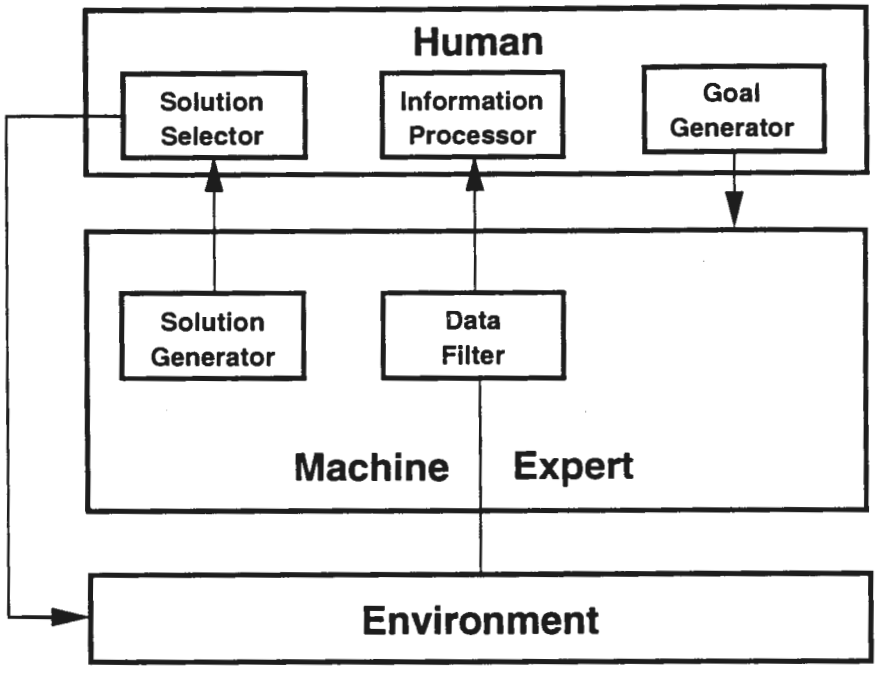



FIGURE 6.1-1. IMPROVED HUMAN PERFORMANCE GENERATOR
(Wiener and Nagel 1988)

Keeping the pilot in the loop is a concern, whether due to use of AI or due to use of any cockpit automation system. Reliance on yet another system may lead to further complacency and inattentiveness. Failure of an automated system creates increased crew workload, often at a time when this additional workload is unwelcome.

Any system, whether analog or digital, is prone to failure. A number of factors, including system complexity and unforeseen circumstances, prevent avionic systems from operating error free over their life cycle. The unintentional changing of a single bit of a digital system, whether by a software or hardware design error or externally induced, can produce significant undesirable consequences. While it may be possible for the system to monitor itself and flag most of these errors, not all problems are identified and announced to the system operator. Other systems may be used to monitor system health and reduce the probability of an undetected failure. Therefore, the role of the human as a system monitor should remain, regardless of the level of automation.

## 6.4 Automation and Other Human Factors Concerns

Other factors need to be examined, in addition to the problem of keeping the pilot in the loop. Human factors concerns over automation systems reflect some of the same concerns presented by AI systems. Some of these concerns are identified by Gabriel (1993) and consist of the following:

- Loss of Situation Awareness - Monitoring an automated system that is reliable is a boring task. The crew becomes distracted easily and diverts attention elsewhere. When this happens, awareness of the current situation is diminished.

- Loss of Proficiency - Competence in a skill requires practice. How to retain skills that are replaced or diminished by an automated system is a human factors issue.

- Reduced Job Satisfaction - Reduced job satisfaction can occur since the crew may no longer feel that their skills are important.

- Overconfidence in the System - Overconfidence in the system is a negative effect of automation. If the system malfunctions, the crew may not be prepared to handle the consequences.

- Intimidation by Automation and/or Complacency - A crew lacking experience or being overconfident may be reluctant to take control when the automated system does not perform well. In addition, an automated system may be designed in a way that does not communicate everything that the crew should know. This reluctance and uncertainty may reduce the ability of the crew to fulfill its responsibility.

- Increased Training Requirements - System complexity requires increased training. Part of the required training is to handle failure modes. Reprogramming can also be complex. Maintaining proficiency is necessary and, therefore, increased training is required.

- Inability of the Crew to Exercise Authority - This results when the crew is further removed from the decision making process by factors such as intimidation and the envelope protection features of the system.

- Design Induced Error - One of the goals of automation is to reduce human error. A concern is that while the frequency of errors is decreased, the consequence of the errors that do occur is much greater due to the control exerted by the automated system. Concerning this, Nigel (in Wiener and Nigel 1988) states:

> As aircraft become more highly automated, and the manual control skills of the pilot are displaced by aircraft systems (autopilots) which perform the general guidance and control functions associated with management of the flight path, discrete actions... seem to be assuming increasing importance in the causation of incidents and accidents.

# 7. CONCLUSION

AI technology can provide users with many benefits in a number of applications. However, there are limitations as to what can be performed with an AI-based system. It will never be able to replace genuine intelligence. AI deals with automating procedures that are already known. If little is known about a procedure, it is better to research the procedure than to attempt development based on vague specifications.

With currently available technology, ESs perform better as assistants or advisors than as primary decision makers. Humans will always have the advantage over ESs in that they have the power of human sensory pattern recognition and flexibility. AI systems will not be capable of replacing humans in most applications and will be used mainly to augment the capabilities of the user.

Another issue that needs to be addressed is the proper application for an AI-based system. AI technology is not efficient at solving all types of problems. Intense number manipulation and linear problems would be better performed using conventional methods. There are certain types of problems that naturally lend themselves to being solved using AI techniques.

Human experts often are in short supply. ESs are useful because the knowledge of experts can be incorporated into knowledge bases, allowing users who are not experts to benefit from the expert's knowledge. NN systems are helpful for applications such as pattern recognition. Fuzzy logic systems are useful for systems that normally require human intuition, are difficult to control with conventional techniques, or are difficult to model.

While many benefits can be gained from the use of AI-based technology, a number of issues need to be addressed before such systems are fielded in critical applications of airborne avionics. This section summarizes and highlights some key issues concerning certification of AI-based technology.

## 7.1 Learning Systems

One of the certification concerns mentioned in the National Plan for Aviation Human Factors is that airborne software not be allowed to remove the aircraft from the safe operating envelope. The capability of software to maintain an aircraft within the design envelope can be demonstrated for conventional, deterministic software. However, this would be difficult to demonstrate for AI-based systems with learning capability. Therefore, learning systems are unlikely to appear in civil aircraft (Naser 1991).

## 7.2 Software Level Determination

While ESs have been proposed and may be developed for military aircraft for the purpose of intervening and assuming control in life-threatening situations, it

is unlikely that this would happen in the near future for civil aircraft. It is more likely that ESs introduced in the near future for civil use will perform monitoring and pilot assistant functions. Certification specialists most likely will not require these systems to undergo the rigorous inspections and tests to which critical systems are subjected. One may argue that since the system actually controls nothing, it is not able to affect the performance and safety of the aircraft. On the other hand, receiving incorrect advice from a system that is normally relied upon by the captain and first officer can have dire consequences. Certification authorities will need to address these issues early in the development process, since determination of the software criticality level of a system affects inspection and documentation intensity and, therefore, implementation cost.

## 7.3 Enhancing Safety

An objection may be raised that we are just adding another layer of uncertainty on top of a system that is already plagued with unknown failures. Digital systems have unique failure modes compared to analog systems. Failures that are detected in aircraft operation at 40,000 feet are sometimes not reproducible on the test bench. Some digital systems problems magically disappear when the breaker is cycled off and on by the flight crew. Additionally, failures of digital systems are not always detected in a timely fashion. With this level of uncertainty already existing, would the addition of another level of uncertainty further degrade safe flying?

A similar objection can be raised when multiple access data buses or high-speed parallel backplanes are implemented for critical systems. Technologies formerly unknown in avionics are continually being designed and implemented and certification specialists address these issues as they arise. At issue is how AI and ES technology are perceived. If confidence in the technology is lacking, then its application to any application is viewed as a degradation. If it is perceived as a sufficiently mature technology that excels in certain applications (as it does), then its application in cockpit systems is seen as a benefit.

## 7.4 Hardware and Software Failures

There are numerous ways to express the same functionality using conventional software for system design. As tools automate more of the development process, design is performed using higher levels of representation.

One architectural design technique, mentioned in RTCA/DO-178B, for limiting the impact of software errors is the use of redundant, dissimilar software. The degree of dissimilarity and the advantages gained by utilizing this technique are minimized by the higher levels of representation available to designers. For ESs, this is also the case, since one of the attractions of this technology is that knowledge can be coded at a high level. Redundancy implies that the expert's rules would have to be different for accomplishing the same ends. Knowledge base partitioning and system monitoring techniques, along with extensive, detailed V&V, may be the solutions needed for ESs designed to perform critical functions.

The effect of hardware failures for AI-based systems also needs to be examined. What effect will hardware failures have on the conclusions reached by an ES? In the case of a pilot-aiding system, will failures be detectable only by the bad advice given by the system? Hardware error detection schemes will be critical to ensure system reliability, as they are with any automated system.

### 7.5 Ground-Based Avionics?

There is a related issue when considering the certification of AI-based airborne software. Some proposed pilot aid systems rely upon the availability of data link to receive company-related information required by the system. Data link is a two-way digital communication system. Using data link, one could transmit an aircraft's complete status, environmental data, and other necessary information to a ground-based facility on a communication channel. If that channel has sufficient bandwidth for a timely response, the exact AI-based program could be executed at the ground-based facility with the results transmitted back to the aircraft.

Such a system would not be subject to the certification requirements for airborne software, yet the only significant difference would be the physical location of the CPU that executes the AI-based software. While such a scenario may be impractical at the current time, certification specialists should consider the impacts of relying on this technology for airborne support, wherever the data are generated.

### 7.6 Verification and Validation Issues

One of the primary activities that requires addressing by developers and certification specialists concerns V&V methods for AI systems. Numerous V&V techniques that have been successfully applied to ESs have been identified in this report. Identifying a set of tests that can be demonstrated to satisfy safety requirements will be a challenge.

All parts of the ES, such as the database, knowledge base, inference engine, interfaces, and custom software need to be subjected to suitable inspection. Knowledge bases need to be complete. Missing rules will produce errors. For complex ESs, demonstrating that the knowledge base is complete may be difficult, since the expert may not even be aware of what is done to solve a problem. Also, it is impossible to guarantee that an ES contains all of the expert's knowledge.

The knowledge base also needs to be error free. Many of the techniques used with conventional software may be used to demonstrate that the system will not produce incorrect results. Also crucial to the knowledge base is maintaining data integrity. Maintaining an accurate, up-to-date, database is essential for product integrity throughout the product life cycle. Knowledge acquisition, representation, and data transformation techniques should be tested.

As with conventional software, key to the certification process for ESs will be the faithful application of a thorough V&V process, applied uniformly and starting with the product specification. Life cycle factors such as supportability, maintainability, and CM need to be considered and tailored for ESs as necessary.

## 7.7 Tool Qualification

Tools at all levels of the development process can assist in achieving cost, safety, and other goals. Tools enhance the development process by reducing development time and assisting in the elimination of human design error. Tools to automate knowledge base verification could provide substantial benefits when addressing knowledge base integrity issues. Standardization of knowledge base representation may be necessary to implement the automation process. Also, tool qualification issues require addressing since a new set of tools focusing on this field will require validation.

## 7.8 Design Guidance

ESs should be scrutinized using conventional V&V techniques where possible. RTCA/DO-178B needs to be examined to identify details of the processes that need to be modified to accommodate differences in AI-based and conventional development. Further effort is needed to develop relevant design guidance for AI-based systems.

## 7.9 Human Factors Issues for Expert Systems

Developing human-centered automation and designing advanced technology that will capitalize on the relative strengths of humans and machines are key to the success and usefulness of AI. Also, applying certification criteria to advanced technology systems, as part of the development process, will provide significant advantages for the testing phase.

The use of AI on the flightdeck brings up many issues relating to automation that need to be dealt with. Human factors specialists will need to address areas such as possible loss of situation awareness and proficiency, in addition to overconfidence in the system or intimidation by the system. The overall significance of incorporating probabilistic systems in the cockpit should be addressed as well.

Wickens (1993) states that too little is known about the way pilots make decisions and, therefore, ESs should not be entrusted with decision recommendations. While this is certainly a concern and has been the focus of much research for the PA and other studies, the many benefits of using AI-based technology beg to be investigated. At issue is not the complete understanding of the human thought process, but using a logical, consistent, unbiased decision tool (inference engine) to make conclusions based on available data. Simple systems, such as diverters and diagnostic aids based on ESs present logical approaches to solving complex problems in an accurate and timely fashion.

AI-based systems for commercial aircraft are being researched and tested. These systems can offer distinct advantages over conventional software systems. Ultimately, such systems can offer economic advantages and contribute to flight safety. AI-based pilot aiding systems have the potential to contribute meaningfully to aviation. A solution that matches the technology with the problem to be solved, using a systems engineering approach, will be a solid start for developers of AI-based avionics.

BIBLIOGRAPHY


Adeli, Hojjat, Knowledge Engineering Volume 1, Fundamentals, McGraw-Hill Publishing Company, New York, NY, 1990.

Anderson, Bruce A., et al., "Knowledge Engineering for a Flight Management Expert System," Proceedings of the IEEE 1985 National Aerospace and Electronics Conference, IEEE Service Center, Piscataway, NJ, 1985.

Anderson, James A., "Neural-Network Learning and Mark Twain's Cat," IEEE Communications, Volume 30, No. 9, September 1992.

Antonisse, James H. and Karl S. Keller, "Dynamic Evaluation of Imprecisely Specified Knowledge," IEEE/AIAA Digital Avionics Systems Conference Proceedings, IEEE Service Center, Piscataway, NJ, 1986.

Aptronix[1], Inc., NEWS11.TXT, San Jose, CA, July 1992.

Aptronix[2], Inc., What is Fide?, San Jose, CA, July 1992.

Araya, Agustin and Sanjay Mittal, "Compiling Design Plans from Descriptions of Artifacts and Problem Solving Heuristics," Proceedings of the 10th International Joint Conference on Artificial Intelligence, Milan, Italy, August 23-28, 1987.

Aviation Week & Space Technology, "Software Developed for Using Neural Network Techniques in Problem Solving," June 17, 1991.

Avionics Update, "Army Awards Contract for "Intelligent" Helicopter Associate," Avionics, September 1993.

Ballard, Dan and Dave Nielsen, "A Real-Time Knowledge Processing Executive for Army Rotorcraft Applications," IEEE/AIAA 11th Digital Avionics Systems Conference Proceedings, IEEE Service Center, Piscataway, NJ, 1992.

Ballard, Dan and Lisa Owsley, "Artificial Intelligence in the Helicopter Cockpit of the Future," IEEE/AIAA 10th Digital Avionics Systems Conference Proceedings, IEEE Service Center, Piscataway, NJ, 1991.

Barber, Richard and George Imiah, "Delivering the Goods with Lisp," Communications of the ACM, Volume 34, No. 9, September 1991.

Bartee, Thomas C., ed., Expert Systems and Artificial Intelligence: Applications and Management, Howard W. Sams and Company, Indianapolis, IN, 1988.

Becker, Lee, et al., "Translating Expert System Rules into Ada Code with Verification and Validation," NASA Contractor Report 187505, NASA Langley Research Center, Hampton, VA, June 1991.

Berardi, L., et al., "System Architectures," AGARD Conference Proceedings 417: The Design, Development and Testing of Complex Avionics Systems, Advisory Group for Aerospace Research and Development, Neuilly Sur Seine, France, December 1987.

Berning, Sandra, Douglas P. Glasson, and James A. Guffey, "Adaptive Tactical Navigation Concepts," Proceedings of the IEEE 1986 National Aerospace and Electronics Conference, IEEE Service Center, Piscataway, NJ, 1986.

Berning, Sandra, Douglas P. Glasson, and Gary A. Matchett, "Functionality and Architectures for an Adaptive Tactical Navigation System," Proceedings of the IEEE 1987 National Aerospace and Electronics Conference, IEEE Service Center, Piscataway, NJ, 1987.

Berning, Sandra, Douglas P. Glasson, and James A. Guffey, "Adaptive Tactical Navigation Program," AGARD Conference Proceedings 499: Machine Intelligence for Aerospace Electronic Systems, Advisory Group for Aerospace Research and Development, Neuilly Sur Seine, France, May 1991.

Berning, Sandra, Douglas P. Glasson, and Jean-Michael L. Pomarede, "Knowledge Engineering for the Adaptive Tactical Navigator," Proceedings of the IEEE 1988 National Aerospace and Electronics Conference, IEEE Service Center, Piscataway, NJ, 1988.

Bezdek, James C., Analysis of Fuzzy Information, CRC Press, Boca Raton, FL, 1987.

Bezdek, James C., "Computing with Uncertainty," IEEE Communications, Volume 30, No. 9, September 1992.

Bezdek, James C. and Pal K. Sankar, Fuzzy Models for Pattern Recognition, IEEE Computer Society Press, Los Alamitos, CA, 1991.

Biema, Michael, "Parallelism in Lisp," Proceedings of the 10th International Joint Conference on Artificial Intelligence, Milan, Italy, August 23-28, 1987.

Bittermann, Vincent, et al., "FINDER: Flight-Plan Interactive Negotiation and Decision Aiding System for Enroute Rerouting," 12th International Conference on Artificial Intelligence, Expert System, Natural Language, Avignon, France, June 1-6, 1992.

Boehm, Barry W., "A Spiral Model of Software Development and Enhancement," Uma G. Gupta ed., Validating and Verifying Knowledge-Based Systems, IEEE Computer Society Press, Los Alamitos, CA, 1991.

Bouchard, Philippe O., Keynote Address, AGARD Conference Proceedings 499: Machine Intelligence for Aerospace Electronic Systems, Advisory Group for

Aerospace Research and Development, Neuilly Sur Seine, France, September 1991.

Bowyer, M. R. and S. A. Cross, "Parallel Knowledge Based Systems Architectures for In-Flight Mission Management," AGARD Conference Proceeding 504: Air Vehicle Mission Control and Management, Advisory Group for Aerospace Research and Development, Neuilly Sur Seine, France, October 1991.

Boys, Randy and Katherine Palko, "Automation and Dynamic Allocation: Engineering Issues and Approaches," Proceedings of the IEEE 1988 National Aerospace and Electronics Conference, IEEE Service Center, Piscataway, NJ, 1988.

Breuker, Joost, et al., "A Shell for Intelligent Help Systems," Proceedings of the 10th International Joint Conference on Artificial Intelligence, Milan, Italy, August 23-28, 1987.

Briot, Jean-Pierre and Pierre Cointe, "A Uniform Model for Object-Oriented Languages Using the Class Abstraction," Proceedings of the 10th International Joint Conference on Artificial Intelligence, Milan, Italy, August 23-28, 1987.

Brower, Ronald W., Larry E. French, and Richard W. Linderman, "LISP Garbage Collection Using Content-Addressable Memory," Proceedings of the IEEE 1987 National Aerospace and Electronics Conference, IEEE Service Center, Piscataway, NJ, 1987.

Brown, David and John M. Carson, "Embedded Expert Systems for Avionics Applications," IEEE/AIAA Digital Avionics Systems Conference Proceedings, IEEE Service Center, Piscataway, NJ, 1986.

Brubaker[1], David L., "Fuzzy-logic Basics: Intuitive Rules Replace Complex Math," EDN, Volume 37, No. 13, June 18, 1992.

Brubaker[2], David L., "Fuzzy-logic System Solves Control Problem," EDN, Volume 37, No. 13, June 18, 1992.

Brubaker, David L., "Everything You Always Wanted to Know About Fuzzy Logic," EDN, March 31, 1993.

Brule, James F., "Fuzzy Systems - A Tutorial," 1985.

Butler, G. F. and M. J. Corbin, "FLEX: FORTRAN Library for Expert Systems," RAE Working Paper, MM 273/88, December 1988.

Carlton, Kyp A., "Artificial Intelligence Supportability," Proceedings of the IEEE 1987 National Aerospace and Electronics Conference, IEEE Service Center, Piscataway, NJ, 1987.

Carpenter, Gail A. and Stephen Grossberg, "A Self-Organizing Neural Network for Supervised Learning, Recognition, and Prediction," IEEE Communications, Volume 30, No. 9, September 1992.

Chappell, Alan R., "Knowledge-Based Reasoning in the Paladin Tactical Decision Generation System," IEEE/AIAA 11th Digital Avionics Systems Conference Proceedings, IEEE Service Center, Piscataway, NJ, 1992.

Charniak, Eugene and Drew McDermott, Introduction to Artificial Intelligence, Addison-Wesley Publishing Company, Reading, MA, 1985.

Chen, David C., "An Expert Planner for the Dynamic Flight Environment," Proceedings of the IEEE 1985 National Aerospace and Electronics Conference, IEEE Service Center, Piscataway, NJ, 1985.

