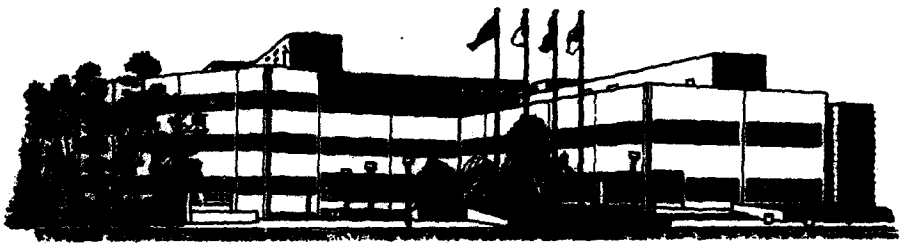
U.S. Department of Transportation
Federal Aviation Administration

DOT/FAA/CT-88/10

HANDBOOK - VOLUME II
DIGITAL SYSTEMS VALIDATION

CHAPTER 18
AVIONIC DATA BUS
INTEGRATION TECHNOLOGY



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
Handbook - Volume II Digital Systems Validation Chapter 18 Avionic Data Bus Integration Technology	November 1993
	6. Performing Organization Code
7. Author(s) D. Elwell, L. Harrison, J. Hensyl, and N. VanSuetendael	8. Performing Organization Report No. DOT/FAA/CT-88/10-Vol-2-Ch-18
9. Performing Organization Name and Address Computer Resource Management, Inc. 200 Scarborough Drive, Suite 108 Pleasantville, New Jersey 08232	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 Tutorial Handbook Chapter 18
	14. Sponsoring Agency Code 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

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

LIST OF ILLUSTRATIONS

LIST OF TABLES

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.

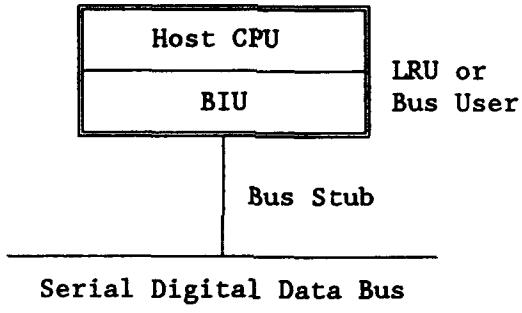


FIGURE 2.1-1. DATA BUS COMPONENTS

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.

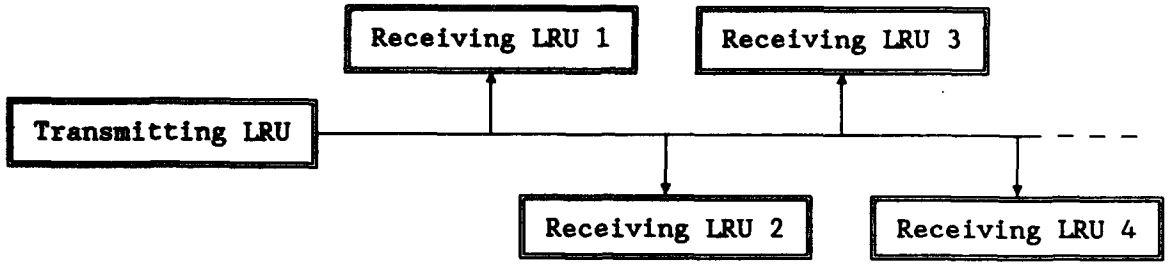


FIGURE 2.1-2. UNIDIRECTIONAL BUS ARCHITECTURE

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.

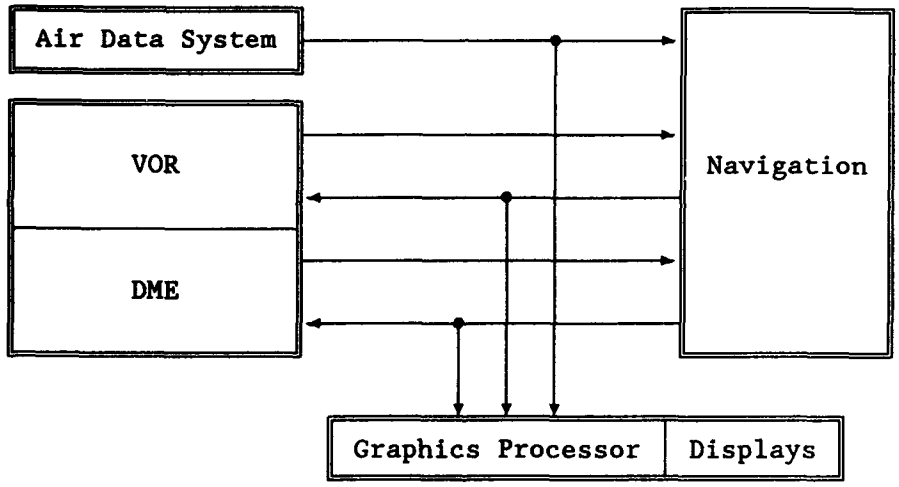


FIGURE 2.1-3. AVIONIC SYSTEM USING UNIDIRECTIONAL BUSES
(Hitt 1986)

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.