Clarkson, Douglas, "Living Computers," Electronics World and Wireless World, March 1990.

Clocksin, W. F. and C. F. Mellish, Programming in Prolog, Springer-Verlag Berlin, Heidelberg, Germany, 1984.

Coats, Pamela K., "Why Expert Systems Fail," Uma Gupta ed., Validating and Verifying Knowledge-Based Systems, IEEE Computer Society Press, Los Alamitos, CA, 1991.

Cochran, Keith G., "Artificial Intelligence Techniques Applied to Vehicle Management System Diagnostics," IEEE/AIAA 10th Digital Avionics Systems Conference Proceedings, IEEE Service Center, Piscataway, NJ, 1991.

Cohen, Paul R., Heuristic Reasoning About Uncertainty: An Artificial Intelligence Approach, Pittman Advanced Publishing Program, Boston, MA, 1985.

Coleman, Lisa A., "Lockheed First to Test Neural Net Chip," Military and Aerospace Electronics, Volume 4, No. 3, March 15, 1993.

Conner, Doug, "Designing a Fuzzy Logic Control System," EDN, Volume 38, No. 7, March 1993.

Cox, Earl, "Fuzzy Fundamentals," IEEE Spectrum, Volume 29, No. 10, October 1992.

Cox, Earl, "Adaptive Fuzzy Systems," IEEE Spectrum, Volume 30, No. 2, February 1993.

Culbert, Chris, Gary Riley, and Robert T. Savely, "Approaches to the Verification of Rule-Based Expert Systems," Uma G. Gupta ed., Validating and Verifying Knowledge-Based Systems, IEEE Computer Society Press, Los Alamitos, CA, 1991.

Curran, Jim, Trends in Advanced Avionics, Iowa State University Press, 1992.

Day, P. O. and M. K. Hook, "An AI-Based Fault Diagnosis Aid for Complex Electronic Systems," IEEE/AIAA 8th Digital Avionics Systems Conference Proceedings, IEEE Service Center, Piscataway, NJ, 1988.

Daysh, Colin, et al., "A NASA/RAE Cooperation in the Development of a Real-Time Knowledge Based Autopilot," AGARD Conference Proceedings 499: Machine Intelligence for Aerospace Electronic Systems, Advisory Group for Aerospace Research and Development, Neuilly Sur Seine, France, May 1991.

Decision Aiding System for Commercial Aircraft, Sextant Avionique Flight Control Systems Division, Briefing to FAA Technical Center, Atlantic City International Airport, NJ, February 1993.

Deeb, Joyce M. and Andrew G. Philpot, "Data Management in Large-Scale AI Systems," Proceedings of the IEEE 1988 National Aerospace and Electronics Conference, IEEE Service Center, Piscataway, NJ, 1988.

Delgrande, James P., "A Formal Approach to Learning from Examples," Proceedings of the 10th International Joint Conference on Artificial Intelligence, Milan, Italy, August 23-28, 1987.

Denning, Peter J., "The Science of Computing," American Scientist, Volume 74, January-February 1986.

DeWalt, Michael, Avionics Training Seminar, Kansas City, MO, December 8-9, 1992.

DOD-STD-2167A, "Defense System Software Development," February 29, 1988.

Donker, J. C., "Reasoning with Uncertain and Incomplete Information in Aerospace Applications," AGARD Conference Proceedings 499: Machine Intelligence for Aerospace Electronic Systems, Advisory Group for Aerospace Research and Development, Neuilly Sur Seine, France, May 1991.

Dreyfus, Hubert and Stuart Dreyfus, "Why Computers May Never Think Like People," Technology Review, January 1986.

Dupuy, Serge, "Contribution of Expert Systems to Avionics Advance Application on AS30 Laser Weapon System," Software Engineering and its Application to Avionics, AGARD-CP-439, Advisory Group for Aerospace Research and Development, Neuilly Sur Seine, France, November 1988.

Eimer, Erhard O., "Decision Aids: Disasters Waiting to Happen?" Proceedings of the IEEE 1987 National Aerospace and Electronics Conference, IEEE Service Center, Piscataway, NJ, 1987.

Elwell, D., et al., Avionic Data Bus Integration Technology, DOT/FAA/CT-91/19, U.S. Department of Transportation, Federal Aviation Administration, December 1991.

Elwell, D. and N. VanSuetendael, Software Quality Metrics, DOT/FAA/CT-91/1, U.S. Department of Transportation, Federal Aviation Administration, 1991.

Endres, Gunther, "Towards the 'Intelligent' Aircraft," Interavia Aerospace Review, Volume 46, April 1991.

Engelmore, Robert and Tony Morgan, Blackboard Systems, Addison-Wesley Publishing Company, New York, NY, 1988.

Farrell, Robert, "Intelligent Case Selection and Presentation," Proceedings of the 10th International Joint Conference on Artificial Intelligence, Milan, Italy, August 23-28, 1987.

Fey, J., et al., "Expert System for the Tornado Ground-Based Check-Out System," AGARD Conference Proceedings 499: Machine Intelligence for Aerospace Electronic Systems, Advisory Group for Aerospace Research and Development, Neuilly Sur Seine, France, May 1991.

Feyock, Stefan and Stamos Karamouzis, "Design of an Intelligent Information System for In-Flight Emergency Assistance," 1991 Goddard Conference on Space Applications of Artificial Intelligence, NASA Conference Proceeding CP-3110, May 1991.

Feyock, Stefan and Dalu Li, "Simulation-Based Reason about the Physical Propagation of Fault Effects," 1990 Goddard Conference on Space Applications of Artificial Intelligence, NASA Conference Proceeding CP-3068, May 1990.

Fikes, Richard and Tom Kehler, "The Role of Frame-Based Representation in Reasoning," Communications of the ACM, Volume 28, No. 9, September 1985.

A Framework for Embedded Avionics Expert Systems, Boeing Advanced Systems Public Relations Sales Brochure, Seattle, WA.

Frankovich, Ken, Ken Pedersen, and Stanley Bernsteen, "Expert System Applications to the Cockpit of the '90s," Proceedings of the IEEE 1985 National Aerospace and Electronics Conference, IEEE Service Center, Piscataway, NJ, 1985.

French, Scott W. and David O. Hamilton, Expert Systems V&V Guidelines Workshop, February 1992.

Friedland, Peter, "Special Section on Architectures for Knowledge-Based Systems," Communications of the ACM, Volume 28, No. 9, September 1985.

Frisch, Alan M., "Inference Without Chaining," Proceedings of the 10th International Joint Conference on Artificial Intelligence, Milan, Italy, August 23-28, 1987.

Frueh, George T., "Semiconductor Highlight," Electronic Component News, Volume 37, No. 7, July 1993.

The Fuzzy Source, Volume 2, Issue 1, Togai Infralogic, Inc., Irvine, CA, 1992.

Gabriel, Richard F., "Cockpit Automation," Kim M. Cardosi and M. Stephen Huntley eds., Human Factors for Flight Deck Certification Personnel, Draft, DOT-VNTSC-FAA-93-4, U.S. Department of Transportation, Research and Special Programs Administration, May 1993.

Galdes, Deborah K. and Philip J. Smith, "Building an Intelligent Tutoring System: Some Guidelines from a Study of Human Tutors," Proceedings of the Human Factors Society 34th Annual Meeting, Santa Monica, CA, 1990.

Geddes, Norman D., "Verification and Validation Testing of the Pilot's Associate," IEEE/AIAA 10th Digital Avionics Systems Conference Proceedings, IEEE Service Center, Piscataway, NJ, 1991.

Genesereth, Michael R. and Matthew L. Ginsberg, "Logic Programming," Communications of the ACM, Volume 28, No. 9, September 1985.

Gibbons, Greg D., et al., "Expert Systems: They Ain't What They Used to Be!" Proceedings of the IEEE 1988 National Aerospace and Electronics Conference, IEEE Service Center, Piscataway, NJ, 1988.

Gibbons, Greg D., Jonathan S. Abel, and Jill V. Josselyn, "ASAP: AI-Based Situation Assessment and Planning," Proceedings of the IEEE 1988 National Aerospace and Electronics Conference, IEEE Service Center, Piscataway, NJ, 1988.

Glickstein, Ira, Steve Ruberg, and Lt. John Marsh, "Database Management for Integrated Avionics System," Proceedings of the National Aerospace and Electronics Conference, IEEE Service Center, Piscataway, NJ, 1992.

Glover, Richard D., "Application Experience with the NASA Aircraft Interrogation and Display System: A Ground-Support Equipment for Digital Flight Systems," IEEE/AIAA 5th Digital Avionics Systems Conference, Seattle, WA, October 31-November 3, 1983.

Glover, Richard D. and Richard R. Larson, "A Knowledge Based Application of the Extended Aircraft Interrogation and Display System," NASA Technical Memorandum 4327, Dryden Flight Research Facility, Edwards, CA, October 1991.

Gordon, Sallie E., Rhonda A. Kinghorn, and Kim A. Schmierer, "Representing Expert Knowledge for Instructional System Design: A Case Study," Proceedings of the Human Factors Society 35th Annual Meeting, Santa Monica, CA, 1991.

Gordon, Sallie E. and Vickie Lewis, "Knowledge Engineering for Hypertext Instructional Systems," Proceedings of the Human Factors Society 34th Annual Meeting, Santa Monica, CA, 1990.

Graham, Joyce M., "An Intelligent Spatial Database System for Interaction with a Real-Time Piloting Expert System," Proceedings of the IEEE 1987 National Aerospace and Electronics Conference, IEEE Service Center, Piscataway, NJ, 1987.

Green, Christopher J. R., "On the Use of Requirements in the Development of Knowledge-Based Systems," Uma G. Gupta ed., Validating and Verifying Knowledge-Based Systems, IEEE Computer Society Press, Los Alamitos, CA, 1991.

Green, Christopher J. R. and Marlene M. Keyes, "Verification and Validation of Expert Systems," Uma G. Gupta ed., Validating and Verifying Knowledge-Based Systems, IEEE Computer Society Press, Los Alamitos, CA, 1991.

Grimshaw, Capt. Jeffrey D. and Craig S. Anken, "A Distributed Environment for Testing Cooperating Expert Systems," AGARD Conference Proceedings 499: Machine Intelligence for Aerospace Electronic Systems, Neuilly Sur Seine, France, September 1991.

Hall, Lawrence O. and Abraham Kandel, "The Evolution from Expert Systems to Fuzzy Expert Systems," Fuzzy Expert Systems, Abraham Kandel ed., CRC Press, Boca Raton, FL, 1992.

Hammer, John, M., "Verification and Validation of Knowledge Bases in Associate Systems," DARPA Symposium on Associate Technology, Fairfax, VA, June 6-7, 1991.

Hammerstrom, Dan, "Working with Neural Networks," IEEE Spectrum, Volume 30, No. 7, July 1993.

Harvey, David S., "Artificial Intelligence in Combat Avionics," Avionics, Volume 17, No. 1, January 1993.

Hayes-Roth, Frederick, "Rule-Based Systems," Communications of the ACM, Volume 28, No. 9, September 1985.

Hertz, John, Anders Krogh, and Richard G. Palmer, Introduction to the Theory of Neural Computation, Addison Wesley Publishing Company, Redwood City, CA, 1991.

Hillman, David V., "Integrating Neural Nets and Expert Systems," AI Expert, June 1990.

Hillman, Donald J., "Artificial Intelligence," Human Factors, Volume 27, No. 1, 1985.

Holla, K. and B. Benninghofen, "A Threat Management System," AGARD Conference Proceedings 499: Machine Intelligence for Aerospace Electronic Systems, Advisory Group for Aerospace Research and Development, Neuilly Sur Seine, France, September 1991.

Hui, Patrick J. and Amiya R. Nayak, "EXNAV: An Intelligent Sensor Integrator," Proceedings of the IEEE 1988 National Aerospace and Electronics Conference, IEEE Service Center, Piscataway, NJ, 1988.

Humphrey, Timothy L., "Using PROLOG in Natural Language Systems," Proceedings of the IEEE 1985 National Aerospace and Electronics Conference, IEEE Service Center, Piscataway, NJ, 1985.

Hunter, Lawrence and David J. States, "Bayesian Classification of Protein Structure," IEEE Expert, Volume 7, No. 4, August 1992.

The Huntington Group, Huntington Technical Brief, No. 34, Menlo Park, CA, January 1993.

Iline, Herman and Henry Kanoui, "Extending Logic Programming to Object Programming: The System LAP," Proceedings of the 10th International Joint Conference on Artificial Intelligence, Milan, Italy, August 23-28, 1987.

Improving Aircraft Safety, National Academy of Sciences, Washington, DC, 1980.

Intel[1] Corporation, 80170NW - Electrically Trainable Analog Neural Network, Santa Clara, CA, May 1990.

Intel[2] Corporation, "New Wave" Computing? - An Introduction to Neural Networks, November 1990.

Iorgulescu, Daniela T., David F. Giere, and Carol S. Giffen, "Artificial Expertise in Systems Engineering," IEEE/AIAA 10th Digital Avionics Systems Conference Proceedings, IEEE Service Center, Piscataway, NJ, 1991.

Irwin, J. David, Basic Engineering Circuit Analysis, 2nd ed., Macmillan Publishing Company, New York, NY, 1987.

Jafar, Musa and A. Terry Bahill, "Interactive Verification of Knowledge-Based Systems," IEEE Expert, Volume 8, No. 1, 1993.

Janowitz, Joan, Handbook-Volume III Digital Systems Validation Book Plan, DOT/FAA/CT-93/16, U.S. Department of Transportation, Federal Aviation Administration, July 1993.

Johnson, Sally C. and Ricky W. Butler, "Design for Validation," IEEE/AIAA 10th Digital Avionics Systems Conference Proceedings, IEEE Service Center, Piscataway, NJ, 1991.

Johnson, William B., "Advanced Technology for Aviation Maintenance Training: An Industry Status Report and Development Plan," Proceedings of the Human Factors Society 34th Annual Meeting, Santa Monica, CA, 1990.

Johnson, William B. and Jeffrey E. Norton, "Integrated Systems for Training, Aiding, and Information Retrieval," Proceedings for the East-West Conference in Emerging Computer Technology in Education, Moscow, Russia, 1992.

Kang, Yue and A. Terry Bahill, "A Tool for Detecting Expert System Errors," Uma G. Gupta ed., Validating and Verifying Knowledge-Based Systems, IEEE Computer Society Press, Los Alamitos, CA, 1991.

Kartalopoulos, Stamatios V., "A Plateau of Performance?," IEEE Communications, Volume 30, No. 9, September 1992.

Kearsley, G. P., ed., Artificial Intelligence and Instruction, Addison-Wesley Publishing Company, Reading, MA, 1987.

Keller, Forrest L., "Basic Electronic Systems Certification," IEEE/AIAA 11th Digital Avionics Systems Conference Proceedings, IEEE Service Center, Piscataway, NJ, 1992.

Khaksari, Gholam H., "Expert Diagnostic System," The First International Conference on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems, June 1-3, 1988.

Klein, Gary and Roberta Calderwood, "Human Factors Considerations for Expert Systems," Proceedings of the IEEE 1986 National Aerospace and Electronics Conference, IEEE Service Center, Piscataway, NJ, 1986.

Klimasauskas, Casimir C., "Neural Networks: An Engineering Perspective," IEEE Communications, Volume 30, No. 9, September 1992.

Kosko, Bart, Neural Networks and Fuzzy Systems, Prentice Hall Publishing Company, Englewood Cliffs, NJ, 1992.

Krogmann, Uwe K., "Introduction to Neural Computing and Categories of Neural Network Applications to Guidance, Navigation, and Control," AGARD Lecture Series 179: Artificial Neural Network Approaches in Guidance and Control, Advisory Group for Aerospace Research and Development, Neuilly Sur Seine, France, September 1991.

Lawler, R. W. and M. Yazdani, eds., Artificial Intelligence and Education Volume 1: Learning Environments and Tutoring Systems, Ablex, Norwood, NJ, 1987.

Lawrence, Jeannette, Introduction to Neural Networks, Sylvia Ludeking ed., California Scientific Software, 1991.

Layer, Kevin D. and Chris Richardson, "LISP," Communications of the ACM, Volume 34, No. 9, September 1991.

Lea, Robert N., "Application of Fuzzy Sets to Rule-Based Expert System Development," Telematics and Informatics, Volume 6, Nos. 3 and 4, 1989.

Leavitt, C. A. and D. M. Smith, "Integrated Dynamic Planning in the Pilot's Associate," AIAA Guidance, Navigation, and Control Conference, Part 1, Boston, MA, August 14-16, 1989.

Lee, Chuen-Chien, "A Self-Learning Rule-Based Controller Employing Approximate Reasoning and Neural Net Concepts," International Journal of Intelligent Systems, Volume 6, 1991.

Leeper, Kenneth R., "Artificial Intelligence Programming in Ada," IEEE/AIAA 9th Digital Avionics Systems Conference Proceedings, IEEE Service Center, Piscataway, NJ, 1990.

Legg, Gary, "Special Tools and Chips Make Fuzzy Logic Simple," EDN, Volume 37, No. 14, July 6, 1992.

Legg, Gary, "Microcontrollers Embrace Fuzzy Logic," EDN, Volume 38, No. 19, September 1993.

Leinweber, David and David Carleton, "The PICON Real-Time Expert System Tool," Proceedings of the IEEE 1986 National Aerospace and Electronics Conference, IEEE Service Center, Piscataway, NJ, 1986.

Lenorovitz, David R. and Ray A. Reaux, "Integrating Human Factors Guidance Information within the USI Design/Rapid Prototyping Process," Proceedings of the IEEE 1986 National Aerospace and Electronics Conference, IEEE Service Center, Piscataway, NJ, 1986.

Level5 Object User's Guide, Information Builders, Inc., New York, NY, 1992.

Levi, Keith R., et al., "An Explanation-Based-Learning Approach to Knowledge Compilation," IEEE Expert, Volume 7, No. 3, June 1992.

Levine, Robert I., Diane E. Drang, and Barry Edelson, A Comprehensive Guide to AI and Expert Systems, McGraw-Hill Publishing Company, New York, NY, 1986.

Lim, Ee-Peng and Vladimir Cherkassky, "Semantic Networks and Associative Databases," IEEE Expert, Volume 7, No. 4, August 1992.

Lischke, Michael P. and Kenneth L. Meyer, "TEAMS: Technical Aircraft Expert Maintenance System," Proceedings of the IEEE 1992 National Aerospace and Electronics Conference, IEEE Service Center, Piscataway, NJ, 1992.

Lizza, Carl, Sheila Banks, and Michael A. Whelan, "Pilot's Associate: Evolution of a Functional Prototype," AGARD Conference Proceedings 499: Machine Intelligence for Aerospace Electronic Systems, Advisory Group for Aerospace Research and Development, Neuilly Sur Seine, France, September 1991.

Lizza, Carl and Carl Friedlander, "The Pilot's Associate: A Forum for the Integration of Knowledge Based Systems and Avionics," Proceedings of the IEEE 1988 National Aerospace and Electronics Conference, IEEE Service Center, Piscataway, NJ, 1988.

Lovesey, E. J. and R. I. Davis, "Integrating Machine Intelligence into the Cockpit to Aid the Pilot," AGARD Conference Proceedings 499: Machine Intelligence for Aerospace Electronic Systems, Advisory Group for Aerospace Research and Development, Neuilly Sur Seine, France, September 1991.

MacDonald, Jim and Klaus K. Obermeier, "Towards a Taxonomy for Knowledge Representation Schemas," Proceedings of the IEEE 1986 National Aerospace and Electronics Conference, IEEE Service Center, Piscataway, NJ, 1986.

Madni, Azad M., "HUMANE: A Knowledge-Based Simulation Environment for Human-Machine Function Allocation," Proceedings of the IEEE 1988 National Aerospace and Electronics Conference, IEEE Service Center, Piscataway, NJ, 1988.

Manheimer, Jerry M. and Thomas A. Kanarski, "The Application of Psychological Scaling Techniques in Modeling Expert Knowledge," Proceedings of the IEEE 1986 National Aerospace and Electronics Conference, IEEE Service Center, Piscataway, NJ, 1986.

Marcot, Bruce, "Testing Your Knowledge Base," Uma G. Gupta ed., Validating and Verifying Knowledge-Based Systems, IEEE Computer Society Press, Los Alamitos, CA, 1991.

Marcus, Sandra, "Salt: A Knowledge Acquisition Tool that Checks and Helps Test a Knowledge Base," Uma G. Gupta ed., Validating and Verifying Knowledge-Based Systems, IEEE Computer Society Press, Los Alamitos, CA, 1991.

Masotto, Tom, Carol Babikyan, and Richard Harper, "Knowledge Representation into Ada Parallel Processing," NASA Contractor Report 187451, NASA Langley Research Center, Hampton, VA, July 1990.

McCay, Scott, "CLIM: The Common Lisp Interface Manager," Communications of the ACM, Volume 34, No. 9, September 1991.

McCoy, Michael S. and Randy M. Boys, "Human Performance Models Applied to Intelligent Decision Support Systems," Proceedings of the IEEE 1987 National Aerospace and Electronics Conference, IEEE Service Center, Piscataway, NJ, 1987.

McManus, John W., "Design and Analysis Tools for Concurrent Blackboard Systems," IEEE/AIAA 10th Digital Avionics Systems Conference Proceedings, IEEE Service Center, Piscataway, NJ, 1991.

McNeese, Michael D., "Humane Intelligence: A Human Factors Perspective for Developing Intelligent Cockpits," Proceedings of the IEEE 1986 National Aerospace and Electronics Conference, IEEE Service Center, Piscataway, NJ, 1986.

McNeese, Michael D. and Brian S. Zaff, "Knowledge as Design: A Methodology for Overcoming Knowledge Acquisition Bottlenecks in Intelligent Interface Design," Proceedings of the Human Factors Society 35th Annual Meeting, Santa Monica, CA, 1991.

McNulty, Christa, "Knowledge Engineering for a Piloting Expert System," Proceedings of the IEEE 1987 National Aerospace and Electronics Conference, IEEE Service Center, Piscataway, NJ, 1987.

Mehrotra, Mala, "Rule Groupings: A Software Engineering Approach Towards Verification of Expert Systems," NASA Contractor Report 4372, NASA Langley Research Center, Hampton, VA, May 1991.

Mehrotra, Mala and Chris Wild, "Multi-Viewpoint Clustering Analysis," Workshop on Verification, Validation, and Testing of Intelligent Systems, 9th IEEE Conference on Artificial Intelligence for Applications, Orlando, FL, March 2, 1993.

Minges, Mark E., "Integrated Communications, Navigation, Identification, Avionics (ICNIA) Expert Systems for Fault Tolerant Avionics," AGARD Conference Proceedings 499: Machine Intelligence for Aerospace Electronic Systems, Advisory Group for Aerospace Research and Development, Neuilly Sur Seine, France, September 1991.

Mitsubishi Electric, Product Literature, Japan, 1992.

Mitta, Deborah, Newton C. Ellis, and Dick B. Simmons, "Human Factors Data: Knowledge Sources for Intelligent Design Associates," Proceedings of the Human Factors Society 34th Annual Meeting, Santa Monica, CA, 1990.

Miyamoto, Sadaaki, Fuzzy Sets in Information Retrieval and Cluster Analysis, Kluwer Academic Publishers, Boston, MA, 1990.

Morley, Richard E. and William A. Taylor, "What is Artificial Intelligence?," Digital Design, April 1986.

Morley, Richard E. and William A. Taylor, "Artificial Intelligence Basics: Hardware Follows Software," Digital Design, May 1986.

Morley, Richard E. and William A. Taylor, "Why Bother with Expert Systems?," Digital Design, July 1986.

Muller, Hans, "LispView: Leverage Through Integration," Communications of the ACM, Volume 34, No. 9, September 1991.

Murphy, Thomas, "AI Apprentice," AI Expert, Volume 8, No. 4, April 1993.

Naser, Joseph A., "Nuclear Power Plant Expert System Verification and Validation," Uma G. Gupta ed., Validating and Verifying Knowledge-Based Systems, IEEE Computer Society Press, Los Alamitos, CA, 1991.

The National Plan for Aviation Human Factors, Volume 1, Draft, U.S. Department of Transportation, Federal Aviation Administration, April 1991.

The National Plan for Aviation Human Factors, Volume 2, Draft, U.S. Department of Transportation, Federal Aviation Administration, November 1990.

Nelson, Dale E. and Steven K. Rogers, "A Taxonomy of Neural Network Optimality," Proceedings of the IEEE 1992 National Aerospace and Electronics Conference, IEEE Service Center, Piscataway, NJ, 1992.

NeuroDynamx Literature, NDX Neural Accelerators, NeuroDynamx, Inc., Boulder, CO, 1994.

Ng, Andrew, "A Cooperative Expert System Architecture for Embedded Avionics," Proceedings of the IEEE 1988 National Aerospace and Electronics Conference, IEEE Service Center, Piscataway, NJ, 1988.

Nilsson, Nils J., Principles of Artificial Intelligence, Tioga Publishing Company, Palo Alto, CA, 1980.

Nordwall, Bruce D., "Cockpit Safety Researchers Eye Verbal Advice System," Aviation Week & Space Technology, Volume 137, No. 14, October 5, 1992.

Normile, Dennis, "Mimicking the Brain," Popular Science, November 1992.

Norton, J. E. and W. B. Johnson, "Microcomputer Intelligence for Technical Training (MITT): The Evolution of an Intelligent Tutoring System," Proceedings, NASA 1991 Conference on Intelligent Computer-Aided Training, Houston, TX, 1991.

Obermeier, Klaus K. and Janet J. Barron, "Time to Get Fired Up," Byte, August 1989.

O'Keefe, Robert M., Osman Balci, and Eric P. Smith, "Validating Expert System Performance," Uma G. Gupta ed., Validating and Verifying Knowledge-Based Systems, IEEE Computer Society Press, Los Alamitos, CA, 1991.

O'Keefe, Robert M. and Daniel O'Leary, "Expert System Verification and Validation: A Survey and Tutorial," Workshop on Verification, Validation, and Testing of Intelligent Systems, 9th IEEE Conference on Artificial Intelligence for Applications, Orlando, FL, March 2, 1993.

O'Leary, Daniel E., "Measuring the Quality of Computer Model Performance," Workshop on Verification, Validation, and Testing of Intelligent Systems, 9th IEEE Conference on Artificial Intelligence for Applications, Orlando, FL, March 2, 1993.

Omron Electronics, Inc., "An Introduction to Fuzzy Logic and its Application in Control Systems," Fuzzy Logic, a 21st Century Technology, 1991.

Ostgaard, John C. and D. Reed Morgan, "PAVE PILLAR and PAVE PACE: Avionics System Architecture for the 21st Century," ERA Seminar Proceedings: Military Avionics Architectures for Today and Tomorrow, ERA Report No. 88-0437, ERA Technology Ltd., Surrey, England, 1989.

Ovenden, C. R., "AI Applications to Tactical Decision Aids," IEEE/AIAA 8th Digital Avionics Systems Conference Proceedings, IEEE Service Center, Piscataway, NJ, 1988.

Palko, Katherine and Randy Boys, "Augmenting the Pilot/Vehicle Interface Through Capability Assessment," Proceedings of the IEEE 1987 National Aerospace and Electronics Conference, IEEE Service Center, Piscataway, NJ, 1987.

Papp, M. L., W. R. Braisted, and R. F. Taylor, "A Prototype Expert System for Analysis of Turbine Engine Components," Proceedings of the IEEE 1992 National Aerospace and Electronics Conference, IEEE Service Center, Piscataway, NJ, 1992.

Pearce, Michael, et al., "Case-Based Design Support," IEEE Expert, 1992.

Pederson, Ken, "Part I: The Well-Structured Rule, the Basic Building Block for Knowledge Bases," Uma Gupta ed., Validating and Verifying Knowledge-Based Systems, IEEE Computer Society Press, Los Alamitos, CA, 1989.

Pederson, Ken, "Well-Structured Knowledge Bases: Part II," Uma Gupta ed., Validating and Verifying Knowledge-Based Systems, IEEE Computer Society Press, Los Alamitos, CA, 1989.

Pederson, Ken, "Well-Structured Knowledge Bases: Part III," Uma Gupta ed., Validating and Verifying Knowledge-Based Systems, IEEE Computer Society Press, Los Alamitos, CA, 1989.

Penn, Brian S., "Design Considerations in the Development of the Meta-Expert System," Proceedings of the IEEE 1986 National Aerospace and Electronics Conference, IEEE Service Center, Piscataway, NJ, 1986.

Perlin, Mark, Revision of "An OPS5 Primer: Introduction to Rule-Based Expert Systems," by Porter D. Sherman and John C. Martin, IEEE Communications, Volume 31, No. 1, January 1993.

Piers, M. A. and J. C. Donker, "A Knowledge Based Assistant for Diagnosis in Aircraft Maintenance," AGARD Conference Proceedings 449: Machine Intelligence for Aerospace Electronic Systems, Advisory Group for Aerospace Research and Development, Neuilly Sur Seine, France, September 1991.

Pilet, S. C. and R. O. Stenerson, "Avionics Expert Systems: The Transition to Embedded Systems," Proceedings of the IEEE 1987 National Aerospace and Electronics Conference, IEEE Service Center, Piscataway, NJ, 1987.

Plant, Robert T., "Pattern Directed Inference Systems: A Development Methodology," Workshop on Verification, Validation, and Testing of Intelligent Systems, 9th IEEE Conference on Artificial Intelligence for Applications, Orlando, FL, March 2, 1993.

Plant, Robert T. and Panagiotis Tsoumpas, "An Integrated Methodology for Knowledge-Based System Development," Workshop on Verification, Validation, and Testing of Intelligent Systems, 9th IEEE Conference on Artificial Intelligence for Applications, Orlando, FL, March 2, 1993.

Plutowski, Mark, "VOSS-VHSIC Hosted Expert Systems," Proceedings of the IEEE 1986 National Aerospace and Electronics Conference, IEEE Service Center, Piscataway, NJ, 1986.

Pohlmann, Lawrence D. and J. Roland Payne, "Pilot's Associate Demonstration One: A Look Back and Ahead," Proceedings of the IEEE 1986 National Aerospace and Electronics Conference, IEEE Service Center, Piscataway, NJ, 1986.

Polson, Martha Campbell, "Status and Future Directions of Intelligent Tutoring Systems," Proceedings of the Human Factors Society 33rd Annual Meeting, Santa Monica, CA, 1989.

Prevot, T., R. Onken, and H. L. Dudek, "Knowledge-Based Planning for Controlled Airspace Flight Operation as Part of a Cockpit Assistant," AGARD Conference Proceeding 504: Air Vehicle Mission Control and Management, Advisory Group for Aerospace Research and Development, Neuilly Sur Seine, France, October 1991.

Psotka, Joseph, "Advancing the Mind/Machine Interface: Qualitative Simulations, Hypertext, and Natural Language Processing," Proceedings of the Human Factors Society 33rd Annual Meeting-1989, Santa Monica, CA, 1989.

Pukite, P. R., J. Pukite, and D. S. Barnhart, "Expert System for Redundancy and Reconfiguration Management," Proceedings of the IEEE 1992 National Aerospace and Electronics Conference, IEEE Service Center, Piscataway, NJ, 1992.

Rao, Ming and Tsung-Shann Jiang, "A New Method to Design Intelligent Control Systems," Proceedings of the IEEE 1988 National Aerospace and Electronics Conference, IEEE Service Center, Piscataway, NJ, 1988.

Rich, Elaine and Kevin Knight, Artificial Intelligence, 2nd ed., McGraw-Hill Publishing Company, New York, NY, 1991.

Roberts, K., "TACAID - A Knowledge Based System for Tactical Decision Making," AGARD Conference Proceedings 499: Machine Intelligence for Aerospace Electronic Systems, Advisory Group for Aerospace Research and Development, Neuilly Sur Seine, France, September 1991.

Rocha, A. F., et al., "The Physiology of the Expert System," Fuzzy Expert Systems, Abraham Kandel ed., CRC Press, Boca Raton, FL, 1992.

Rock, Denny, Don Malkoff, and Ron Stewart, "AI and Aircraft Health Monitoring," AI Expert, February 1993.

Rolston, David W., Principles of Artificial Intelligence and Expert Systems Development, McGraw-Hill Publishing Company, New York, NY, 1988.

Roth, Al, "The Practical Application of PROLOG," AI Expert, Volume 8, No. 4, April 1993.

Rothman, Michael J., "IJCNN '92," IEEE Expert, Volume 8, No. 1, February 1993.

Rouse, William B., Norman D. Geddes, and Renwick E. Curry, "An Architecture for Intelligent Interfaces: Outline of an Approach to Supporting Operators of Complex Systems," Proceedings of the IEEE 1986 National Aerospace and Electronics Conference, IEEE Service Center, Piscataway, NJ, 1986.

Rouse, William B., Norman D. Geddes, and John M. Hammer, "Computer-Aided Fighter Pilots," IEEE Spectrum, Volume 27, No. 3, March 1990.

RTCA/DO-160C, "Environmental Conditions and Test Procedures for Airborne Equipment," Radio Technical Commission for Aeronautics, Washington, DC, December 1989.

RTCA/DO-178B, "Software Considerations in Airborne Systems and Equipment Certification," Radio Technical Commission for Aeronautics, Washington, DC, December 1992.

RTCA Paper No. 548-93/SC180-18, Radio Technical Commission for Aeronautics, Washington, DC, February 2, 1994.

Rudolph, F. M., D. A. Homoki, and G. A. Sexton, "Diverter' Decision Aiding for In-Flight Diversions," NASA Contractor Report 182070, NASA Langley Research Center, Hampton, VA, August 1990.

Rushby, John, "Quality Measures and Assurance for AI Software," NASA Contractor Report 4187, NASA Langley Research Center, Hampton, VA, October 1988.

Rushby, John and Judith Crow, "Evaluation of an Expert System for Fault Detection, Isolation, and Recovery in the Manned Maneuvering Unit," NASA Contractor Report 187466, NASA Langley Research Center, Hampton, VA, December 1990.

Rushby, John and R. Alan Whitehurst, "Formal Verification of AI Software," NASA Contractor Report 181827, NASA Langley Research Center, Hampton, VA, February 1989.

Ryder, Joan M., et al., "An Integrated Embedded Training and Decision Aiding Design Methodology," Proceedings of the Human Factors Society 34th Annual Meeting, Santa Monica, CA, 1990.

Sadeghi, Tom and Gerry Mayville, "Fault-Tolerant, Flight-Critical Control Systems," AGARD Conference Proceedings 456: Fault Tolerant Design Concepts for Highly Integrated Flight Critical Guidance and Control Systems, Advisory Group for Aerospace Research and Development, Neuilly Sur Seine, France, October 13, 1989.

Savant, Jr., C. J., Martin S. Roden, and Gordon L. Carpenter, Electronic Circuit Design, An Engineering Approach, The Benjamin/Cummings Publishing Company, Inc., Menlo Park, CA, 1987

Schaefer, B. A., et al., "A Knowledge-Based Intelligent Tutoring System: ACQUIRE™-ITS," AGARD Conference Proceedings 499: Machine Intelligence for Aerospace Electronic Systems, Advisory Group for Aerospace Research and Development, Neuilly Sur Seine, France, September 1991.

Schultz, Roger D. and James R. Geissman[1], "Bridging the Gap Between Static and Dynamic Verification," Uma G. Gupta ed., Validating and Verifying Knowledge-Based Systems, IEEE Computer Society Press, Los Alamitos, CA, 1991.

Schultz, Roger D. and James R. Geissman[2], "Verification and Validation of Expert Systems," Uma G. Gupta ed., Validating and Verifying Knowledge-Based Systems, IEEE Computer Society Press, Los Alamitos, CA, 1991.

Schutte, Paul C. and Kathy H. Abbott, "An Artificial Intelligence Approach to Onboard Fault Monitoring and Diagnosis for Aircraft Applications," AIAA Guidance and Control Conference, Williamsburg, VA, 1986.

Schwartz, Daniel G. and George J. Klir, "Fuzzy Logic Flowers in Japan," IEEE Spectrum, Volume 29, No. 7, July 1992.

Schwartz, Tom J., "IEEE Gets Fuzzy," IEEE Expert, Volume 7, No. 3, June 1992.

Segre, Alberto Maria, "Applications of Machine Learning," IEEE Expert, Volume 7, No. 3, June 1992.

Seidman, Abraham N., "Neural Networks and Digital Avionics," IEEE/AIAA/NASA Digital Avionics Systems Conference Proceedings, IEEE Service Center, Piscataway, NJ, 1990.

Selcon, Stephen J., "Decision Support in the Cockpit: Probably a Good Thing?" Proceedings of the Human Factors Society 34th Annual Meeting, Santa Monica, CA, 1990.

Semple, W. G., "Development of Tactical Decision Aids," AGARD Conference Proceedings 499: Machine Intelligence for Aerospace Electronic Systems, Advisory Group for Aerospace Research and Development, Neuilly Sur Seine, France, September 1991.

Sewell, Daniel R. and William B. Johnson, "The Effects of Rapid Prototyping on User Behavior on System Design," Journal of the Washington Academy of Sciences, Volume 80, No. 2, June 1991.

Shaw[1], Julie, "Neural Network Resource Guide," AI Expert, Volume 8, No. 2, February 1993.

Shaw[2], Julie, "AI Language Resource Guide," AI Expert, Volume 8, No. 4, April 1993.

Shear, David, "Neural Network and Fuzzy Logic Combine to Create COP8 Code," EDN, Volume 38, No. 11, May 27, 1993.

Shelnutt, Jack B., et al., "Pilot's Associate Demonstration One: A Look Inside," Proceedings of the IEEE 1986 National Aerospace and Electronics Conference, IEEE Service Center, Piscataway, NJ, 1986.

Sheppard, John W. and William R. Simpson, "Expert Systems for Diagnostic Testing," Avionics, October 1992.

Shepherd, William T. and William B. Johnson, "Aircraft Maintenance Challenges and Human Factors Solutions," Seminar on Flight Safety and Human Factors, International Civil Aviation Organization, November 1991.

Shewhart, Mark, "Interpreting Statistical Process Control (SPC) Charts Using Machine Learning and Expert System Techniques," Proceedings of the IEEE

1992 National Aerospace and Electronics Conference, IEEE Service Center, Piscataway, NJ, 1992.

Silbert, Mark, et al., "A Tool for Development of AI Hybrid Systems," IEEE/AIAA Digital Avionics Systems Conference Proceedings, IEEE Service Center, Piscataway, NJ, 1986.

Simpson, Patrick K., "Neural Network Paradigms," AGARD Lecture Series 179: Artificial Neural Network Approaches in Guidance and Control, Advisory Group for Aerospace Research and Development, Neuilly Sur Seine, France, September 1991.

Sitz, Joel R. and Todd H. Vernon, "Flight Control System Design Factors for Applying Automated Testing Techniques," NASA Technical Memorandum 4242, October 1990.

Small, Ronald A. and Charles W. Howard, "A Real-Time Approach to Information Management in a Pilot's Associate," IEEE/AIAA 10th Digital Avionics Systems Conference Proceedings, IEEE Service Center, Piscataway, NJ, 1991.

Smith, Carolyn and Horace Sklar, "Embedded Expert Systems for Fault Detection and Isolation," IEEE/AIAA Digital Avionics Systems Conference Proceedings, IEEE Service Center, Piscataway, NJ, 1986.

Sorrells, Mark E., "A Time-Constrained Inference Strategy for Real Time Expert Systems," Proceedings of the IEEE 1985 National Aerospace and Electronics Conference, IEEE Service Center, Piscataway, NJ, 1985.

Steele, Jr., Guy L., Common LISP: The Language, Digital Press, 1984.

Tazelaar, Jane Morrill, "Neural Networks," Byte, August 1989.

Texas Instruments Literature, "Fuzzy Logic Moves into DSP Design Arena," Integration, North American Edition, Volume 11, No. 1, Texas Instruments, 1994.

Togai, Masaki and Horoyuki Watanabe, "Expert System on a Chip: An Engine for Approximate Reasoning," Fuzzy Expert Systems, Abraham Kandel ed., CRC Press, Boca Raton, FL, 1992.

Touretzky, David S. and Dean A. Pomerleau, "What's Hidden in the Hidden Layers?," Byte, August 1989.

Valstar, Jacob E., "Expert System Development on a Spreadsheet," Proceedings of the IEEE 1986 National Aerospace and Electronics Conference, IEEE Service Center, Piscataway, NJ, 1986.

Waibel, Alex and John Hampshire, "Building Blocks for Speech," Byte, August 1989.

Waldron, Vince, Harold W. Sharp, and Lt. Scott A. Stefanov, "Distributed Expert Management System," Proceedings of the IEEE 1987 National Aerospace and Electronics Conference, IEEE Service Center, Piscataway, NJ, 1987.

Wagner, Elaine A., "Onboard Automatic Aid and Advisory for Pilots of Control-Impaired Aircraft," Journal of Guidance, Control, and Dynamics, Volume 14, No. 4, July-August 1991.

Wang, David C. and James Thompson, "An Adaptive Data Sorter Based on Probabilistic Neural Networks," Topics in Engineering, Volume 3, AIL Systems, Inc., Deer Park, NY, 1992.

Warn, Keith, "Expert System Shell Standardization - The Controversy," Proceedings of the IEEE 1987 National Aerospace and Electronics Conference, IEEE Service Center, Piscataway, NJ, 1987.

Waterman, Donald A., A Guide to Expert Systems, Addison-Wesley Publishing Company, Reading, MA, 1986.