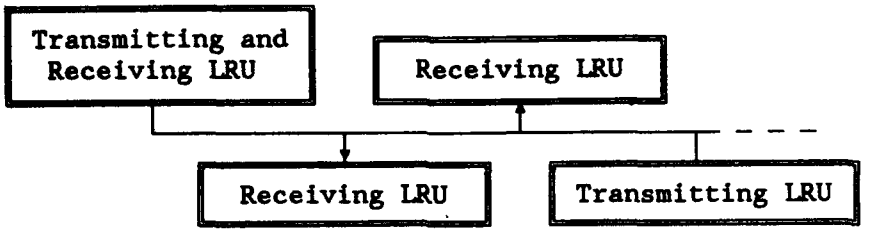


FIGURE 2.1-4. BIDIRECTIONAL BUS ARCHITECTURE

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.

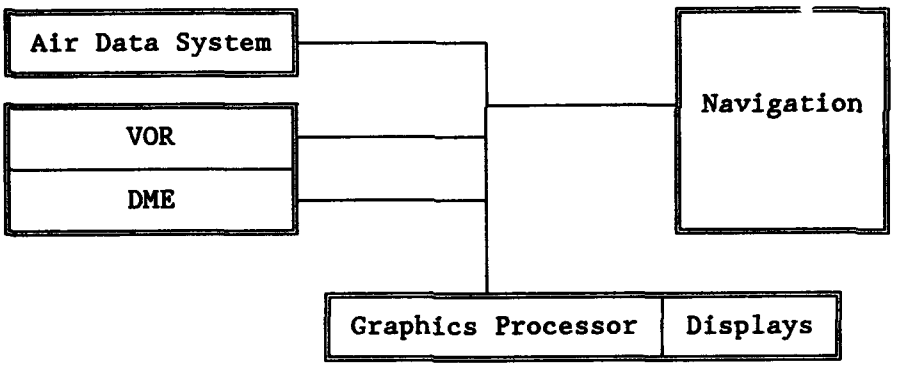


FIGURE 2.1-5. AVIONIC SYSTEM USING A BIDIRECTIONAL BUS

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.

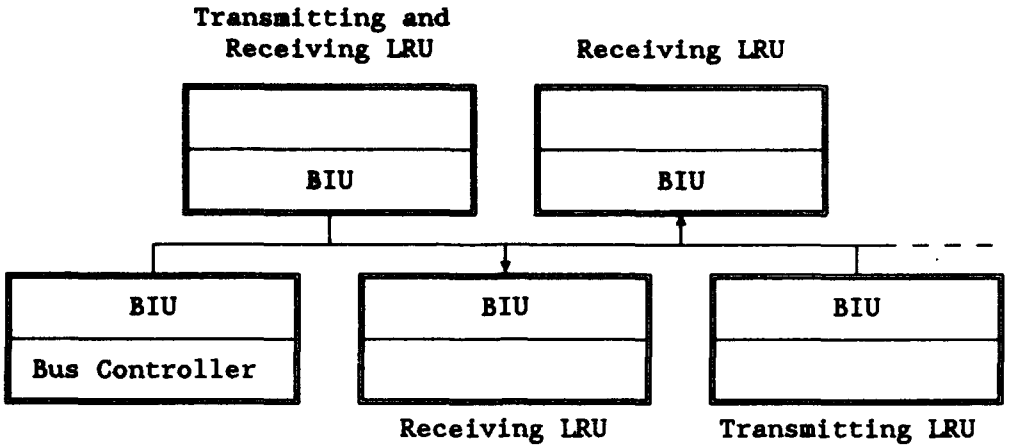


FIGURE 2.1-6. BIDIRECTIONAL BUS ARCHITECTURE, CENTRAL CONTROL

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.

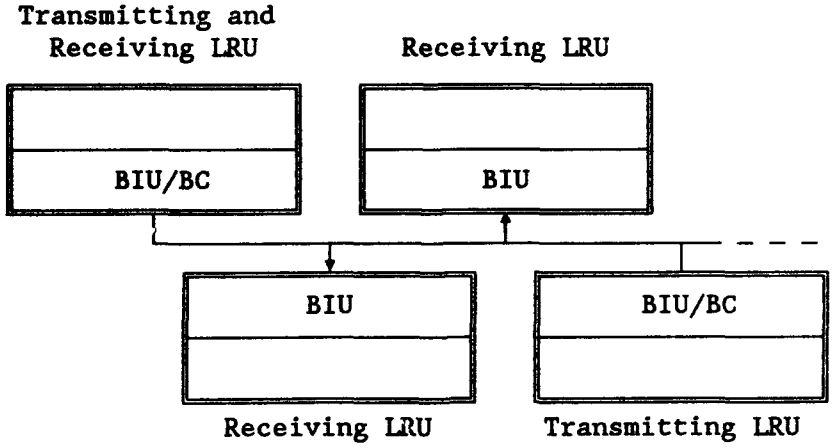


FIGURE 2.1-7. BIDIRECTIONAL BUS ARCHITECTURE, DISTRIBUTED CONTROL

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 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 DATAC was retrofitted to the NASA TSRV 737 CSDB has been used in retrofits	Clifton Shaw and Sutcliffe 1988 Holmes 1986 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.