Weiss, Ray, "Fuzzy Coprocessor Performs up to 870,000 Evaluations per Second," EDN, Volume 38, No. 22, October 28, 1993.

Weiss, Sholom M. and Casimir A. Kulikowski, A Practical Guide to Expert Systems, Rowman and Allanheld Publishers, Totowa, NJ, 1984.

Wellens, Rodney A. and Michael D. McNeese, "A Research Agenda for the Social Psychology of Intelligent Machines," Proceedings of the IEEE 1987 National Aerospace and Electronics Conference, IEEE Service Center, Piscataway, NJ, 1987.

Wenger, E., Artificial Intelligence and Tutoring Systems: Computational and Cognitive Approaches to the Communication of Knowledge, Morgan Kauffmann, Los Altos, CA, 1987.

Whitaker, Leslie A., Richard H. Stottler, and James A. King, "Case-Based Reasoning: Taming the Similarity Heuristic," Proceedings of the Human Factors Society 34th Annual Meeting, Santa Monica, CA, 1990.

White, David J. and Elizabeth A. Bart, "An Expert System for VLSI Datapath Synthesis," Proceedings of the IEEE 1988 National Aerospace and Electronics Conference, IEEE Service Center, Piscataway, NJ, 1988.

Wickens, Christopher D., "Decision Making," Kim M. Cardosi and M. Stephen Huntley eds., Human Factors for Flight Deck Certification Personnel, Draft, DOT-VNTSC-FAA-93-4, U.S. Department of Transportation, Research and Special Programs Administration, May 1993.

Wickens, Christopher D., Engineering Psychology and Human Performance, 2nd ed., Harper Collins Publishing Company, 1992.

Widman, Lawrence E., Kenneth A. Loparo, and Norman R. Nielsen, Artificial Intelligence, Simulation, and Modeling, John Wiley and Sons Publishing Company, New York, NY, 1989.

Wiederholt, Bradley J., "MITT Writer: An Authoring System for Developing Intelligent Tutors for Complex Technical Domains," Galaxy Scientific Corporation, 1991.

Wiener, Earl L. and David C. Nigel, Human Factors in Aviation, Academic Press, Inc., San Diego, CA, 1988.

Wilber, George F. and E. Jean Dryer, "Strategic Real-Time Airborne Electronic Warfare Using Knowledge Base Techniques," Proceedings of the IEEE 1988 National Aerospace and Electronics Conference, IEEE Service Center, Piscataway, NJ, 1988.

Williams, Dr. R. M. and J. J. Davidson, "AI for RPVs, Sensor Driven Airborne Replanner, for a Robotic Aircraft Sensor Platform," AGARD Conference Proceedings 499: Machine Intelligence for Aerospace Electronic Systems, Advisory Group for Aerospace Research and Development, Neuilly Sur Seine, France, September 1991.

Williams, Tom, "Fuzzy Logic is Anything but Fuzzy," Computer Design, April 1992.

Winston, Patrick Henry, Artificial Intelligence 2nd Edition, Addison-Wesley Publishing Company, Reading, MA, 1984.

Wood, James, et al., "A Rule-Based Logic Design Assistant," Proceedings of the IEEE 1986 National Aerospace and Electronics Conference, IEEE Service Center, Piscataway, NJ, 1986.

Woolf, Beverly and Tom Murray, "A Framework for Representing Tutorial Discourse," Proceedings of the 10th International Joint Conference on Artificial Intelligence, Milan, Italy, August 23-28, 1987.

"Workshop on Verification, Validation, and Testing of Intelligent Systems," 9th IEEE Conference on Artificial Intelligence for Applications, Orlando, FL, March 2, 1993.

Xiaofeng, Li, "What's So Bad About Rule-Based Programming?," IEEE Software, September 1991.

Yen, Mike, "Lisp-to-Ada Reengineering Issues and Support Environments for Fielding Real-Time Systems," IEEE/AIAA 11th Digital Avionics Systems Conference Proceedings, IEEE Service Center, Piscataway, NJ, 1992.

Yops, Jr., Thomas E. and John R. Moore, "An Integrated Vehicle Management System Concept for Military Transports," Proceedings of the IEEE 1986 National Aerospace and Electronics Conference, IEEE Service Center, Piscataway, NJ, 1986.

Young, Christina M., "Fuzzy Controller for a Pitot-Static Test Set," National Aerospace and Electronics Conference, IEEE Service Center, Piscataway, NJ, 1993.

Ziemacki, Mike, "Fuzzy Logic Microcontroller," ECN, Volume 37, No. 3, March 1993.

## GLOSSARY

A PRIORI. Pertaining to deductive reasoning from assumed axioms or supposedly self-evident principles, without reference to experience.

ADAPTIVE FUZZY SYSTEMS. Fuzzy controllers with the ability to learn and explain their reasoning.

ALGORITHM. A set of well-defined rules for the solution of a problem in a finite number of steps.

ANALOG. Pertaining to devices, data, circuits, or systems that operate with variables that are presented by continuously measured voltages or other quantities.

ARC. The part of a semantic network that connects one object to another.

ARTIFICIAL INTELLIGENCE. The subfield of computer science concerned with emulating human intelligence by use of software and hardware techniques.

ASSISTANT. An Expert System that performs a technically limited subset of an expert's task.

ASSOCIATIVE MEMORY NETWORK. A network where different input patterns, if sufficiently similar, become associated with one another (trigger the same response).

ASYNCHRONOUS. Operating at a speed determined by the circuit functions rather than by timing signals.

ATTRACTOR. The geometrical pattern toward which the trajectory of a dynamical system, represented by a curve in phase space, converges in the course of time.

AXON. The pathway that carries the output of a biological neuron.

BACK PROPAGATION. An algorithm used to adjust weights connecting neurons in successive layers of multi-layer networks. Back propagation learns by example and repetition.

BACKWARD CHAINING. A method of problem solving that works backward from a known conclusion in an attempt to find a path of reasoning in order to justify the conclusion.

BASIN OF ATTRACTION. The collection of all possible initial conditions of a dynamical system for which the trajectories representing that system in phase space will converge to a particular attractor.

BINARY. Composed of or characterized by two parts or elements.

BLACK BOX. A component having known input and output, that can be readily inserted into or removed from a specific place in a larger system without knowledge of the component's detailed internal structure.

BLACKBOARD. The controlling agent in a blackboard system that tracks activity between the knowledge sources.

BLACKBOARD SYSTEM. An Expert System that uses multiple knowledge sources that share data.

BOOLEAN LOGIC. The basic mathematics needed to study the logic design of digital systems. Boolean logic consists of two values and three basic operators.

BREADTH FIRST. The method of searching for solutions by considering all possible subgoals on one level before proceeding to the next level.

CERTAINTY FACTOR. A value that represents the level of belief associated with a fact or a rule.

CERTIFICATION. The process of obtaining FAA approval for the design, manufacture, and/or sale of aircraft and associated systems, subsystems, and parts.

CLOSED-LOOP. A family of automatic control units linked together with a process to form an endless chain; the effects of control action are constantly measured so that if the controlled quantity departs from the norm, the control units act to bring it back.

CLUSTERING. A technique to group similar objects together under the same label.

CODIFYING. A method of representing knowledge or data so they can be stored in a knowledge base or database for access by a computer.

COLLEAGUE. An Expert System that performs a significant subset of an expert's task.

CONNECTION WEIGHT. A method of representing the strength of the connection between nodes.

CONVENTIONAL PROGRAMMING. The use of standard programming languages, as opposed to application development languages, financial planning languages, query languages, and report programs.

CONVERGENCE. The point when the outputs of a Neural Network are no longer changing.

CRISP SET. The membership concept used in traditional system modelling, where objects are either in or out of the set with no intermediate step.

DATA DRIVEN. Describing the execution of a program in a data flow system, in which an instruction is carried out whenever all its input values are present.

DECLARATIVE. Generally, the knowledge found in the knowledge base; independent knowledge that does not contain procedures or methods of handling other knowledge.

DEDUCTIVE. Learning by logical inference or inferring from available knowledge.

DEFUZZIFICATION. The process of translating fuzzy, or non-crisp values into crisp, clearly defined values.

DEMON. A procedure activated by changing or accessing values in a database.

DENDRITE. The pathway that carries the input of a biological neuron.

DEPTH FIRST. A method of searching for a solution by pursuing each branch of the search tree to the end before considering another branch.

DOT PRODUCT. The inner product of vectors $(x_1, \ldots, x_n)$ and $(y_1, \ldots, y_n)$ from n-dimensional euclidean space is the sum of $s_i y_i$, as i ranges from 1 to n.

EMPIRICAL MODEL. The model of a system based on data and the principles of statistics. Empirical models produce relationships, not insight.

EVENT. A change to the blackboard of a Blackboard System.

EXPERT. 1. An Expert System that approaches an expert's level of performance within a given domain. 2. A person who has mastered solving specific types of problems.

EXPERT SYSTEM. A computer system designed to simulate the problem solving behavior of a human expert.

EXPERT SYSTEM SHELL. An Expert System development tool and run-time environment with a modifiable knowledge base.

FACETTE. A piece of knowledge in a frame.

FEED FORWARD NEURAL NETWORK. A Neural Network where the only connections are from a neuron in one layer to a neuron in the next layer.

FILTER. A transmission network used in electrical systems for the selective enhancement of a given class of input signals.

FORWARD CHAINING. A method of solving problems by beginning with certain data and moving down the inference chain until a solution is reached.

FRAME. A way of representing Expert System knowledge that consists of a set of slots that contain data.

FRAME LABEL. A unique frame identification.

FUZZIFICATION. The process of converting crisp inputs into fuzzy values.

FUZZY CONTROL VARIABLE. A parameter whose value determines the action taken by fuzzy rules.

FUZZY EXPERT SYSTEM. An Expert System that uses fuzzy logic techniques to solve problems.

FUZZY LOGIC. The theory of representing vague or imprecise concepts to model inherent conditions.

FUZZY SET. An extension of the concept of a set, in which the characteristic function which determines membership of an object in the set can take on any value between 0 and 1.

FUZZY TERMS. Words used for descriptions that may not be precise in determining set memberships.

GRACEFUL DEGRADATION. A programming technique to prevent catastrophic system failure by allowing the machine to operate, though in a degraded mode, despite failure or malfunction of several integral units or subsystems.

HETEROASSOCIATIVE. A Neural Network where output patterns are distinct from input patterns.

HEURISTIC. Any rule of thumb, strategy, or technique used to limit the time required to search for solutions in large problem spaces.

HEURISTIC REASONING. Using an expert's rules of thumb, in the absence of precise control mechanisms, to reduce the space that must be searched for a solution.

HIDDEN LAYER. The middle layer in a Neural Network which takes outputs from the input layer, processes them, and passes them to the output layer.

INDUCTIVE. Learning by information repetition.

INFERENCE CHAIN. The path of reasoning that the inference engine follows to find a solution to the problem.

INFERENCE ENGINE. The part of an Artificial Intelligence system that uses knowledge in the knowledge base and acquired knowledge about the problem to form an expert solution.

INFERENCE MECHANISM. Controls the use of the knowledge base and databases when solving a problem.

INPUT LAYER. The first layer of neurons in a Neural Network. The input layer accepts the input values and passes them to the hidden layer. The input layer does not process data.

KNOWLEDGE ACQUISITION. The process of gathering the required information from various sources by a knowledge engineer.

KNOWLEDGE BASE. A form of coded knowledge about a specific domain.

KNOWLEDGE-BASED SYSTEM. A computer system whose usefulness derives primarily from a database containing human knowledge in a computerized format.

KNOWLEDGE ENGINEER. The person responsible for extracting, organizing, and encoding the knowledge related to the problem, and importing it into a knowledge base or database.

LABEL. A data item that serves to identify a data record, or a symbolic name used in a program to mark the location of a particular instruction or routine.

LAYER. A row of neurons in a Neural Network; there are usually three layers in a network.

LOGICAL SUM. The final unified result of the rule processes of a fuzzy system.

MACHINE LEARNING. The process or technique by which a device modifies its own behavior as the result of its past experience and performance.

META-KNOWLEDGE. Knowledge that reveals details about the system knowledge.

MODEL. A mathematical or physical system, obeying certain specified conditions, whose behavior is used to understand a physical, biological, or social system to which it is analogous in some way.

NEURAL NETWORK. A system modelled after the brain, used to solve problems by a mapping of input data to output data.

NEURO-FUZZY. Integrating a Neural Network and fuzzy logic to solve problems.

NEURON. The computational element in a Neural Network. A neuron computes the weighted sum of the inputs from other neurons and produces an output.

NODE. 1. Representation of an object that is related to other objects via connection weights. 2. The part of a Neural Network that calculates and processes information.

NOISE. Meaningless or erroneous bits that must be ignored or removed from a signal, especially in communication channels.

NON REAL-TIME LEARNING. Learning done in a Neural Network before attempting to solve application problems.

NONLINEARITY. A system in which the outputs do not correspond to the inputs in a direct or inversely proportional relationship.

OBJECT-ORIENTED. A representation form for knowledge in which all properties of an object are associated by the outgoing and incoming arcs at its node; the representation is geared towards manipulating objects as independent pieces of knowledge.

OPPORTUNISTIC REASONING. The process of solving problems in a way that is not uniform or predictable. Opportunistic reasoning attempts to solve the problem in the best possible way for each individual situation. Rules fire when they are appropriate, without a priori ordering.

OUTPUT LAYER. The layer of neurons in a Neural Network that takes the outputs from the hidden layer, processes them, and then produces the output of the network.

PARALLEL PROCESSING. Performing many different calculations simultaneously to reach one conclusion or solution.

PREDICATE CALCULUS. A formal language of classical logic that uses functions and predicates to describe relations between individual entities.

PROBABILITY. A theory dealing with the uncertainty that results from random behavior. Probability is defined over the numeric range of 0 to 1.

PROCESSING ELEMENT. The part of the Neural Network where the computations are performed.

PRODUCTION RULES. A common technique for representing procedural knowledge in an Artificial Intelligence system.

REAL-TIME LEARNING. Learning done in a Neural Network while solving an application problem.

RECURSION. A technique in which an apparently circular process is used to perform an iterative process.

RULE-BASED. An Artificial Intelligence program where knowledge is represented as rules.

RULES. A method of representing knowledge in the form of IF THEN statements.

SCHEDULER. A part of an inference engine that determines which knowledge source should be activated, and in what order.

SEARCH TREE. The branches of possible solutions to a problem that start with initial information and spread out following different paths.

SELF-SUPERVISED. A method of training a Neural Network without the use of an external monitor. A feedback device that detects errors and adjusts the connection weights accordingly is used.

SEMANTIC NETWORK. A method of representing knowledge using nodes and arcs.

SLOTS. Subdivisions making up the frame that may contain data, procedures, or pointers to other frames.

SPURIOUS STATE. A point in a Neural Network that acts as an attractor, but is not one of the stored patterns.

STABILITY. The property of a system which remains under control and responds in a reasonable manner to an applied input.

SUPERVISED LEARNING. A method of training a Neural Network in which an external supervisor monitors the activities, and learning occurs on the basis of direct comparison of the output of the network with known correct answers.

SYMBOLIC PROCESSING. A method of processing knowledge where the relationships among the knowledge are stored using symbolic representations; thus, the system can deal freely with objects and not be concerned with their composition.

SYNAPSES. Pathways in a biological neuron that transmit pulses from one neuron to another.

SYNCHRONOUS. In step, or in phase, as applied to two or more circuits, devices, or machines.

TRADITIONAL PROGRAMMING. The use of standard programming languages, as opposed to application development languages, financial planning languages, query languages, and report programs.

TRAINING. A change in connection weight values of a Neural Network that results in capturing information that can be recalled later.

TRANSFER FUNCTION. The mathematical relationship between the output of a control system and its input: for a linear system, it is the Laplace transform of the output divided by the Laplace transform of the input under conditions of zero initial-energy storage.

TRANSPARENCY. A characteristic of knowledge in an Expert System referring to its independence from other knowledge. There is no processing information in the transparent knowledge.

UNSUPERVISED LEARNING. A method of training a Neural Network, where no external monitor is involved; the Neural Network organizes itself by grouping and generating its own classification of inputs.

VERY LARGE SCALE INTEGRATION. Describing Integrated Circuits with more than 1000 elements.

WEIGHT MATRIX. The collection of connection weights for an entire Neural Network.

## ACRONYMS AND ABBREVIATIONS

| | |
|---|---|
| μC | micro Controller |
| μm | micrometer |
| AC | Advisory Circular |
| ACO | Aircraft Certification Office |
| AGARD | Advisory Group for Aerospace Research and Development |
| AI | Artificial Intelligence |
| AIAA | American Institute of Aeronautics and Astronautics |
| AIM | Airmen's Information Manual |
| ANSI | American National Standards Institute |
| ARINC | Aeronautical Radio, Incorporated |
| ARP | Aerospace Recommended Practice |
| ARPA | Advanced Research Projects Agency (formerly DARPA) |
| ASIC | Application Specific Integrated Circuit |
| ATC | Air Traffic Control |
| ATE | Automatic Test Equipment |
| BABBAGE | Boeing Advanced BlackBoard Ada Generation Environment |
| BIT | Built-In Test |
| CASSY | Cockpit Assistant System |
| CDU | Control Display Unit |
| CE | Certification Engineer |
| CF | Certainty Factor |
| CM | Configuration Management |
| CMOS | Complimentary Metal-Oxide Semiconductor |
| CNI | Communication, Navigation, and Identification |
| CPU | Central Processing Unit |
| DOD | Department of Defense |
| DOS | Disk Operating System |
| DOT | Department of Transportation |
| DSP | Digital Signal Processor |
| EEPROM | Electrically Erasable Programmable Read Only Memory |
| EPES | Emergency Procedures Expert System |
| ES | Expert System |
| ETANN | Electrically Trainable Analog Neural Network |
| F/FA | Fault and Failure Analysis |
| FAA | Federal Aviation Administration |
| FAR | Federal Aviation Regulation |
| FCA | Fuzzy Computational Accelerator |
| FIDE | Fuzzy Inference Development Environment |
| FIL | Fuzzy Inference Language |
| FINDER | Flight-Plan Interactive Negotiation and Decision-Aiding System for Enroute Rerouting |
| FLEX | FORTRAN Library for EXpert systems |
| GaAs | gallium arsenide |
| GUI | Graphical User Interface |

| | |
|---|---|
| HOL | High-Order Language |
| IC | Integrated Circuit |
| IEEE | Institute of Electrical and Electronics Engineers |
| IETM | Interactive Electronic Technical Manual |
| IJCNN | International Joint Conference on Neural Networks |
| I/O | Input/Output |
| IV&V | Independent Verification and Validation |
| k | thousand |
| KBAP | Knowledge-Based AutoPilot |
| LED | Light Emitting Diode |
| LISP | LISt Processor |
| MEP | Mission Equipment Package |
| MHz | MegaHertz |
| mm | millimeter |
| NASA | National Aeronautics and Space Administration |
| NN | Neural Network |
| ONR | Office of Naval Research |
| PA | Pilot's Associate |
| PC | personal computer |
| PE | Processing Element |
| PGA | pin grid array |
| PMA | Portable Maintenance Aid |
| PROLOG | PROgramming in LOGic |
| RAE | Royal Aerospace Establishment |
| RISC | Reduced Instruction Set Computer |
| RPA | Rotorcraft Pilot's Associate |
| RTCA | Requirements and Technical Concepts for Aviation (formerly Radio Technical Commission for Aeronautics) |
| SAE | Engineering Society for Advancing Mobility Land Sea Air and Space (formerly Society of Automotive Engineers) |
| SC | Special Committee |
| STC | Supplemental Type Certificate |
| TC | Type Certificate |
| TEAMS | Technical Expert Aircraft Maintenance System |
| TSO | Technical Standing Order |
| TTL | Transistor-Transistor Logic |
| USAF | United States Air Force |
| UUT | Unit Under Test |
| V&V | Verification and Validation |
| VLSI | Very Large Scale Integration |