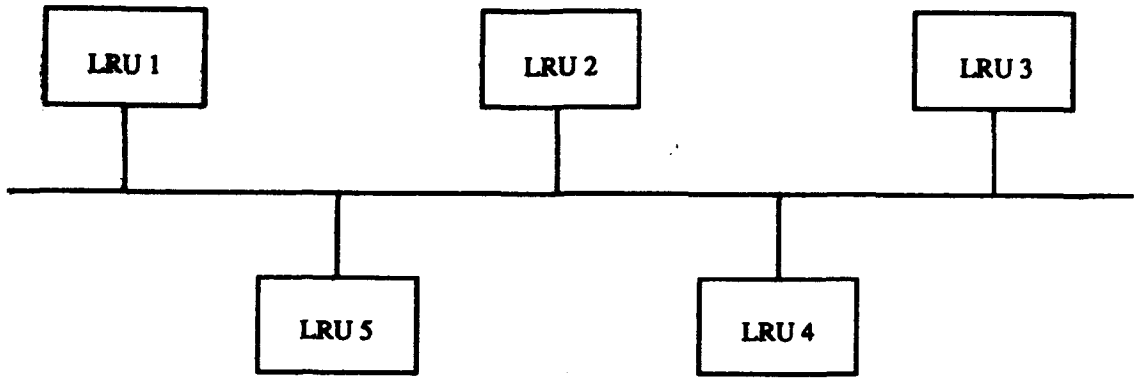
Computer networks are designed for purposes such as database access, integrated voice and data transmission, resource sharing, file transfer, process control, and general communication. On the other hand, avionic data buses were viewed only as a way to save wiring and weight and enhance system performance 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.

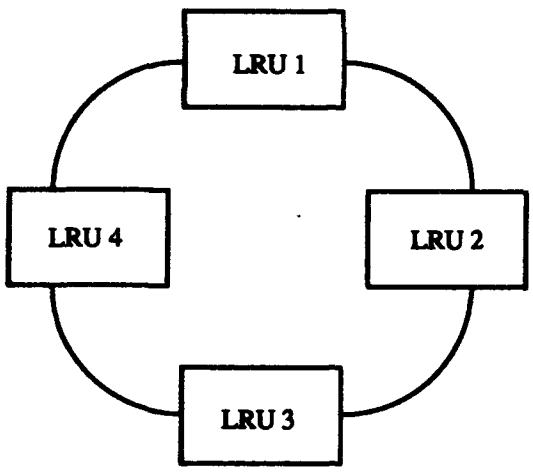
5.1.1.1 Basic Bus Architectures

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.

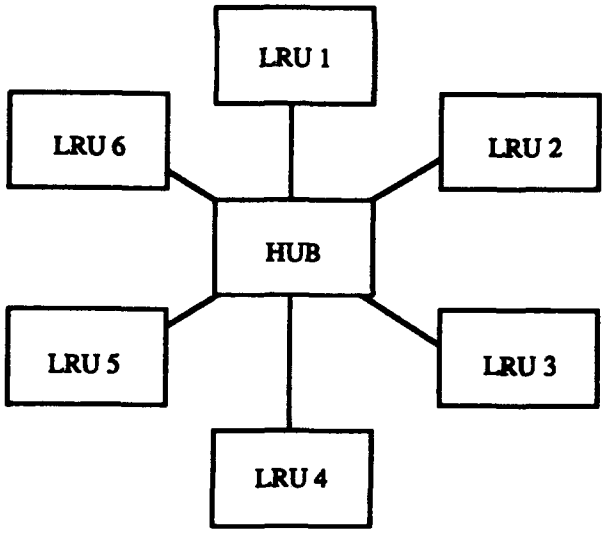
Some of the possible topologies for connecting one data bus to another are shown in figure 5.1-2. When one data bus is controlled by another it is called a hierarchical topology. This is common in a military data bus topology, which uses the MIL-STD-1553 bus. In civilian aircraft, it is common for buses 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