INDEX


Advanced Research Projects Agency 20-49, 20-63
Advisory Circular 20-71
Air Traffic Control 20-2, 20-60, 20-61
Aircraft Certification Office 20-70, 20-71
Airmen's Information Manual 20-61, 20-79
Application Specific Integrated Circuit 20-51, 20-52
Applications 20-1, 20-19, 20-25, 20-35, 20-41, 20-57
Arc 20-15
ARINC 20-70
Artificial Intelligence
  definition 20-1
  development systems and tools 20-52
  history 20-5
  integrated systems 20-41
  software 20-45
Associative network 20-15
Automatic Test Equipment 20-65
Axon 20-38
Back propagation 20-37
Backward Chaining 20-18
Black box testing 20-78, 20-85, 20-88, 20-89
Blackboard System 20-23
  control 20-25
  knowledge source 20-25
Boeing Advanced BlackBoard Ada Generation Environment 20-48
Built-In Test 20-65
Certainty Factor 20-15
Certification 20-69
  guidelines 20-70
Certification specialist 20-79, 20-93, 20-98, 20-99
Chaining 20-17
CLIPS 20-48
Cockpit Assistant System 20-2, 20-59
Communication, Navigation, and Identification 20-66
Completeness 20-76
Complimentary Metal-Oxide Semiconductor 20-49, 20-50, 20-52
Configuration Management 20-90
Consistency 20-76
Control Display Unit 20-60
correctness 20-75, 20-80
  functional 20-75
  resource consumption 20-76
  safety 20-76
  user-interface 20-76
  utility 20-76
Crisp Logic 20-28
Database 20-4, 20-7
Decision support system 20-59
Dendrite 20-38
Department of Defense 20-2, 20-94
Diagnostic System 20-65
Digital Signal Processor 20-54
Diverter 20-60
DOD-STD-2167A 20-73
Electrically Trainable Analog Neural Network 20-49
Emergency Procedures Expert System 20-58, 20-59
Expert System 20-7
  architecture 20-7
  classes 20-8
  definition 20-6
  fuzzy 20-43
  implementation 20-8
  shell 20-81
  testability issues 20-92
  Tools 20-46
  verification and validation 20-78
  verification and validation techniques 20-85
Explanation of Reasoning 20-9
Facette 20-17
Fault and Failure Analysis 20-70
Federal Aviation Administration 20-49, 20-69, 20-70, 20-71, 20-94
Federal Aviation Regulations 20-61, 20-70, 20-79
Flight Management Expert System 20-58

Flight-Plan Interactive Negotiation and Decision-Aiding System 20-61
formal verification 20-88
FORTRAN Library for EXpert systems 20-48, 20-59
Forward Chaining 20-18
Frames 20-15
fuzzy control
defuzzification 20-31
fuzzification 20-29
inference mechanism 20-29
Fuzzy Inference Development Environment 20-54
Fuzzy Inference Language 20-54
Fuzzy Logic 20-26
Fuzzy set 20-31, 20-34, 20-43
Heuristic Reasoning 20-20
Hidden layer 20-40
High-Order Language 20-83, 20-84, 20-90
human factors 20-93, 20-100
automation 20-96
Independent Verification and Validation 20-74, 20-78
Inference chain 20-14, 20-18
Inference Engine 20-7
Interactive Electronic Technical Manual 20-65
Intuition 20-1, 20-5, 20-35, 20-84, 20-97
Knowledge Acquisition 20-18
Knowledge Base 20-7
updating 20-17
Knowledge Representation 20-9
Knowledge-Based AutoPilot 20-59
LISP 20-6, 20-45, 20-46, 20-48, 20-58, 20-80
Machine Intelligence 20-2
Machine learning 20-17, 20-54
Meta-knowledge 20-11, 20-18
National Aeronautics and Space Administration 20-2, 20-48, 20-59, 20-60, 20-94
Natural language 20-4, 20-9, 20-26, 20-46
Navigation System 20-59
Neural Accelerator 20-51
Neural Network 20-36, 20-37, 20-38, 20-40, 20-41, 20-42, 20-49, 20-66, 20-97
connections 20-39
construction 20-38
convergence 20-38
definition 20-36
layers 20-40
learning 20-37
transfer functions 20-38
Neural Processor 20-49
Office of Naval Research 20-66
Opportunistic reasoning 20-25
Pilot's Associate 20-63, 20-79, 20-80, 20-100
Processing Element 20-38, 20-39
Production rule 20-24, 20-54
PROLOG 20-6, 20-45, 20-46
Reduced Instruction Set Computer 20-57, 20-58
reliability 20-91
Robotics 20-3
Rotorcraft Pilot's Associate 20-63, 20-65
Rotorcraft transmission health monitoring system 20-65
Royal Aerospace Establishment 20-48, 20-59
RTCA 20-70
Special Committee 20-71
RTCA/DO-178B 20-72
Rule 20-6, 20-7
Rule Architecture 20-14
Rule-Based System 20-10
SAE 20-70
Scheduler 20-23, 20-24, 20-25
Search tree 20-20
Searching 20-20
breadth first 20-20, 20-21
depth first 20-21
Semantic Networks 20-15
Slot 20-15
software development cycle 20-79
Special Committee 20-71
Supervised Learning 20-37
Supplemental Type Certificate 20-69
supportability 20-90
Synapse 20-38
Technical Expert Aircraft Maintenance System 20-66, 20-67/68
Technical Standing Order 20-69
Termination 20-77
test bias elimination 20-87
testing
coverage 20-88
data consistency 20-87
implementation 20-85
knowledge acquisition 20-88

knowledge consistency 20-88
knowledge representation 20-88
rule consistency 20-86
sensitivity analysis 20-87
specification-directed analysis 20-87
structural 20-87
Transistor-Transistor Logic 20-49, 20-50
Type Certificate 20-69
United States Air Force 20-63
Unsupervised Learning 20-38
validation
definition 20-74
verification
automating 20-89
definition 20-73
documentation 20-74
Verification and Validation 20-77
conventional 20-73
dynamic testing 20-77
Expert System 20-78
general testing methods 20-78
inference engine 20-82
static testing 20-77
techniques for Expert Systems 20-85
VLSI 20-52
Weight matrix 20-39

DOT/FAA
CT-88/10
V2,chp20

Digital systems validation handbook : volume 2, chapter 20 artificial intelligence with applications for aircraft.

# GLOSSARY
# and
# ACRONYMS

NOTICE

This document is disseminated under the sponsorship of the U.S. Department of Transportation in the interest of information exchange. The United States Government assumes no liability for the contents or use thereof.

The United States Government does not endorse products or manufacturers. Trade or manufacturers' names appear herein solely because they are considered essential to the objective of this report.

# GLOSSARY

α-FAULT. (10) A fault activated by the baseline program (see ß-FAULT).

A-SPECIFICATION. (9) The highest level specification typically produced by the contracting organization to define a system (see MIL-STD-1521).

ABSORPTION LOSS. (11) Attenuation or retention of electromagnetic energy passing through a material, a shield. Absorption loss and reflection loss contribute to total shielding effectiveness (SE).

ACCESS. (18) The process of a transmitting bus user obtaining control of a data bus in order to transmit a message.

ACTION INTEGRAL. (13) The action integral is a critical factor in the production of damage. It relates to the energy deposited or absorbed in a system. This energy cannot be defined without knowing the resistance of the system. The instantaneous power dissipated in a resistor is $I^2R$ and is expressed in watts. For the total energy expended, the power must be integrated over time to get the total joules, watt-seconds. By specifying the integral of $i(t)^2$ over the time interval involved, a useful quantity is defined for application to any resistance value. In the case of lightning, this quantity is defined as the action integral and is specified as $i(t)^2dt$ over the time the current flows.

ACTIVE FAULT. (10) A fault that can produce an error (for some input) while executing the current program.

ACTUAL TRANSIENT LEVEL. (13) The actual transient level is the level of transients which actually appear at the system interfaces as a result of the external environment. This level may be less than or equal to the transient control level but should not be greater.

ADDRESSING CAPACITY. (6) The number of components addressable by the protocol used on a given data bus.

ADVISORY CIRCULAR. (18) An external FAA publication consisting of nonregulatory material of a policy, guidance, and informational nature.

AIR TRANSPORT AIRCRAFT. (18) Aircraft used in interstate, overseas, or foreign air transportation.

AIRCRAFT LIGHTNING INTERACTION. (13) An encounter with lightning that produces sufficient current within or voltages along an aircraft skin or structure to pose a threat to the aircraft electrical/electronic systems, as a result of a direct lightning attachment.

AIRWORTHINESS STANDARDS. (18) Parts 23, 25, 27, 29, and 33 of the Code of Federal Regulations, Title 14, Chapter 1, Subchapter C.

AMBIENT. (16) The substance which absorbs heat from the heat sink.

ANALYTICAL REDUNDANCY. (7) The use of software algorithms which use known mathematical relationships between different sensors for sensor failure detection and replace most of additional redundant sensor hardware.

ANALYTICAL ROOT SOLUTION. (4) Information obtained from the roots of the characteristic equations of the airplane model such as short-period or phugoid frequency response.

ANGLE OF ATTACK. (4) Angle between the longitudinal axis of an aircraft and the direction of movement.

ANODIZE. (11) A preparation by electrolytic process that deposits a protective oxide, insulating film on a metallic surface (aluminum). The oxide defeats electrical bonding. Alodine and iridite finishes on aluminum are conductive.

ANTAGONISTIC QUALITY FACTORS. (17) Quality Factors with conflicting attributes.

APERTURE. (11) An opening, such as a nonconductive panel joint, slot, or crack, allowing electromagnetic energy to pass through a shield.

ARCHITECTURE. (18) The design and interaction of components of a computer system.

ARTIFICIAL INTELLIGENCE. (16) The characteristics of a machine programmed to imitate human intelligence functions.

ASSURANCE ASSESSMENT. (4) Procedures whose purpose is to ensure that a proposed system functions according to design specifications.

ASYNCHRONOUS MESSAGES. (6) Electronic signals with transmission times that are not known a priori. These may include priority signals requiring immediate access to the bus.

ATTACHMENT POINT. (13) A point of contact of the lightning flash with the aircraft.

AUDIO FREQUENCY (AF). (11) The spectrum (20 to 20,000 Hz) of human hearing, often defined as extending from approximately 20 Hz to 50 kHz and sometimes to 150 kHz. Audio noise is nuisance hum, static, or tones from power line 400 Hz, switching regulator and digital clock harmonics, or HF, VHF transmitter frequencies.

AUTOFEATHER. (16) To automatically and swiftly feather the propeller when the engine fails to drive it.

AUXILIARY PROGRAMS. (10) Software executed occasionally.

AVALANCHING LATENT FAULTS. (10) The successive activation of latent faults.

AVIONIC. (18) Electronic equipment used in aircraft.

ß-FAULT. (10) A fault not activated by the baseline program (see α-FAULT).

BABBLING TRANSMITTER. (18) A bus user that transmits outside its allocated time.

BACKSHELL. (11) Metal shell connecting circuit shields or overbraid to an electrical connector.

BACKWARD RECOVERY. (9) Restoration of the system to some previous known correct state and restarting the computation from that point.

BALANCED CIRCUIT. (11) A signal, acting line-to-line, between two conductors having symmetrical voltages identical and equal in relation to other circuits and to ground. "Differential mode" is line-to-line; "common mode" is line to ground.

BALANCED CONFIGURATION. (18) A bus using the HDLC protocol that connects only primary stations.

BANDWIDTH (BW). (11) Frequencies bounded by an upper and lower limit in a given band associated with electronic devices, filters, and receivers.

BASELINE PROGRAM. (10) A set of continuously executed software modules.

BENIGN FAULT. (10) A fault that cannot produce an error while executing the current program, regardless of input, but may produce an error for some other program.

BIDIRECTIONAL DATA BUS. (18) A data bus with more than one user capable of transmitting.

BINARY SEARCH. (17) A searching algorithm in which the search population is repeatedly divided into two equal or nearly equal sections.

BIT-ORIENTED PROTOCOL. (18) A communication protocol where message frames can vary in length, with single bit resolution.

BIT TIME. (6) The time it would take to transmit one bit. Usually this is "blank" time when nothing is being transmitted. One nth of the bus speed (i.e., on a 1 kHz bus, the bit time is $10^{-3}$ seconds).

BITS. (17) Binary digits.

BLOCK TRANSFER. (6) A data transfer mode allowing the transfer of variable length data blocks.

BOND, ELECTRICAL. (11) Electrical connection at two metallic surfaces securely joined to assure good conductivity often 2.5-mΩ maximum for electrical/elec-

tronic units and 1Ω for electrostatic dissipation or safety. A "faying surface" bond maintains contact between relatively large or long surfaces. Inherently bonded parts are permanently assembled and conductivity exists without special preparation: such as with welding, brazing.

BRAID, OVERBRAID. (11) Fine metallic conductors woven to form a flexible conduit or cableway and installed around insulated wires to provide protection against electric fields and radio frequencies. Best when peripherally connected to backshells. A grounding strap/jumper may be made of braid.

BRIDGE. (18) A BIU that is connected to more than one bus for the purpose of transferring bus messages from one bus to another, where all the buses follow the same protocol.

BROADBAND. (12) A frequency spectrum which is wide compared to the bandwidth of the device used to detect it.

BROADCAST. (4) Transmission of messages to all terminals without reference to the identification of the receiving station or terminal.

BROADCAST CAPABILITY. (6) The capacity to transmit messages to all terminals simultaneously.

BROADCAST DATA BUS. (18) A data bus where all messages are transmitted to all bus users.

BUFFER. (18) Memory used to hold segments of the data transferred between asynchronous processes.

BUS. (18) A conductor that serves as a common connection of a signal to multiple users.

BUS CONTROLLER. (18) The electronic unit that is designed to control the bus communication of all users for a centrally controlled bus.

BUS INTERFACE UNIT. (18) The electronics that interface the host CPU of an LRU to a bus medium.

BUS MESSAGE. (18) A complete set of bits that can be transferred between two bus users.

BUS NETWORK. (18) The collection of all BIUs and bus media associated with one bus.

BUS OVERLOAD. (18) The condition that exists when the time it takes to transmit outstanding messages on a bus exceeds the time allotted for those transmissions.

BUS USER. (18) Any LRU attached to a bus.

BYZANTINE RESILIENCE. (5) A fault tolerant process which is tolerant of intermittent faults that can send good information part of the time.

CABLE OR HARNESS. (11) A bundle of separate, insulated, electrical circuits, shielded or unshielded, usually long and flexible and having breakouts, terminations, overbraid, and mounting provisions completely assembled.

CABLEWAY. (11) A solid metallic housing (liner, foil, coating) surrounding and shielding insulated electrical conductors. Also called conduit, tray, or raceway. Crosswise or transverse openings or breaks in the metallic cableway cause noise voltages to be transferred to internal wire circuits.

CANARD. (16) A tail-first aerodyne, usually with auxiliary horizontal surface at the front and a vertical surface at the back.

CAT IIIa LANDING. (6) One of several landing categories defined in FAR 91. CAT IIIa implies the need for an instrument landing approach.

CENTRAL BUS CONTROL. (18) The bus control approach where a single electronic unit attached to a bus controls all the communication of the bus users.

CENTRAL CONTROL. (6) Control from one master, whether stationary or non-stationary.

CERTIFICATION. (18) The process of obtaining FAA approval for the design, manufacture, and/or sale of aircraft and associated systems, subsystems, and parts.

CHARACTER-ORIENTED PROTOCOL. (18) A communication protocol where messages can vary in length, with single character resolution.

CHARGE TRANSFER. (13) The integral of the current over its entire duration $\int i(t)dt$, in coulombs.

CHECKSUM. (18) An error detection code produced by performing a binary addition, without carry, of all the words in a message.

CHORD. (4) The straight line segment intersecting or touching an airfoil profile at two points.

CLOSED-LOOP. (18) A system where the output is a function of the input and the system's previous output.

CODE. (17) The subset of software which exists for the sole purpose of being loaded into a computer to control it.

COMMAND/RESPONSE. (6) "Operation of a data bus system such that remote terminals receive and transmit data only when commanded to do so by the controller." (MIL-STD-1553 Designer's Guide, 1983, p. II-3.)

COMMAND/RESPONSE DATA BUS. (18) A data bus whose protocol initiates each data transfer with a command and terminates the transfer after a proper response is received.

COMMON MODE IMPEDANCE. (11) Impedance or resistance shared by two or more circuits so that noise voltages/currents generated by one are impressed on the others.

COMMON MODE REJECTION. (11) The ability of wiring or an electronic device to reject common mode (line-to-ground) signals and maintain fidelity of differential mode (line-to-line) signals.

COMMON MODE SIGNAL. (11) Identical and equal signals on input conductors or at the terminals of a device relative to ground.

COMPLEMENTARY QUALITY FACTORS. (17) Quality Factors with interrelated attributes.

COMPONENT DAMAGE. (13) Condition arising when the electrical characteristics of a circuit component are permanently altered beyond its specifications.

CONDUCTED EMISSION (CE) OR INTERFERENCE. (11) Voltage/current noise signals entering or leaving a unit on interface conductors. Emission is the general term, interference is undesired noise.

CONFIGURATION MANAGEMENT. (18) The precise control and documentation of the configuration of an entity at any time during its development and deployment.

CONTENT ADDRESSING. (6) The system of identifying message recipients based on information embedded in the message. This is in contrast to destination terminal addresses.

CONTENTION PROTOCOL. (18) A protocol that allows users to randomly access the bus at any time. When bus contention results, each user tries again to access the bus without contention.

CONTROL LAW. (7) The physical relationship between various sensors and control surfaces.

CONTROL REGISTER. (18) A register in an IC controller that receives commands from a host processor.

CONTROL STRUCTURES. (17) Programming constructs which direct the flow of control.

CORONA. (13) A luminous discharge that occurs as a result of an electrical potential difference between the aircraft and the surrounding atmosphere.

COUPLING. (11) The transfer of energy between wires or components of a circuit electrostatically, electromagnetically, or directly.

COVERAGE. (5) The conditional probability of the system successfully recovering from a component fault and continuing to perform the intended functions correctly, given the presence of the fault. Coverage is the measure of effectiveness of a system's utilization of redundant hardware. Coverage can be qualified and applied to many different components of a system and phases of

recovery process. Examples include, fault detection coverage, fault isolation coverage, latent fault coverage, sensor failure coverage, and memory failure coverage.

COVERAGE. (7) The percent confidence level of a given analytical redundancy fault detection and isolation algorithm for all types of faults.

COVERAGE. (9) The probability that when a fault occurs, it will be detected and recovery from the fault will be successful.

CRITICAL. (13) Functions whose failure would contribute to or cause a failure condition which would prevent the continued safe flight and landing of the aircraft.

CROSS COUPLING (CROSSTALK). (11) Transfer of signals from one channel, circuit, or conductor to another as an undesired or nuisance signal or the resulting noise.

DAMAGE. (11) The irreversible failure of a component.

DATA BUS. (6,18) A system for transferring data between discrete pieces of equipment in the same complex.

DATA BUS PROTOCOL. (18) The set of rules that governs the transfer of data between data bus users.

DATA LATENCY. (6,18) The delay from the time when a piece of information becomes available at a source terminal to the time it is received at the destination.

DATA LINK ASSURANCE OF RECEIPT. (6) The guarantee of good data through the data link level.

DATA REASONABLENESS CHECK. (18) A check performed to see if a value of data is within reasonable bounds for the given context.

dBμV. (12) Decibels referred to one microvolt. Zero db represents one microvolt.

DECIBEL (dB). (11,12) Decibel expresses the ratio between two amounts of power, P1 and P2, at two separate points in a circuit. By definition, the number of dB = 10 log to the base 10 of (P1/P2). For special cases, when a standard power level P2 = 1 mW or 1 W or 1 kW, then the ratio is defined as "dBm," "dBw," or "dBKW." Because $P = V^2/R$ and also $I^2R$, decibels express voltage and current ratios. Ideally, the voltages and currents are measured at two points having identical impedances. By definition, dB = 20 log V1/V2 and dB = log I1/I2. For convenience, V2 or I2 are often chosen as 1 μV or 1 μA and the ratio is defined as dB above a μV or dB above a μA when graphing emission or susceptibility limits.

DECOUPLED MANEUVERS. (4) Changes in an aircraft's direction and attitude in one axis without affecting direction or attitude in other axes.

DEFAULT DATA. (18) An alternative value used for a parameter whenever the normal data is not supplied.

DESIGN ERROR. (4) A functional flaw resulting from a misinterpretation of the specifications of the system.

DESIGN MARGIN. (13) The difference between the equipment transient design levels and the transient control level.

DETERMINISTIC. (6) A system where all parameters are known, as opposed to a statistical system where the outcome is subject to the laws of probability.

DETERMINISTIC PROTOCOL. (18) A protocol where all parameters are known so that its various states are predictable in sequence and time.

DIAGNOSTIC FILTER. (7) An analytical algorithm which processes data from N functionally related sensors. The data are used to estimate some sensor outputs and assess the correct functioning of the sensors.

DIELECTRIC STRENGTH. (11) Voltage withstand capability that an insulating material sustains before destructive arcing and current flow, usually expressed in volts per mil thickness. Dielectric withstand voltage is the voltage level at which insulation breakdown occurs.

DIFFERENTIAL MODE (DM) SIGNAL. (11) The signal in a two-wire circuit measured from line-to-line.

DIGITAL DATA BUS. (18) A data bus that uses digital electronic signals.

DIRECT EFFECTS. (13) Any physical damage to the aircraft or onboard systems due to the direct attachment of the lightning channel. This includes tearing, bending, burning, vaporization, or blasting of aircraft surfaces or structures, and damage to electrical/electronic systems.

DISSIMILAR REDUNDANCY. (18) The redundancy of systems that provide a redundancy of function, but by a different form.

DISSIMILAR SOFTWARE. (18) Redundant computer programs that provide a redundancy of function, but by a different form.

DISTRIBUTED BUS CONTROL. (18) The bus control approach where the total communication control job is distributed across the bus users, each controlling the communications during its period of responsibility.

DISTRIBUTED CONTROL. (6) Concurrent control from multiple points in the data bus system.

DOUBLE FAIL-OPERATIONAL SYSTEM. (4) A quadruplex (or higher) redundant flight-control system which is designed to incur failures in two redundant lanes (or channels) before it fails.

DUAL-DUAL ARCHITECTURE. (4) Two parallel dual computers with a voting plane at the output of each dual computing lane.

DUAL FAIL-OPERATIONAL. (7) A reliability requirement placed on a system which requires the system to be operational after two failures have occurred.

DUAL GROUND. (11) Equipment case ground/return through two independent circuit paths to structure implemented in flammable zones and water leakage areas-each path meeting electrical conductivity (resistance) requirements.

ELECTRIC FIELD. (11) High-impedance, radiated voltage field, positive or negative, from a voltage source as contrasted to a low-impedance magnetic field from a current source.

ELECTROMAGNETIC COMPATIBILITY (EMC). (11) Operation within performance specification in the intended electromagnetic interference environment.

ELECTROMAGNETIC INTERFERENCE (EMI). (11) Conducted and radiated voltage/current noise signals, broadband (BB) or narrow band (NB), that degrade the specified performance of equipment.

ELECTROMIGRATION. (5) Drifting of metal atoms toward the cathode of a cathode ray tube.

ELECTROSTATIC CHARGE. (11) Electric potential energy with a surrounding electric field, uniform or nonuniform, moving or at rest, on a material.

EMISSION. (11) Voltage/current noise on a wire or in space. Broadband emission has uniform spectral energy over a wide frequency range and can be identified by the response of a measuring receiver not varying when tuned over several receiver "bandwidths." Or, energy present over a bandwidth greater than the resolution bandwidth where individual spectral components cannot be resolved. Broadband (BB) may be of two types: (1) impulse and coherent varies 20 dB per decade of bandwidth and (2) random or statistical, varies 10 dB per decade. A narrow band (NB) emission or signal, sometimes called continuous wave, occurs at a discrete frequency and does not vary with bandwidth.

EMULATION. (18) The duplication of the behavior of a system with a different system.

ENVELOPE LIMITING. (4) General or additional limits imposed on the structural, "g" limits, speed, attitude, etc. of the aircraft. In some cases, envelope limiting imposes additional constraints on the envelope that cannot be exceeded regardless of pilot inputs.

EQUIPMENT TRANSIENT DESIGN LEVEL. (13) The level of transients which the equipment is qualified to withstand.

EQUIPMENT TRANSIENT SUSCEPTIBILITY LEVEL. (13) The transient level which will result in damage or upset to the system components. This level will be greater than the equipment transient design level.

EQUIVALENCE STATEMENT. (17) A FORTRAN statement which equates two variable names.

ERROR. (4) A mistake in specification, design, production, maintenance, or operation of a system causing undesirable performance.

ERROR. (8) A state of the system which (in the absence of any corrective action by the system) could lead to a failure that would not be attributed to any event subsequent to the error. (More accurately known as an erroneous state.)

ERROR MASKING. (18) The process of masking the presence of avionic errors, possibly by using an electronic voter to override an erroneous input with the values of substitute inputs.

EVENT, EXTREMELY IMPROBABLE. (4) An event with a probability of occurrence on the order of $10^{-9}$ or less.

EVENT, IMPROBABLE. (4) An event with a probability of occurrence on the order of $10^{-5}$ or less.

EVENT, PROBABLE. (4) An event with a probability of occurrence on the order of $10^{-5}$ or greater.

EXTERNAL ENVIRONMENT. (13) Characterization of the natural lightning environment with idealized waveforms for engineering purposes.

FAIL-OPERATIONAL. (7) A reliability requirement placed on a system which requires the system to be operational after a single failure has occurred.

FAIL-SAFE. (7) A reliability requirement placed on a system which requires that safe flight not be hindered even after a failure.

FAIL-SAFE. (18) A design philosophy that ensures that any failure in a system does not result in an unsafe condition after the failure.

FAILURE. (4) The inability of a system, subsystem, unit, or part to perform within specified limits.

FAILURE. (5) The deviation of system behavior from specifications (arithmetic failure, storage failure, flight control function failure.)

FAILURE. (8) The situation when the external behavior of a system does not conform to that prescribed by the system specification.

FAILURE, HARD. (5) Repeated use of the same input and initial conditions results in the same incorrect response.

FAILURE, HIDDEN. (4) A failure that is not manifested at the time of its occurrence.

FAILURE MECHANISM. (5) Any situation that could produce an error condition. Examples of failure mechanisms include metal migration, voltage overstress, and lack of air-conditioning.

FAILURE, PERMANENT. (5) Repeated use of the same input and initial conditions results in the same incorrect response.

FAILURE, SOFT. (5) Repeated use of the same input and initial conditions does not result in the same incorrect response.

FAILURE, TEMPORARY. (5) Repeated use of the same input and initial conditions does not result in the same incorrect response.

FAILURE, TRANSIENT. (5) Repeated use of the same input and initial conditions does not result in the same incorrect response.

FALL-TIME. (12) The time required for pulse amplitude to go from a predefined magnitude to a given level.

FALSE ALARM. (7) The declaration of a fault by a fault detection monitor or algorithm when there is no fault.

FAULT. (4) An error in the operation of a system.

FAULT. (5) The phenomenological reason for a failure (open wire, stuck-at fault, design fault, etc.). In general, any condition preventing a digital component from correctly changing state when directed to change by input parameters. For electrical components there is a one-to-one correspondence between faults and failures. The situation is not so simple with digital circuits. For if the circuit is S-A-1, any input causing a one output will be correctly processed; a little like the stopped clock that is correct twice per day. For a processor having a million or so logic gates, it is not possible to test for all the combinations of input and output states.

FAULT. (8) The adjusted cause of error.

FAULT AVOIDANCE. (9) The attempt to prevent any software faults in the final delivered product through disciplined software development practices, testing, and IV&V.

FAULT CONTAINMENT. (6,9) The capacity of a system to prohibit errors and/or failures from propagating from the source throughout the system.

FAULT CURRENT. (11) The maximum current (magnitude and duration) flowing through a fault point. This current is equal to the supply voltage divided by the dc resistance of power line leads, circuit breakers, and the current return in wire or structure.

FAULT DETECTION. (6) The capacity of a system to determine the occurrence of erroneous operation.

FAULT DETECTION. (7) The determination that a sensor is faulted by using a software algorithm.

FAULT, HARD. (4) A defect in the hardware or software of a digital control system that permanently affects some functional performance of the system.

FAULT INSERTION. (4) A testing technique used to obtain information about data latency and built-in test coverage of a digital flight-control system.

FAULT ISOLATION. (6) The capacity of a system to isolate a failure to the required level so it can reconfigure.

FAULT ISOLATION. (7) The determination that a particular sensor is faulted by using a software algorithm.

FAULT, LATENT. (5) A fault which has not yet caused a failure. (For example, a fault in a memory chip that is not being used for the foreground program or in this particular mode of the system is a latent fault.)

FAULT, SOFT. (4) A transient defect in the software of a digital flight-control system that can be overcome by error-correctable code or by recycling of power to the computer system.

FAULT, STUCK-AT. (5) A logic signal which remains at zero (S-A-0) or one (S-A-1).

FAULT TOLERANCE. (6,9) The capability to endure errors and/or failures without causing total system failure.

FAULT TOLERANCE. (7) Accommodation of sensor hardware faults based on some type of comparator scheme.

FAULT TOLERANCE. (18) The ability of a system to continue operation after a fault, possibly in a degraded condition.

FAULT TOLERANT. (4,9) Software which continues to operate satisfactorily in the presence of faults.

FAULT TOLERANT SYSTEM. (5) A system that continues to function although certain components may have faults.

FAULT TREE ANALYSIS. (4) A top-down deductive analysis that identifies the conditions and functional failures necessary to cause a defined failure condition. The fault tree can be used to establish the probability of the ultimate failure condition occurring as a function of the estimated probabilities of contributory events.

FEDERAL AVIATION REGULATIONS. (18) Subchapter C of the Code of Federal Regulations, Title 14, Chapter 1.

FILTER. (11) Device or unit that passes or rejects a frequency band and is designed to block noise from entering or leaving a circuit or unit.

FINITE STATE MACHINE. {18} A state machine with a finite number of states.

FLIGHT CODE. {4} The application software of the digital flight-control system.

FLIGHT-CRITICAL. {4,7,18} A description of functions whose failure would contribute to or cause a failure condition preventing the continued safe flight and landing of the aircraft.

FLIGHT-ESSENTIAL. {4,18} A description of functions whose failure would contribute to or cause a failure condition which would significantly affect the safety of the airplane or the ability of its crew to cope with adverse operating conditions.

FLIGHT-NONESSENTIAL FUNCTION. {18} A function whose failure could not significantly degrade aircraft capability or crew ability.

FLIGHT-PHASE CRITICAL. {4} A description of functions which are critical only during certain phases of flight.

FLY-BY-GLASS. {16} Flight control system where fiber optics carry the signal.

FLY-BY-LIGHT. {4,16} Flight control system where fiber optics carry the signal.

FLY-BY-WIRE. {4,16} Flight control system with electric signaling.

FORWARD RECOVERY. {9} Restoration of the system to a consistent state by compensating for inconsistencies found in the current state so that the system may continue processing.

FOURIER TRANSFORM. {12} A mathematical method for deriving the frequency spectrum from a time dependent function.

FRAME. {18} A formatted block of data words or bits that is used to construct messages.

FUNCTIONAL PARTITIONING. {18} The partitioning of system functions by placing each group of users, which share a common function, on different data buses.

GATEWAY. {18} A bus user that is connected to more than one bus for the purpose of transferring bus messages from one bus to another, where the buses do not follow the same protocol.

GENERAL AVIATION AIRCRAFT. {18} The non-air transport civil aircraft.

GENERATOR POLYNOMIAL. {18} The polynomial code that is used to generate the remainder in the division of the CRC check.

GIGABIT. {16} One billion bits.

GLASS COCKPIT. {9} Advanced state-of-the-art electronic displays utilizing flat panel and/or cathode ray tube display technology for cockpit instrumentation.

GLOBAL STATE. (18) A state that represents the condition of the entire network being modeled, including senders, receivers, and the communication link.

GROUND. (11) A generic term having multiple meanings and indicating a circuit return path or a voltage reference: not "zero" voltage reference. Four hundred millivolts of noise voltage is common on "quiet" grounds. There are several types of returns and references.

GROUND EFFECT. (4) Increase in aircraft lift when operating near the ground.

HALF-DUPLEX. (18) Bidirectional communication between two entities on a single channel by each having a turn to control the channel.

HAMMING CODE. (18) An error detection and correction code based on the Hamming distance.

HAMMING DISTANCE. (18) The number of bit positions in which two binary words differ.

HANDSHAKING. (18) The reciprocal responses given by two electronic systems to sequence the steps of a transfer of data between them.

HARD FAILURE. (12) A failure that requires a reset of the equipment.

HARDWARE. (17) The physical components of a computer.

HARDWARE-IN-THE-LOOP SIMULATION. (18) A partial simulation of a system; part of the actual system is used in the simulation.

HAZARD FUNCTION. (8) The conditional probability that a fault is exposed in the interval t to $\Delta t$ given that the fault did not occur prior to time t.

IMMUNITY. (11) Capability of a circuit or unit to operate within performance specification in a specified electromagnetic interference environment.

INDIRECT EFFECTS. (13) Voltage and/or current transients induced by lightning in aircraft electrical wiring which can produce upset and/or damage to components within electrical/electronic systems.

INDUCED VOLTAGES. (13) A voltage produced around a closed path or circuit by changing magnetic or electric fields or structural IR voltages.

INITIALIZATION. (6) Setting the beginning parameters and values on system power-up. For redundant systems this includes setting the initial configuration of the system.

INTERNAL ENVIRONMENT. (13) The fields and structural IR potentials produced by the external environment, along with the voltages and currents induced by them.

INTERRUPT VECTOR. (18) The address that points to the beginning of the service routine for an interrupt.

INTERRUPT VECTOR TABLE. (18) The table of interrupt vectors for all interrupts serviced by a system.

ISOLATION. (11) Electrical separation and insulation of circuits from ground and other circuits or arrangement of parts to provide protection and prevention of uncontrolled electrical contact.

JOULE. (12) A unit of energy equal to one watt-second.

JUMPER/STRAP. (11) A short wire, strip, strap, or braid conductor installed to make a safety ground connection, to dissipate electrostatic charge, or establish continuity around a break in a circuit.

KILOBYTE. (16) One thousand bytes.

LABELED ADDRESSING. (6) The system of identifying message recipients based on labels. This is in contrast to destination terminal addresses.

LATENT FAULT. (10) A fault which has not yet produced a malfunction. (In the context of the single-fault model, benign and latent faults are equivalent.)

LIGHTNING FLASH. (13) The total lightning event in which charge is transferred from one charge center to another. It may occur within a cloud, between clouds, or between a cloud and the ground. It can consist of one or more strokes, plus intermediate or continuing currents.

LIGHTNING LEADER STROKE. (13) The leader forms an ionized path for charge to be channeled towards the opposite charge center. The stepped leader travels in a series of short, luminous steps prior to the first return stroke. The dart leader reionizes the return stroke path in one luminous step prior to each subsequent return stroke in the lightning strike.

LIGHTNING RETURN STROKE. (13) A lightning current surge that occurs when the lightning leader makes contact with the ground or an opposite charge center.

LIGHTNING STRIKE. (13) Any attachment of the lightning flash to the aircraft.

LIGHTNING STRIKE ZONES. (13) Locations on the aircraft where the lightning flash will attach or where substantial amounts of electrical current may be conducted between attachment points. The location of these zones on any aircraft is dependent on the aircraft's geometry and operational factors and often varies from one aircraft to another.

LIMITING, VOLTAGE/CURRENT. (11) Semiconductor components, diodes, Transorb, or filter designed to clip and shunt to ground an applied transient or steadystate voltage. Used to protect against noise frequencies, faults, lightning, and inductive switching transients.

LINE REPLACEABLE UNIT. (18) An electronics unit that is made to be replaced on the flight line, as opposed to one that requires the aircraft be taken to the shop for repair.

LINEAR BUS. (18) A bus where users are connected to the medium; one on each end, with the rest connected in between.

LOW-PASS FILTER. (12) An electrical circuit which allows the passage of low frequencies and prevents the passage of high frequencies.

MAGNETIC FIELD. (11) A radiated, low-impedance field having lines of "flux" or magnetomotive force associated with an electrical current.

MALFUNCTION. (11) Failure or degradation in performance that compromises flight safety.

MANCHESTER II MODULATION. (18) A non-return to zero, bipolar modulation of a voltage that encodes bits based on the zero-crossing direction of the signal.

MEAN AERODYNAMIC CHORD (also mean chord). (4) The chord of an airfoil whose length is equal to the area of the airfoil section divided by the span.

MEAN FAILURE RATE. (10) A measure of survivability defined as the reciprocal of the mean time to system failure.

MESSAGE STRUCTURE. (6) The organization of both protocol and data information in a message.

METRIC. (17) A measure.

MICRON. (16) One-millionth of a meter.

MISSED ALARM. (7) The failure of a fault detection monitor or algorithm to detect a fault when there is a sensor fault.

MODELING. (18) Creating a system of mathematical equations that formulate all the significant behavior of a system.

MODULE. (17) A unit of code which implements a function.

MONITORABILITY. (6) The capacity of the protocol to be viewed passively to allow observation of the dynamics of the protocol.

MONOTONIC FUNCTION. (17) A function in which a certain change in the measure always represents a certain change in the property being measured, where either change is simply an increase or decrease in magnitude.

MULTIPLE BURST. (13) A randomly spaced series of bursts of short duration, low amplitude current pulses, with each pulse characterized by rapidly changing currents. These bursts may result from lightning leader progression or branching and may be accompanied by or superimposed on stroke or continuing currents. The multiple bursts appear to be most intense at the time of initial leader attachment to the aircraft.

MULTIPLE STRIKE. (13) Two or more lightning strikes during a single flight.

MULTIPLE STROKE. (13) Two or more return strokes occurring during a single lightning flash.

MULTIPLE TRIP MONITOR. (7) A fault detection algorithm which declares a fault after the sensor output has exceeded a predefined threshold N times.

MULTIVERSION PROGRAMMING. (18) N-version programming.

N-VERSION PROGRAMMING. (18) The independent coding of a number, N, of redundant computer programs that are run concurrently for the purpose of comparing their outputs.

NANOSECOND. (16) One-billionth of a second.

NEGATIVELY STABILIZED. (4) Aircraft design in which the point of effective lift is aft of the center of gravity.

NETWORK CONTROL STRATEGY. (6) The solution proposed by the designer in addressing his specific problem (design flexibility).

NOISE. (11) Conducted or radiated emission causing circuit upset, performance disorder, or undesired sound.

NUMERICAL APERTURE. (6) The angle of acceptance of light from a light source for a given fiber optic cable.

OBJECT CODE. (17) The translation of source code that is loaded into a computer.

OBSERVER. (7) An algorithm which models physical relationships between sensor data and uses the data to provide fault detection for one or more sensors. This is also known as a Luenberger observer or a signal blender.

OPERANDS. (17) The variables or constants on which the operators act.

OPERATORS. (17) Symbols which affect the value or ordering of operands.

OPTIMIZING COMPILER. (17) A computer program which, while translating source code into object code, removes inefficiencies from the code.

OVERHEAD. (18) The message timing gaps, control bits, and error detection bits added to some data to satisfy the data bus protocol.

PARAMETERIZATION CAPABILITY. (6) A measure of how well the attributes of the protocol can be described by parameters.

PARITY. (18) An error detection bit added to a data word based on whether the number of "one" bits is even or odd.

PARTITIONED. (18) Colocated hardware or software functions that are designed so that adverse interactions between them cannot occur.

PEAK RATE OF RISE. (13) The maximum instantaneous slope of the waveform as it rises to its maximum value. Mathematically, the peak rate of rise of a function, i(t), may be expressed as the maximum of d[i(t)]/dt.

PETRI NET. (18) A state analysis diagram that tracks the status of the state transition conditions of a state machine.

PIN LEVEL TEST. (12) An EMC test in which voltage or current is applied directly to a conductor at a connector pin.

POINT-MASS SIMULATION. (4) Same as state variables airplane model (q.v.).

POLLING. (18) A method whereby a CPU monitors the status of a peripheral by periodically reading its status signals.

POLYNOMIAL CODE. (18) A sequence of bits that represents the coefficients of each term in a polynomial.

POSITIVELY STABILIZED AIRCRAFT. (4) Aircraft design in which the effective point of lift is forward of the center of gravity.

PRECIPITATION STATIC (P-static). (11) Electrostatic discharge, corona, arcing, and streamering, steady state or impulsive, causing circuit upset, receiver noise or component damage.

PREDICATE/TRANSITION NETWORK. (4) A bipartite graph (a type of linear graph) to model concurrency between redundant concurrent events. Basically a modified generalized petri net.

PRIMARY STATION. (18) An intelligent HDLC protocol user, usually used to manage the access of other bus users to the bus.

PROGRAM. (17) A detailed set of instructions for accomplishing some purpose.

PROPAGATION DELAY. (18) The time it takes an electrical signal to travel from its source to its destination.

PROTOCOL. (18) The set of rules by which all bus users must abide to access the bus and ensure its specified operation.

Q. (12) The quality factor of a resonant circuit which is the ratio of the energy stored to the power dissipated per cycle.

QUADRUPLEX ARCHITECTURE. (4) The use of four separate lanes (or channels) of computer redundancy. Each lane can fail separately providing a fail-operational capability for the digital flight-control system.

QUALITY MEASURE. (17) A repeatable, monotonic relationship relating measures of objects (a set of numbers) to subjective qualities.

RADIATED EMISSION (RE). (11) Electromagnetic energy transmitted and propagated in space usually considered as audio frequency or radio frequency noise.

RADIO FREQUENCY (RF). (11) Frequencies in the electromagnetic spectrum used for radio communications extending from kilohertz to gigahertz.

RADIO FREQUENCY INTERFERENCE (RFI). (11) Electromagnetic interference in the radio frequency range.

RECONFIGURATION. (6) The capacity of a system to rearrange or reconnect the system elements or functions.

RECONFIGURATION. (18) The process of a system reassigning which hardware performs a particular function.

RECOVERY BLOCK. (18) A block of code executed upon detection of a fault to recover from the erroneous condition that results.

RECOVERY CACHE. (9) The location used to preserve input values until the outputs resulting from them have been accepted.

REDUNDANCY MANAGEMENT. (7) The computer processing which is needed to implement fault detection and isolation algorithms.