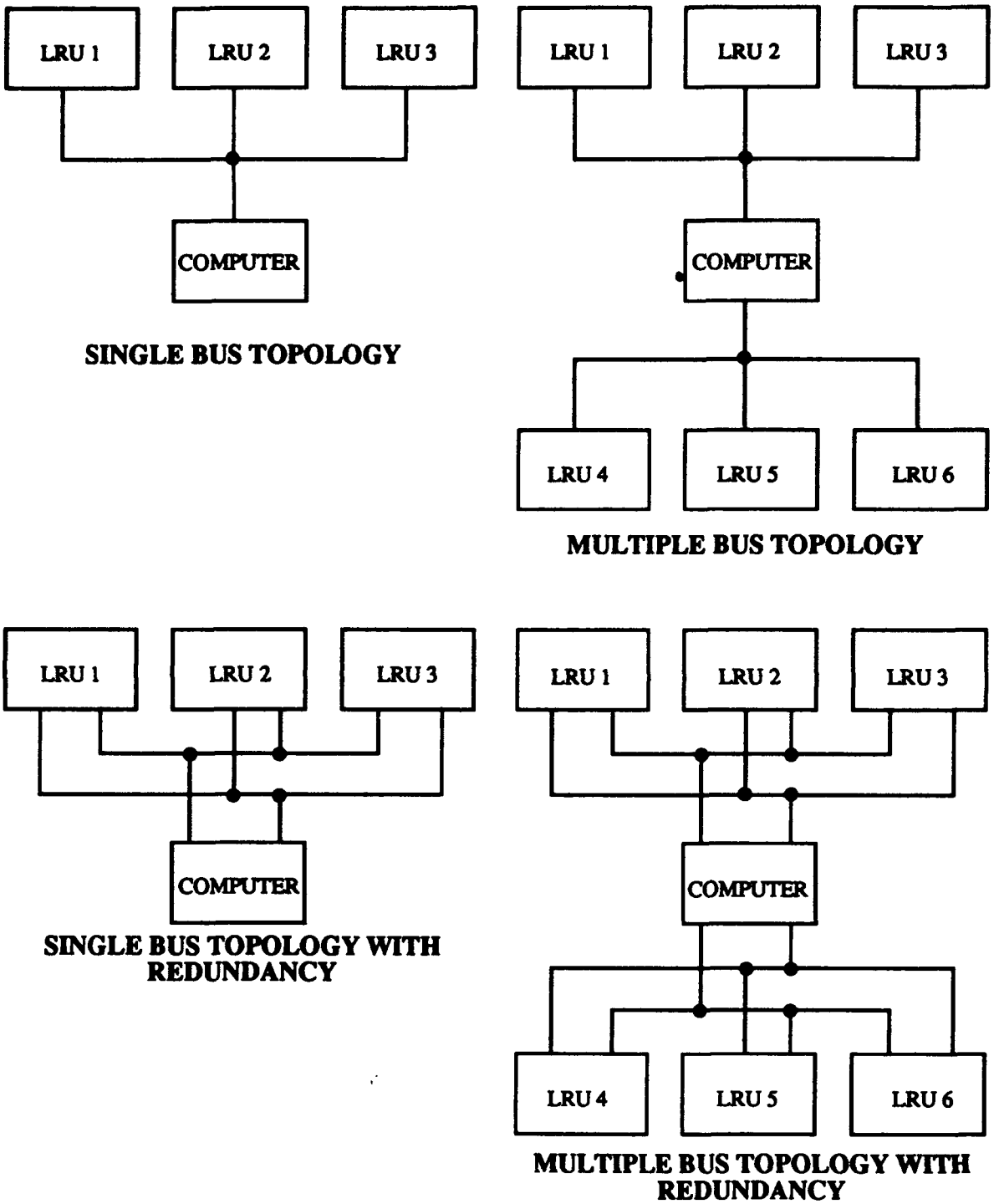


FIGURE 5.1-2. LINEAR DATA BUS TOPOLOGIES

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.

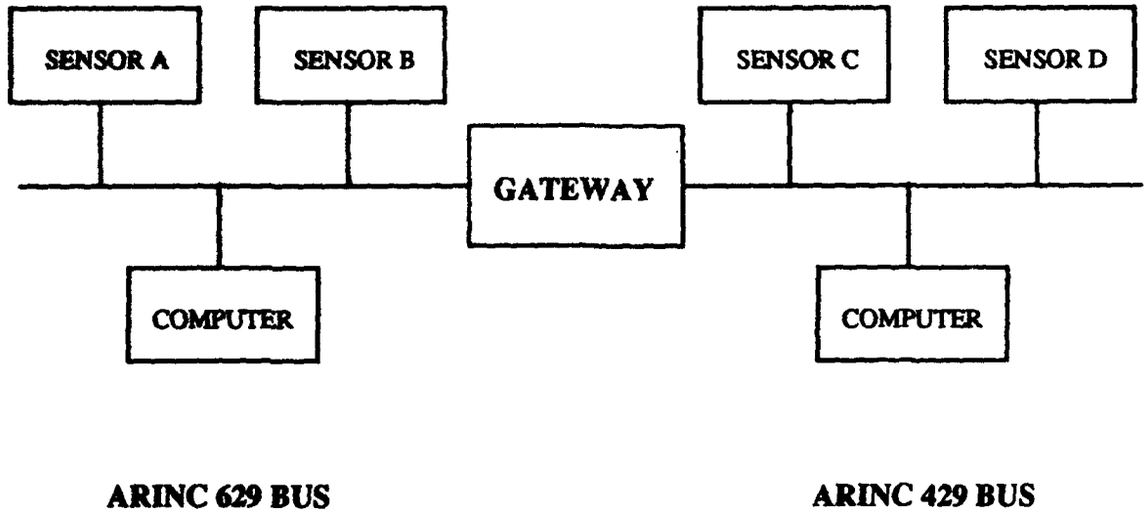
One possible implementation of a gateway may require a conversion from parity error detection to a CRC detection technique requiring the generation of a CRC check word. A gateway may implement this conversion in hardware or software and will have a throughput delay based on the particular implementation 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.