REFERENCE. (11) 1. Structure, for electronics, shields, power. 2. A grid of wires, solid sheet, or foil. 3. A wire from circuit to grounding block or case. 4. A wire from circuit to structure. 5. Shield tie. 6. Earth.

REGISTER. (18) A single word of RAM located within an IC controller that is used for transferring data and control information.

RELAXED STATIC STABILITY AIRCRAFT. (4) An aircraft whose center of gravity is behind the wing's point of effective lift.

RELIABILITY ANALYSIS. (4) A means of determining the probability of failure in a system. Military flight-critical systems typically are required to have reliability levels of $10^{-5}$ to $10^{-7}$, whereas civil flight-critical systems have reliability levels of $10^{-9}$ or less.

REMOTE TERMINAL. (18) The BIU portion of a MIL-STD-1553 bus user.

RESONANCE. (12) Resonance occurs in an electrical circuit when the energy stored in the inductance is equal to the energy stored in the capacitance.

RETURN. (11) 1. Structure, for power, fault, and "discrete" circuits. 2. A grid of wires, solid sheet, or foil. 3. A wire from circuit load back to source or to case. 4. Circuit card "ground plane," also a reference and shield.

RETURN STROKE. (13) See lightning return stroke.

REVERSION MODE. (7) The high level of redundancy in a system having different redundancy requirements for some sensors. Critical sensors may have a high level of redundancy while other sensors have low levels.

RING BUS. (18) A bus where users are connected only to the two adjacent users in a continuous ring; each connected to the next and the last one connected to the first one.

RISE-TIME. (12) The time required for a voltage pulse to reach a predefined magnitude from a given level.

ROBUSTNESS. (9) The ability of the code to perform despite some violation of the assumptions in its specifications usually via substitution of an alternate value and continuation of execution if a software fault is detected.

ROLLBACK. (9) Retrying the calculation in the event that a failure is detected, under the assumption that some external condition may have changed thereby resolving the anomaly.

SEALANT. (11) An applied substance enclosing and protecting the integrity of a joint, fastener, or electrical bond from moisture, contaminants, oxidation, and acid or alkaline corrosion.

SECONDARY STATION. (18) A simple HDLC protocol user.

SENSOR. (7) An instrument which measures a particular physical parameter. The data output may be digital or analog and is utilized by the flight computer.

SENSOR. (18) Any transducer that converts the measurement of a physical quantity to an electrical signal.

SEQUENTIAL LIKELIHOOD RATIO TEST. (7) A fault detection algorithm which is based on two hypothesized density functions of no fault or sensor fault.

SEQUENTIAL PROBABILITY RATIO TEST. (7) See sequential likelihood ratio test.

SERIAL DATA BUS. (18) A data bus capable of sending only one bit at a time, in series.

SERVICE SPECIFICATION. (18) The specification of the service provided by a protocol layer.

SHIELD. (11) A conductive material, opaque to electromagnetic energy, for confining or repelling electromagnetic fields. A structure, skin panel, case, cover, liner, foil, coating, braid, or cable-way that reduces electric and magnetic fields into or out of circuits or prevents accidental contact with hazardous voltages.

SHIELD EFFECTIVENESS (SE). (11) The ability of a shield to reject electromagnetic fields. A measure of attenuation in field strength at a point in space caused by the insertion of a shield between the source and the point.

SHIELDING. (12) Any metallic structure such as the aircraft fuselage or the woven braid on a cable that provides protection against electromagnetic fields.

SIGNAL RETURN. (11) A wire conductor between a load and the signal or driving source. Structure can be a signal and power return. Commonly, it is the low voltage side of the closed loop energy transfer circuit.

SIMULATION. (18) An approximated representation of the behavior of a system with a similar system.

SINGLE-ENDED CIRCUIT. (11) A circuit with source and load ends grounded to case and structure and using structure as return.

SINGLE-POINT FAILURE. (18) A failure of a component that, by itself, causes the failure of the system in which it is contained.

SINUSOID. (12) A wave form that follows the mathematical values of a sine function.

SOFT FAILURE. (12) A failure which causes an alteration of data or missing data.

SOFTWARE. (17) Computer programs and the documentation associated with the programs.

SOFTWARE METRIC. (17) A measure of software objects.

SOFTWARE QUALITY FACTOR. (17) Any software attribute that contributes either directly or indirectly, positively or negatively, toward the objectives for the system in which the software resides.

SOFTWARE QUALITY METRIC. (17) (1) A measure that relates measures of the software objects (the symbols) to the software qualities (quality factors). (2) The measure of a software quality factor.

SOURCE CODE. (17) Code that can be read by people.

SPECIAL CONDITION. (18) A regulatory document that adds to, or otherwise alters, the airworthiness standards for particular aircraft.

STATE-VARIABLE AIRPLANE MODEL (also point-mass model). (4) Fixed aerodynamic variables are used in the solution of the equations of motion of the model instead of using look-up tables in which each derivative varies with airspeed, altitude, etc. The model performance is only accurate at or near the point in the flight envelope for which the variables are chosen.

STATIC MARGIN. (4) The degree of instability in a relaxed statically stable airplane.

STATION. (18) Bus user.

STATIONARY BUS CONTROL. (18) Bus control that is continually performed by a single bus controller, or by one of its backups.

STATUS REGISTER. (18) A register in an IC controller that holds the status of the state of certain controller functions.

STROUD NUMBER. (17) The total number of elementary mental discriminations that a person makes per second.

STRUCTURAL IR VOLTAGE. (13) The portion of the induced voltage resulting from the product of the distributed lightning current, I, flowing through the resistance, R, of the aircraft skin or structure.

STRUCTURE. (11) Basic members, supports, spars, stanchions, housing, skin panels, or coverings that may or may not provide conductive return paths and shields for electrical/electronic circuits.

STUB. (18) The short length of cable used to attach a single LRU to a data bus.

SUBROUTINE. (17) A self-contained body of code which can be called by other routines to perform a function.

SUPER-DIAGNOSTIC FILTER. (7) An algorithm which provides all the capabilities of a diagnostic filter. Additionally, it can isolate a specific faulted sensor. At the current time, this is the most complex technique used to implement analytical redundancy.

SUSCEPTIBILITY. (11) Upset behavior or characteristic response of an equipment when subjected to specified electromagnetic energy. Identified with the point, threshold, or onset of operation outside of performance limits. Conducted Susceptibility (CS) applies to energy on interface conductors; Radiated Susceptibility (RS) to radiated fields.

SWEPT STROKE. (13) A series of successive attachments due to sweeping of the flash across the surface of the airplane by the motion of the airplane.

SYNCHRONOUS MESSAGES. (6) Messages transmitted at a known a priori sequence and time or time interval.

SYSTEM EXPOSURE TIME. (4) The period during which a system may fail. This period extends from the last verified proper functioning to the completion of the next required performance.

SYSTEM FUNCTIONAL UPSET. (13) Impairment of system operation, whether permanent or momentary (e.g., a change of digital or analog state) which may or may not require manual reset.

SYSTEM INTEGRATOR. (18) The developer who has the responsibility to integrate the various subsystems into a working system.

SYSTEM INTEGRITY. (6) The degree to which a system is dependable.

SYSTEM RELIABILITY. (5) The probability of performing a given function from the some initial time, t=0, to time t.

TESTABILITY. (6) A measure of how well the protocol supports completeness of testing and the protocol's ability to produce repeatable or predictable results.

THRESHOLD, NOISE. (11) The lowest electromagnetic interference signal level that produces onset of susceptibility.

THROUGHPUT. (6) The productivity of a data processing system as expressed in computing work per minute or hour.

THYRISTORS. (16) Solid-state devices that convert alternating current to direct current.

TIME CONSTANT. (4) Time required to double the amplitude of the divergent real root in the pitch axis of the aircraft model.

TOKEN PASSING PROTOCOL. (18) A protocol that limits bus access to the user that has just received the token word.

TRANSIENT CONTROL LEVEL. (13) The maximum allowable level of transients appearing at the systems interfaces as a result of the defined external environment.

TRANSPARENT RECOVERY. (4) Correcting a soft fault without interrupting the system's intended performance.

TRIBOELECTRIC CHARGING. (13) Static electricity produced on a structure from the effects of friction.

UNACCEPTABLE RESPONSE. (11) Upset, degradation of performance, or failure, not designated a malfunction, but is detrimental or compromising to cost, schedule, comfort, or workload.

UNBALANCED CONFIGURATION. (18) A bus using the HDLC protocol that connects one primary and one or more secondary stations.

UNDESIRABLE RESPONSE. (11) Change of performance and output, not designated a malfunction or safety hazard, that is evaluated as acceptable as is because of minimum nuisance effects and excessive cost burdens to correct.

UNIDIRECTIONAL DATA BUS. (18) A data bus with only one user that is capable of transmitting.

UPSET. (11) Temporary interruption of performance that is self-correcting or reversible by manual or automatic process.

UPSET. (12) A condition in which the state of a digital device is unintentionally altered, but may be restored by automatic means or by operator intervention.

UPSET. (13) See system functional upset.

VALIDATION. (4,11) Demonstration and authentication that a final product operates in all modes and performs consistently and successfully under all actual

operational and environmental conditions founded upon conformance to the applicable specifications.

VALIDATION. (18) The process of evaluating whether or not items, processes, services, or documents accomplish their intended purpose in their operating environment.

VERIFICATION. (4,11) Demonstration by similarity, previous in-service experience, analysis, measurement, or operation that the performance, characteristics, or parameters of equipment and parts demonstrate accuracy, show the quality of being repeatable, and meet or are acceptable under applicable specifications.

VERIFICATION. (18) The act of reviewing, inspecting, testing, checking, auditing, or otherwise establishing and documenting whether or not items, processes, services, or documents conform to specified requirements.

VOTING PROCEDURE. (8) An algorithm included in fault tolerant software which uses the consensus recovery block method. It compares outputs of the n independent versions and determines which outputs are correct by identifying agreements among two or more versions.

WELL-BEHAVED FUNCTION. (17) A smooth mathematical relationship.

## ACRONYMS AND ABBREVIATIONS

| | |
|---|---|
| $\lambda$ {17} | Language Level |
| $\mu$m {6} | Micrometer |
| $\mu$s {18} | Microsecond |
| $\eta$ {17} | Vocabulary of a Program |
| $\eta_1$ {17} | Number of Unique Operators |
| $\eta_1^*$ {17} | Minimum Number of Unique Operators |
| $\eta_2$ {17} | Number of Unique Operands |
| $\eta_2^*$ {17} | Number of Different Input and Output Parameters |
| $\phi$M {6} | Phase Modulation |
| $1/E_0$ {17} | Average number of discriminations a person is likely to make for each bug introduced into the code. |
| 3-D {16} | Three-Dimensional |
| A/C {11} | Aircraft |
| A/L {3} | Approach/Land |
| AAES {15} | Advanced Aircraft Electrical System |
| ac {3,6,12,15} | Alternating Current |
| AC {3,5,14,17,18} | Advisory Circular |
| ACAP {13} | Advanced Composite Airframe Program |
| ACARS {11,12} | ARC Communications Addressing and Reporting System |
| ACES {13} | Applied Computational Electromagnetics Society |
| ACK {18} | Acknowledge |
| ACO {18} | Aircraft Certification Office |
| ACS {16} | Automatic Control System |
| ACT {11,12} | Active Controls Technology |
| ACT {17} | Analysis of Complexity Tool |
| ADC {11,12} | Air Data Computer |
| ADF {11,12} | Automatic Direction Finder |
| ADI {3} | Automatic Direction Indicator |
| AE {6} | Avionics Equipment |
| AE4L {5,13} | SAE Subcommittee (Lightning) |
| AEEC {18} | Airlines Electronic Engineering Committee |
| AEHP {13} | Atmospheric Electricity Hazards Protection |
| AERA {16} | Automated En Route Air Traffic Control System |
| AES-S {6} | Aerospace and Electronic Systems Society |
| AF {11,12} | Audio Frequency |
| AFBW {4} | Augmented Fly-By-Wire |
| AFCS {11,12} | Automatic Flight Control System |
| AFFDL {7,8,13} | Air Force Flight Dynamics Laboratory |
| AFM {16} | Advanced Fuel Management |
| AFSC {18} | Air Force Systems Command |
| AFWAL {6,13} | Air Force Wright Aeronautical Laboratory |
| AGARD {8} | Advisory Group for Aerospace Research and Development |

| | |
|---|---|
| AHRS (6) | Attitude Heading Reference System |
| AI (16) | Artificial Intelligence |
| AIAA (5,6,9,15,18) | American Institute of Aeronautics and Astronautics |
| AIM (18) | Advanced Integrated MUX |
| AIR (18) | Aerospace Information Report |
| AIRLAB (5,18) | Avionics Integration Research Laboratory |
| AK (7) | Altitude Kinematics |
| ALCM (13) | Air Launched Cruise Missile |
| ALU (3,5,10) | Arithmetic Logic Unit |
| AM (5) | Amplitude Modulated |
| AM (6,14) | Amplitude Modulation |
| AMSC (5) | Document number prefix used by the Department of Defense |
| ANSI (11,12) | American National Standards Institute |
| AOA (4) | Angle of Attack |
| AP (18) | Application Processor |
| APU (11,12,15) | Auxiliary Power Unit |
| AR (7) | Analytical Redundancy |
| ARC (11,12) | Aeronautical Radio, Incorporated |
| ARIES (3) | Automated Reliability Interactive Estimation System |
| ARINC (3,6,18) | Aeronautical Radio, Incorporated |
| ARP (17,18) | Aerospace Recommended Practice |
| ARTERI (15) | Analytical Redundancy Technology for Engine Reliability Improvement |
| ASCB (6,18) | Avionics Standard Communications Bus |
| ASDS (11,12) | Airport Surface Detection System |
| ASEE (5,15) | American Society of Electrical Engineers |
| ASME (5) | American Society of Mechanical Engineers |
| ATCRBS (11,12,14) | Air Traffic Control Radar Beacon System |
| ATF (16) | Advanced Tactical Fighter |
| ATI (16) | Access Time Interval |
| ATTR (5) | Attribute |
| AWACS (14) | Airborne Warning and Control System |
| $\hat{B}$ (17) | Number of Bugs (Estimated) |
| B-dot (13) | Derivative of the magnetic field with respect to time |
| B-GLOSS (5) | Gate Logic Software Simulator developed by Bendix |
| BAC (18) | Balanced Asynchronous Configuration |
| BAT (17) | Battlemap Analysis Tool |
| BB (11,12) | Broadband |
| BC (18) | Bus Controller |
| BCAC (18) | Boeing Commercial Airplane Company |
| BCD (18) | Binary Coded Decimal |
| BCI (12) | Bulk Cable Injection |
| BFCS (18) | Beacon Frame Check Sequence |
| BGU (10) | Bus Guardian Unit |
| BIR (6) | Benchmark Information Rate |
| BIT (6,15,18) | Built-In Test |
| BITE (6,11,12) | Built-In Test Equipment |
| BIU (6,18) | Bus Interface Unit |
| BNR (18) | Binary |

| | |
|---|---|
| BOCP (18) | Bit-Oriented Communications Protocol |
| BP (18) | Basic Protocol |
| bps (6) | bits per second |
| BUSY (18) | Destination Busy |
| BW (11,12) | Bandwidth |
| | |
| C (7) | Comparator |
| C/I (5) | Communicator Interstage |
| CA (18) | Criticality Analysis |
| CAA (14) | Civil Aviation Authority |
| CAD (5) | Computer Aided Design |
| CAP (5) | Collins Application Processor |
| CAPS (3) | Computer Aided Production Simulator |
| CARE (3,5) | Computer Aided Reliability Evaluator |
| CARSRA (3,7) | Computer-Aided Redundant System Reliability Analysis |
| CAS (11,12) | Criticality Advisory System |
| CAST (3) | Complementary Analytic Simulative Technique |
| CBD (5) | Commerce Business Daily |
| CCITT (6) | Consultative Committee for International Telephone and Telegraph |
| CD (6) | Collision Detection |
| cdf (8) | Cumulative Density Function |
| CDU (11,12) | Control Display Unit |
| CE (11,12) | Conducted Emission |
| CE (17,18) | Certification Engineer |
| CFR (17) | Code of Federal Regulations |
| CM (11,12) | Common Mode |
| CMC (18) | Current Mode Coupler |
| CMOS (5,12) | Complimentary Metal-Oxide Semiconductor |
| CONUS (14) | Contiguous United States |
| CP (18) | Combined Protocol |
| CPA (5) | Central Processor - A |
| CPU (5,10,18) | Central Processing Unit |
| CR (5) | Contractor Report |
| CR (6) | Command Response |
| CR/LF (17) | Carriage Return/Line Feed |
| CRC (6,18) | Cyclic Redundancy Check |
| CRMI (2) | Computer Resource Management, Incorporated |
| CRT (11,12,16) | Cathode Ray Tube |
| CS (11,12) | Conducted Susceptibility |
| CSC (9) | Computer Software Component |
| CSCI (9,17) | Computer Software Configuration Item |
| CSDB (18) | Commercial Standard Data Bus |
| CSDL (5,10,15) | Charles Stark Draper Laboratories |
| CSMA (6,16,18) | Carrier Sensed Multiple Access |
| CSMA/CD (6) | Carrier Sense Multiple Access/Collision Detection |
| CT (2,6) | Technical Center (designation used in FAA report numbering scheme) |
| CTA (3) | CAPS Test Adapter |
| CTA (5) | Collins Test Adaptor |
| CTS (18) | Clear To Send |

| | |
|---|---|
| CW {13} | Continuous Wave |
| D {17} | Program Difficulty |
| DADC {6} | Digital Air Data Computer |
| DARPA {16} | Defense Advanced Research Projects Agency |
| DATAC {6,18} | Digital Autonomous Terminal Access Communication |
| dB {6,12} | Decibel |
| dBi {14} | Decibels with respect to one milliampere |
| dBm {6} | Decibels with respect to one milliwatt |
| dc {6,12,15} | Direct Current |
| DC {18} | Display Computer |
| DE {5} | Diagnostic Emulation |
| DEFN {5} | Definition |
| DET {18} | Driver Enable Timer |
| DEV {5} | Development |
| DF {7} | Diagnostic Filter |
| DFC {7} | Digital Flight Control |
| DFCS {3,4,7,16} | Digital Flight Control System |
| DFDAU {11,12} | Digital Flight Data Acquisition Unit |
| DFDR {11,12} | Digital Flight Data Recorder |
| DGAC {14} | Directorate Generale Aviation Civile |
| DISAC {15} | Digital Integrated Servo Actuator Controller |
| DITS {6,11,12,18} | Digital Information Transfer System |
| DM {11} | Differential Mode |
| DM {6} | Delay Modulation |
| DMA {5,6} | Direct Memory Access |
| DMA {18} | Direct Memory Addressing |
| DME {11,12,18} | Distance Measuring Equipment |
| DNA {13} | Defense Nuclear Agency |
| DOD {5,8,12,14,16} | Department of Defense |
| DOE {13} | Department of Energy |
| DOT {2,3,6,7,8} | Department of Transportation |
| DR {17} | Direct Ratio (Average) |
| DRB {9} | Distributed Recovery Block |
| DS {17} | Direct Score |
| DSP {3} | Discrete Switch Panel |
| DTSA {18} | Dynamic Time Slot Allocation |
| E {17} | Programming Effort |
| E-dot {13} | Derivative of the electric field with respect to time |
| E-FIELD {11,12} | Electric Field |
| E/E {11,12} | Electrical/Electronic |
| $E^3$ {11,12} | Electromagnetic Environmental Effects |
| EADI {11,12} | Electronic Attitude Director Indicator |
| ECAC {11,12,14} | Electromagnetic Compatibility Analysis Center |
| ECM {14} | Electronic Counter Measures |
| ECS {11,12} | Environmental Control System |
| EEC {5,11,12,18} | Electronic Engine Control |
| EED {11,12} | Electro-Explosive Device |
| EES {18} | Electromagnetic Emission and Susceptibility |
| EFID {18} | Electronic Flight Instrument Display |