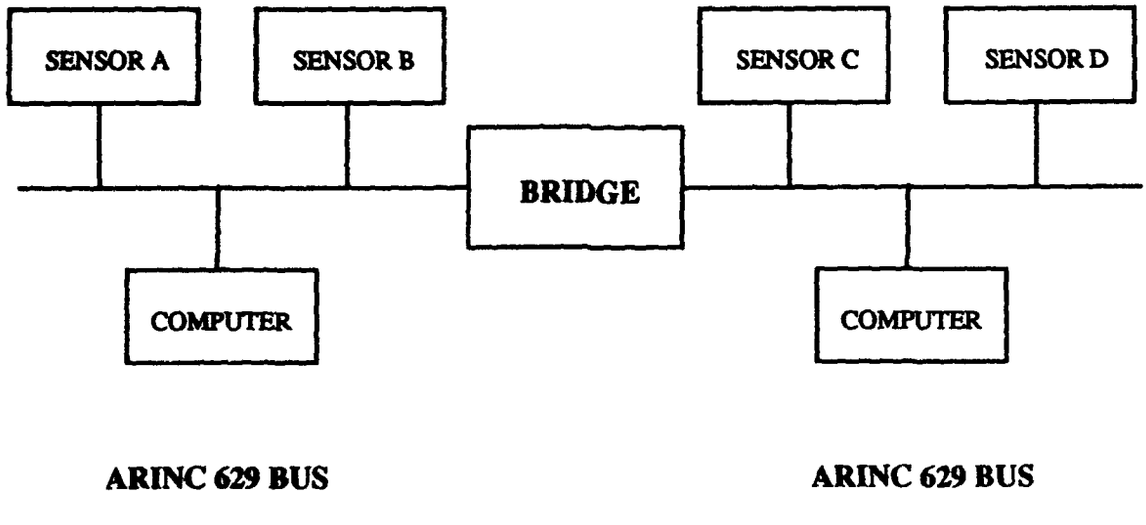


FIGURE 5.1-3. GATEWAY AND BRIDGE USED IN AVIONIC SYSTEMS

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:

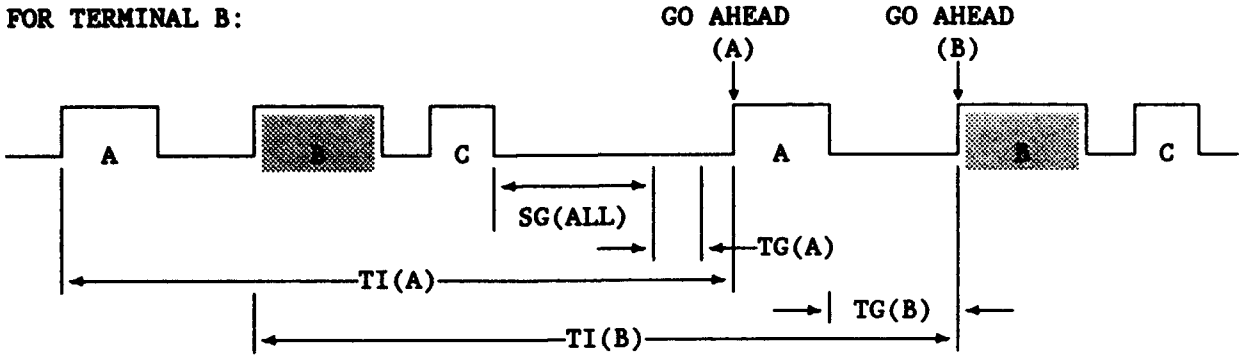
- Level 1 is periodic data (highest priority)
- Level 2 is aperiodic data (mid-priority)
- Level 3 is aperiodic data (lowest priority)

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

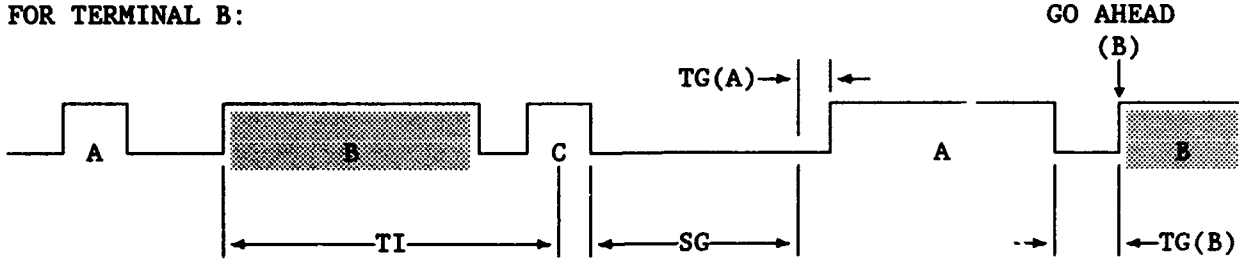
any bus activity. Once all three timers have expired for a user, it may access the bus.



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.

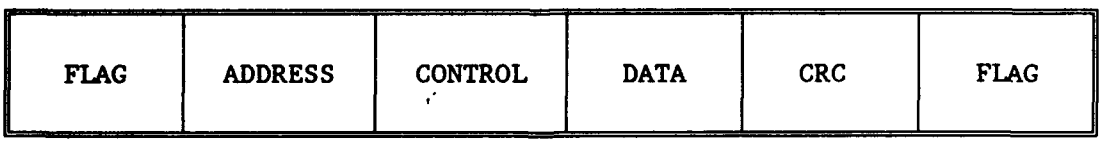


FIGURE 5.1-6. HDLC FRAME FORMAT

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.

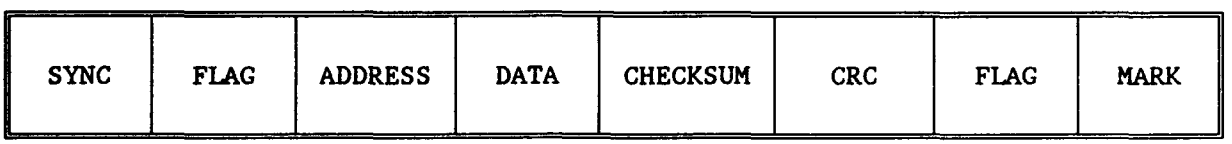


FIGURE 5.1-7. ASCB FRAME FORMAT

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 (μs)	RRT (μs)	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.

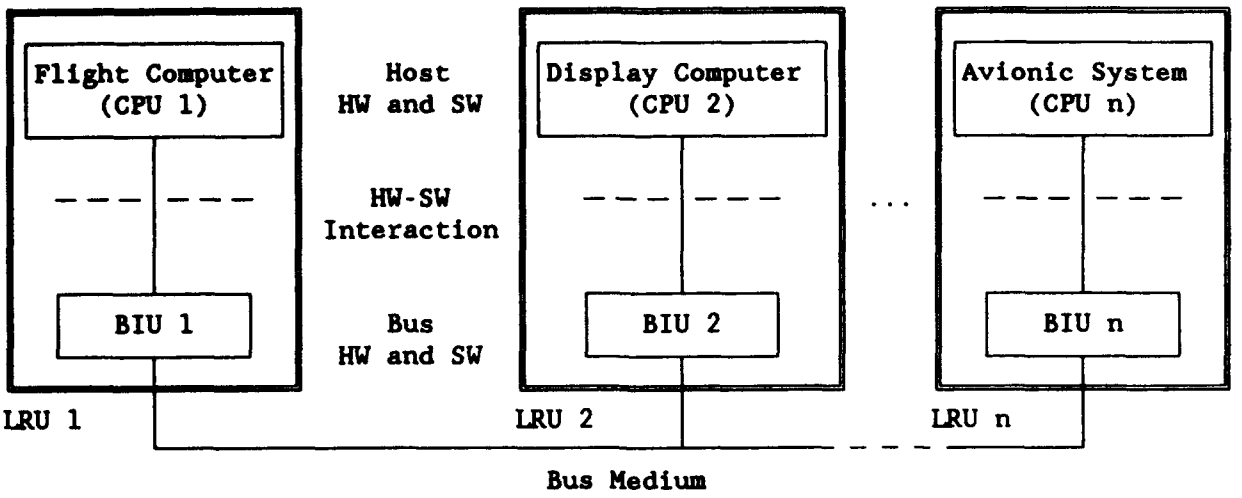


FIGURE 5.2-1. DATA BUS HARDWARE-SOFTWARE INTERFACE

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.

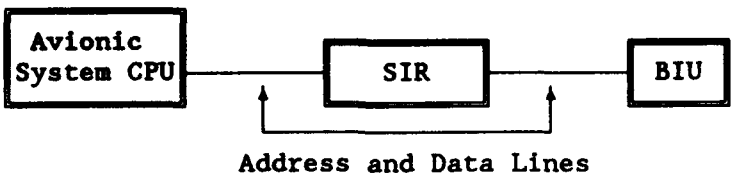


FIGURE 5.2-2. SHARED INTERFACE RAM

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.

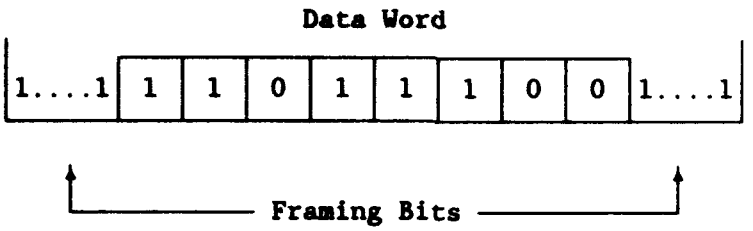


FIGURE 5.2-3. DATA FRAMING

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.