| | |
|---|---|
| EFIS {11,12,18} | Electronic Flight Instrument System |
| EFMA {3} | Executive Failure My A |
| EFMB {3} | Executive Failure My B |
| EFOA {3} | Executive Failure Other A |
| EFOB {3} | Executive Failure Other B |
| EFW {3} | Executive Failure Word |
| EGT {11,12} | Exhaust Gas Temperature |
| EHSI {11,12} | Electronic Horizontal Situation Indicator |
| EIA {18} | Electronic Industries Association |
| EICAS {11,12} | Engine Indication and Crew Alerting System |
| EIU {16} | Electronic Interface Unit |
| EM {5,11,12,13} | Electromagnetic |
| EMA {15} | Electromechanical Actuator |
| EMAS {2,15} | Electromechanical Actuator System |
| EMC {6,11,12,13,14} | Electromagnetic Compatibility |
| EMCad™ {12} | Electromagnetic Computer aided design |
| EME {14} | Electromagnetic Environment |
| EME {11,12} | Electromagnetic Effects |
| EMI {5,6,11,12,13,15,16} | Electromagnetic Interference |
| EMIC {11,12} | Electromagnetic Interference/Compatibility |
| EMP {11,12,13} | Electromagnetic Pulse |
| EMR {5,14} | Electromagnetic Radiation |
| EMUX {6} | Electrical Multiplex |
| ENRZ {6} | Enhanced Non-Return to Zero |
| EOF {17} | End of File |
| EPR {11,12,15} | Engine Pressure Ratio |
| EPROM {16} | Erasable Programmable Read-Only Memory |
| ESD {11,12} | Electrostatic Discharge |
| ESE {11,12} | Electric (field) Shield Effectiveness |
| ESS {5,9} | Electronic Switching System |
| ETDL {13} | Equipment Transient Design Level |
| EUROCAE {14} | European Organization for Civil Aviation Electronics |
| EXCHNG {5} | Exchange |
| EXP {5} | Experiment |
| | |
| F/FA {18} | Fault and Failure Analysis |
| FAA {ALL} | Federal Aviation Administration |
| FADEC {6,15,16} | Full Authority Digital Engine Controller |
| FAFTEEC {16} | Full Authority Fault Tolerant Electronic Engine Control |
| FAR {3,4,6,16} | Federal Acquisition Regulation |
| FAR {17,18} | Federal Aviation Regulation |
| FBL {15,16} | Fly-By-Light |
| FBW {4,7,16} | Fly-By-Wire |
| FCC {3,4,5,6,10,11,12,15,18} | Flight Control Computer |
| FCR {5} | Fault Containment Regions |
| FCS {4,5,10,16} | Flight Control System |
| FCS {18} | Frame Check Sequence |
| FD {7} | Fault Detection |
| FDEP {11,12} | Flight Data Entry Panel |
| FDFM {15} | Fault Detection and Failure Management |

| | |
|---|---|
| FET {5,16} | Field Effect Transistor |
| FI {5} | Fault Insertion Circuitry |
| FI {7} | Fault Isolation |
| FIAT {5} | Fault Injection Automated Testing |
| FICA {15} | Failure Indication and Corrective Action |
| FIIS {5,10} | Fault Insertion and Instrumentation System |
| FIM {5} | Fault Injection Manager |
| FIRE {5} | Fault Injection Receptor |
| FM {5} | Frequency Modulated |
| FM {6,14} | Frequency Modulation |
| FMC {11,12} | Flight Management Computer |
| FMEA {3,9,18} | Failure Mode and Effect Analysis |
| FMECA {5,15} | Failure Modes and Effects Criticality Analysis |
| FMECA {18} | Failure Mode, Effects, and Criticality Analysis |
| FP {17} | Function Point |
| FSM {18} | Finite State Machine |
| ft {14} | feet |
| FT {5} | Fault Tolerant |
| FTA {18} | Fault Tree Analysis |
| FTC {5} | Fault Tree Compiler |
| FTMP {5,10,18} | Fault-Tolerant Multiprocessor |
| FTP {5} | Fault Tolerant Processor |
| | |
| G/E {13} | Graphite Epoxy |
| GA {18} | General Aviation |
| GaAs {6,16} | Gallium Arsenide |
| GAMA {6,18} | General Aviation Manufacturers Association |
| GCR {6} | Group Code Recording |
| GE {5} | General Electric |
| GEMACS {13} | General Electromagnetic Model for the Analysis of Complex Systems |
| GEN {5} | Generation |
| GGLOSS {5} | Generalized Gate-Level Logic System Simulator |
| GLOSS {5} | Gate Logic Software Simulator |
| GNC {16} | Guidance, Navigation, and Control |
| GPC {5} | General Purpose Computer |
| GPS {11,12} | Global-Positioning-System |
| GPWS {11,12} | Ground Proximity Warning System |
| Gr/Ep {11,12} | Graphite/Epoxy |
| GS {10} | Glideslope |
| | |
| H-FIELD {11,12} | Magnetic Field |
| Hl {11,12} | Fan Speed |
| HA {18} | Hazard Analysis |
| HARP {5,18} | Hybrid Automated Reliability Predictor |
| HDBK {5} | Handbook |
| HDLC {6,18} | High-Level Data Link Control |
| HERF {5,14,16,18} | High-Energy Radio Frequency |
| HF {11,12,13,14} | High-Frequency |
| HIRF {15,18} | High-Intensity Radiated Fields |
| HOL {17} | High Order Language |
| HSI {3} | Horizontal Situation Indicator |

| | |
|---|---|
| HSRB (6,18) | High-Speed Ring Bus |
| HVDC (15) | High Voltage dc |
| HW (18) | Hardware |
| Hz (3,15,18) | Hertz |
| | |
| I (13) | Current |
| I (17) | Intelligence Content |
| I-dot (13) | Derivative of the current with respect to time |
| I/O (5,10,15,18) | Input/Output |
| IAA (3) | Integrated Assurance Assessment |
| IAAC (11,12) | Integrated Application of Active Controls Technology (to an Advanced Subsonic Transport Project) |
| IACS (18) | Integrated Avionic Computer System |
| IBM (5) | International Business Machines |
| IC (3,18) | Integrated Circuit |
| ICAO (14) | International Civil Aviation Organization |
| ICIS (5) | Intercomputer Interface Sequencer |
| ICS (5) | Intercomputer Sequencer |
| ID (5) | Identification |
| ID (18) | Identifier |
| IDG (11,12) | Integrated Drive Generator |
| IEEE (5,6,9,17,18) | Institute of Electrical and Electronics Engineers, Incorporated |
| IFC (17) | Information Flow Complexity |
| IFCS (18) | Information Frame Check Sequence |
| IFF (14) | Identification - Friend or Foe |
| IGGLOSS (5) | Gate Logic Software Simulator (improved version developed at NASA Langley) |
| ILS (3,11,12) | Instrument Landing System |
| IMR (18) | Interrupt Mask Register |
| INS (6,11,12) | Inertial Navigation System Institute |
| IOPA (5) | Input/Output Processor - Channel A |
| IOS (5) | Input/Output Subsystem |
| IRS (11,12) | Inertial Reference System |
| ISO (17,18) | International Standards Organization |
| ITT (16) | (Consultative Committee for) International Telegraphy and Telephony |
| IV&V (9) | Independent Verification and Validation |
| IVT (18) | Interrupt Vector Table |
| | |
| JPL (5) | Jet Propulsion Laboratories |
| | |
| K (6) | Thousand |
| kA (13,16) | Kiloampere |
| kHz (5,6,12,14) | Kilohertz |
| km (6) | kilometer |
| | |
| L (17) | Program Level |
| $\hat{L}$ (17) | Estimated Program Level |
| LAN (5) | Local Area Network |
| LaRC (5) | Langley Research Center |

| | |
|---|---|
| LCC [11,12] | Life Cycle Cost |
| LCD [16] | Liquid Crystal Display |
| LED [6] | Light Emitting Diode |
| LF [13] | Low Frequency |
| LOC [11,12] | Localizer |
| LPN [13] | Lumped Parameter Network |
| LRC [6] | Longitudinal Redundancy Check |
| LRRA [11,12] | Low Range Radio Altimeter |
| LRU [5,6,11,12,13,18] | Line Replaceable Unit |
| LSB [6,18] | Least Significant Bit |
| LSI [5,18] | Large Scale Integration |
| LTPB [6,18] | Linear Token Passing Bus |
| LTRI [13] | Lightning and Transients Research Institute |
| LVDT [3] | Linear Voltage Differential Transducer |
| LVDT [15] | Linear Variable Differential Transformers |
| | |
| m [6] | meter |
| m [18] | Original Address of Last Transmission |
| M [5] | Mutual (when used with RLC) |
| M [6] | Million |
| mA [3] | Milliampere |
| MAADS [6] | Multibus Avionic Architecture Design Study |
| MAC [4] | Mean Aerodynamic Chord |
| MAFT [16] | Multicomputer Architecture for Fault Tolerance |
| Mbps [6,16] | Million bytes per second |
| MC [18] | Mode Code |
| MCDP [11,12] | Maintenance Control and Display Panel |
| MCFCS [18] | Message Control Frame Check Sequence |
| MCP [11,12] | Mode Control Panel |
| MDICU [3] | Modular Digital Interface Control Unit |
| MDICU [4,5] | Modular Digital Interface Conversion Unit |
| MDT [6] | Mean Down Time |
| MFCS [18] | Message Frame Check Sequence |
| Mflops [16] | Million floating-point operations per second |
| MFM [6] | Modified-Frequency Modulation |
| MFR [10] | Mean Failure Rate |
| MHz [5,6,12,14,18] | Megahertz |
| mil [11,12] | One thousandths of an inch (0.001) |
| MIL [5] | Military |
| MIL-HDBK [18] | Military Handbook |
| MIL-STD [6,18] | Military Standard |
| ML [18] | Message Length |
| MLE [8] | Maximum Likelihood Estimates |
| MLS [11,12] | Microwave Landing System |
| MOS [5] | Metal-Oxide Semiconductor |
| MPP [16] | Massively Parallel Processor |
| MPSC [18] | Multi-Protocol Serial Controller |
| MPX [5] | Multiplex |
| ms [6,10,18] | Millisecond |
| MSB [6,18] | Most Significant Bit |
| MSE [11,12] | Magnetic (Field) Shielding Effectiveness |
| MSI [5] | Medium Scale Integration |

| | |
|---|---|
| MT {18} | Message Time |
| MTBCF {6} | Mean Time Between Critical Failures |
| MTBF {9,18} | Mean Time Between Failures |
| MTTF {8,10} | Mean Time to Failure |
| MTTR {6,18} | Mean Time To Repair |
| Mux {5} | Multiplexed |
| MUX {4,18} | Multiplexer |
| | |
| n {18} | Address of User Performing Computation |
| N {17} | Implementation Length of a Program |
| N {18} | Maximum Number of Users |
| $\hat{N}$ {17} | Estimated Length |
| $N_1$ {15} | Low Rotor Speed |
| $N_1$ {17} | Total Number of Operator Occurrences |
| $N_1^*$ {17} | Minimum Number of Operators |
| N2 {11,12} | Core Engine Speed |
| $N_2$ {15} | High Rotor Speed |
| $N_2$ {17} | Total Number of Operand Occurrences |
| $N_2^*$ {17} | Minimum Number of Operands |
| NA {6} | Numerical Acceptance |
| NA {3} | Normal Accelerometers |
| NADC {6,7} | Naval Air Development Center |
| Naecon {5} | National Avionics and Electronics Conference |
| NAECON {6} | National Aerospace and Electronics Conference |
| NAND {5} | Not AND |
| NASA {2,3,5,7,13,15,18} | National Aeronautics and Space Administration |
| NASC {13} | Naval Air Systems Command |
| NATO {14} | North Atlantic Treaty Organization |
| NB {11,12} | Narrow Band Signal |
| NCTS {18} | Not Clear To Send |
| NEC {13} | Numerical Electromagnetics Code |
| NEMP {5,12} | Nuclear Electromagnetic Pulse |
| NHPP {8} | Non-Homogeneous Poisson Process |
| nmi {14} | nautical mile |
| NPRM {14} | Notice of Proposed Rulemaking |
| NRZ {6} | Non-return to Zero |
| NRZ-I {6} | Non-return to Zero Inverted |
| NRZ-L {6} | Non-return to Zero Dual Level |
| nsec {6} | Nanosecond |
| NSWC {13} | Naval Surface Weapons Center |
| NVS {8} | N-version Software |
| | |
| OMEGA {11,12} | Very Low Frequency Navigation |
| OMV {16} | Orbital Maneuvering Vehicle |
| OS {5} | Operating System |
| OSI {18} | Open Systems Interconnection |
| OTV {16} | Orbital Transfer Vehicle |
| | |
| P {10} | Processor |
| P-Static {11,12} | Precipitation Static |
| PAL {5} | Programmable Array Logic |
| PAS {6} | Pilot Assist System |

| | |
|---|---|
| PAWS (5) | Padé Approximation With Scaling |
| PBW (15) | Power-By-Wire |
| PC (5) | Personal Computer |
| PC (17) | Processing Complexity |
| PCA (17) | Processing Complexity Adjustment |
| PCS (16) | Primary Control System |
| PCU (11,12) | Power Control Unit |
| pdf (8) | Probability Density Function |
| PE (6) | Phase Encoding |
| pf (12) | picofarad |
| PLA (15) | Power Level Actuator |
| PLA (16) | Power Level Angle |
| PMA (18) | Parts Manufacturer Approval |
| PMS (5) | Physical Message Switch |
| PRF (11,12) | Pulse Repetition Frequency |
| PROC (5) | Processor |
| PROM (3,10,17,18) | Programmable Read-Only Memory |
| psi (15) | pounds per square inch |
| PVI (16) | Pilot/Vehicle Interface |
| PWM (11,12) | Pulse Width Modulation |
| PZ (15) | Piezoelectric |
| | |
| QUAD (5) | Quadruple |
| | |
| R (13) | Resistance |
| R-C (12) | Resistor-Capacitor |
| RADC (8,17) | Rome Air Development Center |
| RAE (13) | Royal Aircraft Establishment |
| RAM (3,5,10,16,17,18) | Random Access Memory |
| RAT (6,18) | Ring Admittance Timer |
| RB (8) | Recovery Block |
| RBDCP (3) | Reliability Block Diagram Computer Program |
| RCA (16) | Radio Corporation of America |
| RCVR (5) | Receiver |
| RDFCS (3) | Redundant Digital Flight Control System |
| RDFCS (4) | Reconfigurable Digital Flight Control System (facility) |
| RDFCS (5) | Reconfigurable Digital Flight Control System |
| RDMI (11,12) | Radio Distance Magnetic Indicator |
| RE (11,12) | Radiated Emission |
| REG (5) | Register |
| REL (3) | Reliability |
| REL COMP (3) | Reliability Computers |
| RF (5,11,12,13,14,16,18) | Radio Frequency |
| RFI (11,12) | Radio Frequency Interference |
| RIM (18) | Ring Interface Module |
| RIU (18) | Ring Interface Unit |
| RL (13,15) | Resistance/Inductance (Out of order in c.15) |
| RLC (5,13) | Resistance/Inductance/Capacitance |
| RLCM (13) | Resistance/Inductance/Capacitance/Mutual |
| RM (7) | Redundancy Management |
| RNRZ (6) | Randomized Non-return to Zero |

| | |
|---|---|
| ROM {5,10} | Read Only Memory |
| RPV {16} | Remotely Piloted Vehicle |
| RR {18} | Read Register |
| RRT {18} | Ring Rotation Time |
| RS {11,12} | Radiated Susceptibility |
| RS {18} | Recommended Standard |
| RSS {4,7} | Relaxed Static Stability |
| RT {5,18} | Remote Terminal |
| RTCA {2,3,11,12,14,16,17,18} | Radio Technical Commission for Aeronautics |
| RTE {18} | Real-Time Executive |
| RTI {5} | Remote Terminal Interface |
| RTS {18} | Request To Send |
| RZ {6} | Return to Zero |
| | |
| S {17} | Stroud Number |
| S-a-0 {5,10} | Stuck at Zero |
| S-a-1 {5,10} | Stuck at One |
| S-GLOSS {5} | Gate Logic Software Simulator developed by Stevens |
| S/A {11,12} | Spectrum Analyzer |
| SAE {2,5,6,13,14,16,17,18} | Society of Automotive Engineers |
| SAI {18} | Systems Architecture and Interfaces |
| SAS {4,7} | Stability Augmentation System |
| SC {18} | Special Condition |
| SCC {18} | System Configuration Controller |
| SCM {18} | Software Configuration Management |
| SCP {18} | Self-Checking Pair |
| SDF {7} | Super-Diagnostic Filter |
| SE {11,12} | Shielding Effectiveness |
| SEU {5} | Single Event Upset |
| SG {18} | Synchronization Gap |
| SHF {11,12} | Super High-Frequency |
| SHRD {5} | Shared |
| SIAM {15} | Society for Industrial and Applied Mathematics |
| SIF {14} | Selective Identification Facility |
| SIM {18} | Serial Interface Module |
| SIR {18} | Shared Interface RAM |
| SLRT {7} | Sequential Likelihood Ratio Test |
| SMF {18} | Self Monitor Function |
| SMOTEC {14} | Special Missions Operation Test and Evaluation Center |
| SO {17} | Second Order (Average) |
| SPRT {7} | Sequential Probability Ratio Test |
| SQA {18} | Software Quality Assurance |
| SQF {17} | Software Quality Factor |
| SQL {5} | Software Query Language |
| SQM {17} | Software Quality Metrics |
| SQPP {17} | Software Quality Program Plan |
| SRS {17} | Software Requirement Specification |
| SSA {18} | System Safety Assessment |
| SSI {5} | Small Scale Integration |
| SSP {3} | Servo Simulation Panel |
| SSPC {15} | Solid-State Power Controller |

| | |
|---|---|
| SSS (17) | System/Segment Specification |
| STANAG (14) | Standardization Agreement (NATO) |
| STC (2) | Supplemental Type Certification |
| STC (18) | Supplemental Type Certificate |
| STEM (5) | Scaled Taylor Expansion Matrix |
| STOL (16) | Short Takeoff and Landing |
| str (6) | string |
| SURE (5) | Semi-Markov Unreliability Range Evaluator |
| SW (18) | Software |
| SYN (5) | Synch |
| | |
| $\hat{T}$ (17) | Estimated Programming Time |
| T/R (6) | Transmitter/Receiver |
| TACAN (14) | Tactical Air Navigation |
| TASRA (3) | Tree Aided System Reliability Analysis |
| $T_c$ (18) | Count Duration |
| TC (2) | Type Certification |
| TC (18) | Type Certificate |
| TCAP (13) | Threshold Circuit Analysis Program |
| TCAS (11,12,16,18) | Traffic Alert and Collision Avoidance System |
| TCL (13) | Transient Control Level |
| TDM (6) | Time Division Multiplex |
| TDMA (18) | Time Division Multiple Access |
| $T_F$ (18) | Frame Time |
| TFCS (18) | Token Frame Check Sequence |
| TFEDF (18) | Token Frame Ending Delimiter Field |
| TG (18) | Terminal Gap |
| THT (6,18) | Token-Holding Timer |
| TI (18) | Transmit Interval |
| TLA (11,12) | Thrust Lever Angle |
| $T_m$ (18) | Wait Time for User |
| TMC (11,12) | Thrust Management Computer |
| TMR (10) | Triple Modular Redundant |
| TRT (18) | Token Rotation Timer |
| TRU (15) | Transformer Rectifier Unit |
| TSDF (18) | Token Starting Delimiter Field |
| TSO (18) | Technical Standard Order |
| TTL (5,11,12,13,16) | Transistor-Transistor Logic |
| TV (14) | Television |
| TWTD (13) | Thin Wire Time Domain |
| TX (5) | Transmit |
| | |
| U.K. (13,14) | United Kingdom |
| U.S. (14) | United States |
| UAC (18) | Unbalanced Asynchronous Configuration |
| UART (15) | Universal Asynchronous Receiver Transmitter |
| UHF (11,12,13,14) | Ultra High-Frequency |
| UNC (18) | Unbalanced Normal Configuration |
| UNIBUS (5) | Universal Bus |
| UPS (15) | Uninterruptible Power Supplies |
| USAF (16) | United States Air Force |
| USB (16) | Upper Surface Blowing |

| | |
|---|---|
| USEG (5) | Unsegmented |
| V (17) | Volume |
| V' (17) | Potential Volume |
| V&V (18) | Verification and Validation |
| V/m (14) | Volt/meter |
| VHF (11,12,13,14) | Very High-Frequency |
| VHSIC (6,16) | Very-High-Speed Integrated Circuits |
| VLF (11,12) | Very Low-Frequency |
| VLSI (5,6,14,18) | Very Large Scale Integration |
| VLSIC (6,16) | Very Large Scale Integrated Circuits |
| VOR (11,12,14,18) | VHF Omnidirectional Range |
| VORTA/VHF (11,12) | Omnirange/Tactical Air Navigation |
| VRC (6) | Vertical Redundancy Check |
| VSI (11,12) | Vertical Speed Indicator |
| VSV (15) | Variable Stator Vane |
| VT (18) | Validation Testing |
| VTOL (16) | Vertical Takeoff and Landing |
| W/P (15) | Fuel Flow to Burner Pressure |
| WAI (3) | Warning Annunciation Indicator |
| WFM (15) | Main fuel metering valve actuator sensor |
| WR (18) | Write Register |
| WRU (11,12) | Weapons Replaceable Unit |
| WSO (17) | Weighted Second Order (Average) |
| XAB (5) | Transmit Compare A B |
| XMT (5) | Transmit |
| XMTR (3) | Transmitter |
| XOR (5) | Exclusive OR |
| ZM (6) | Zero Modulation |