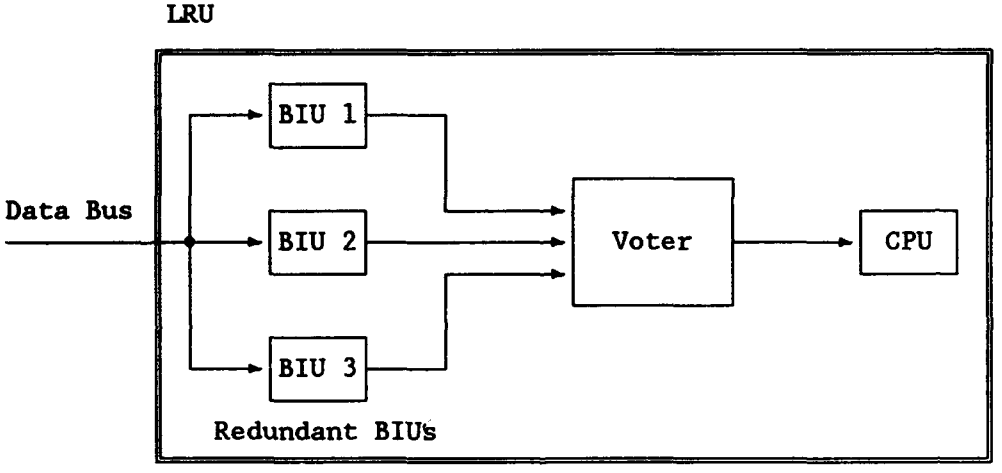


FIGURE 5.2-4. INPUT VOTING

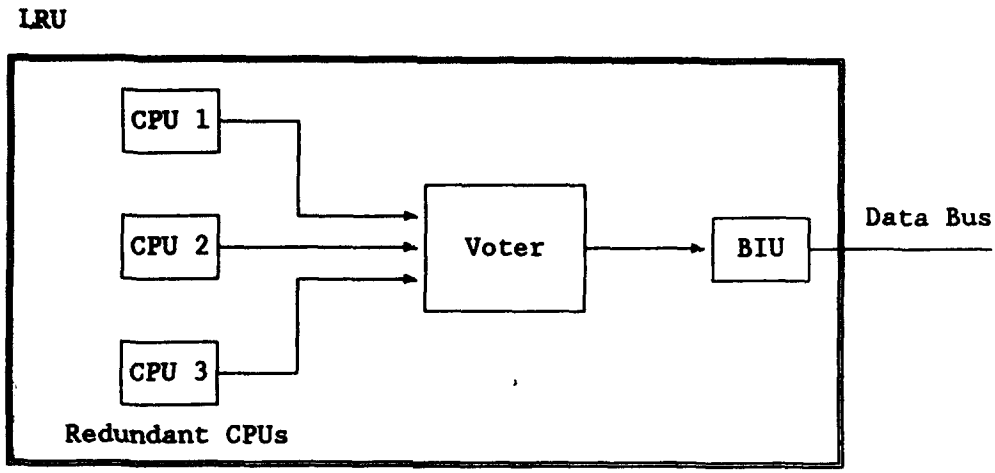


FIGURE 5.2-5. OUTPUT VOTING

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).

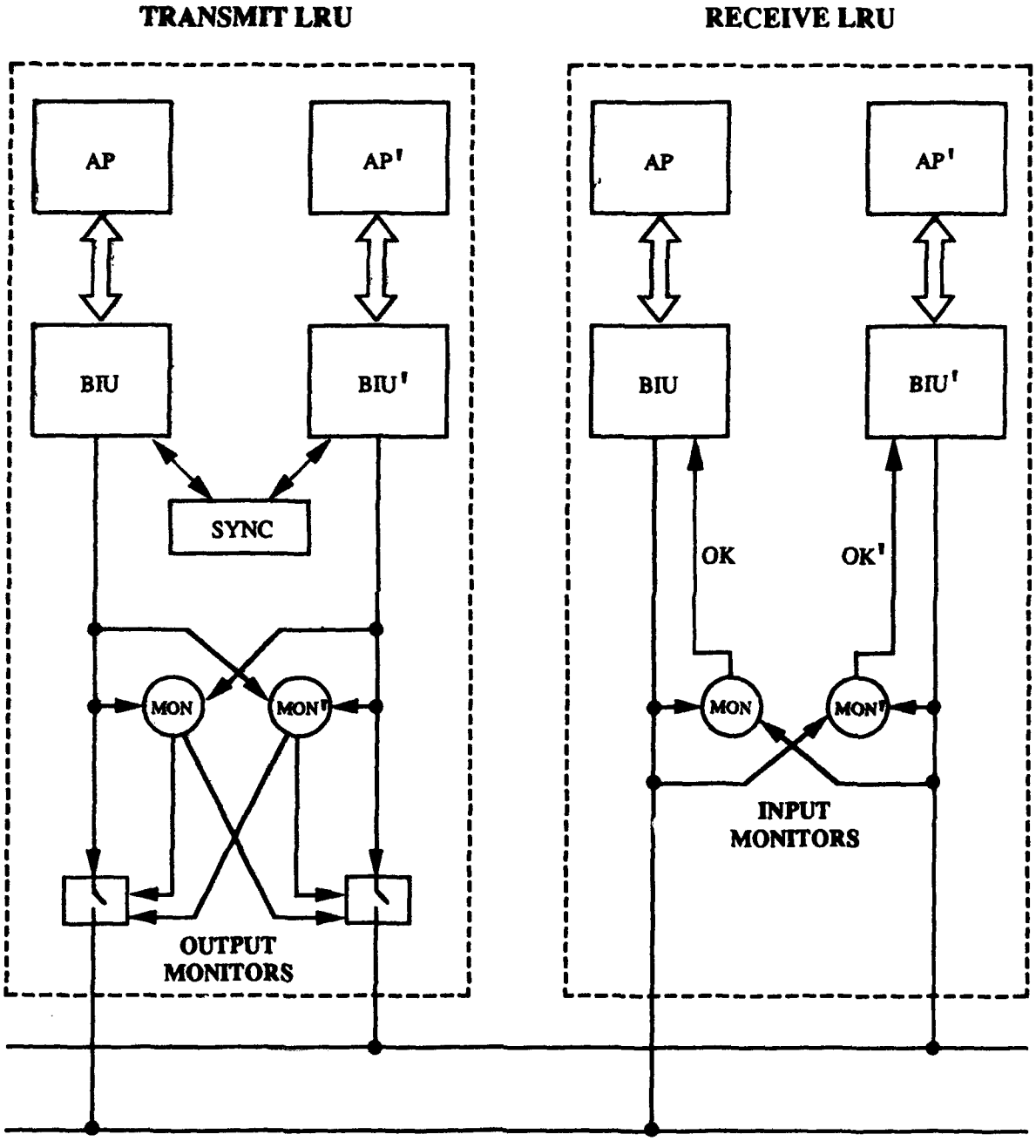


FIGURE 5.2-6. SELF-CHECKING PAIRS
(Driscoll 1983)

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.

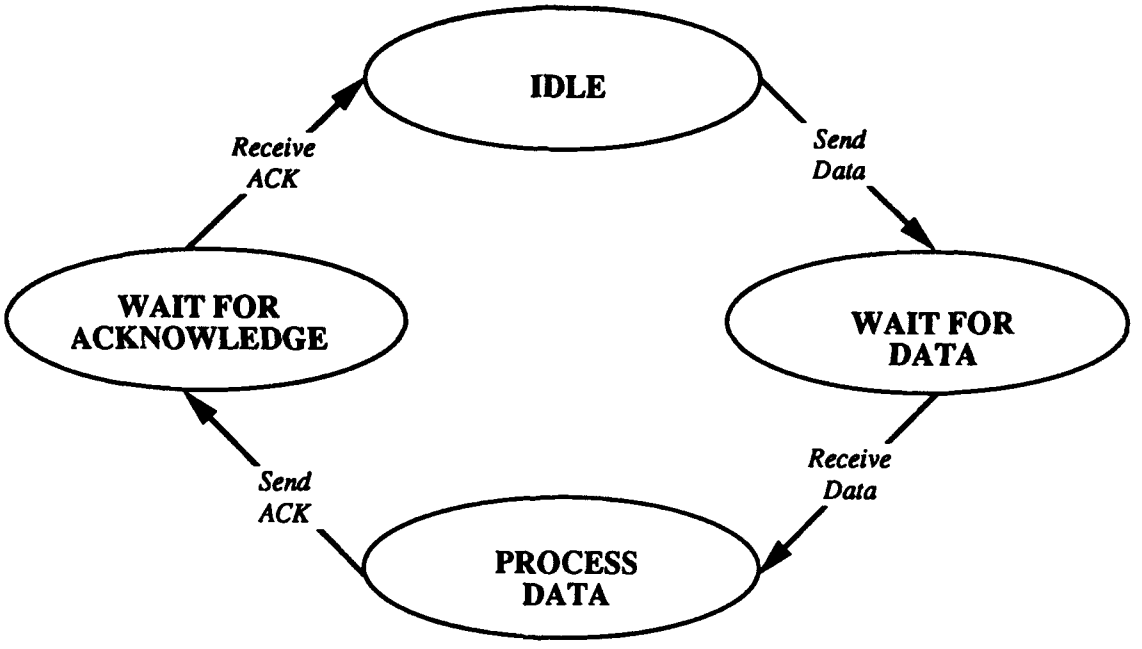


FIGURE 5.3-1. STATE MACHINE
(Merlin 1979)

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."

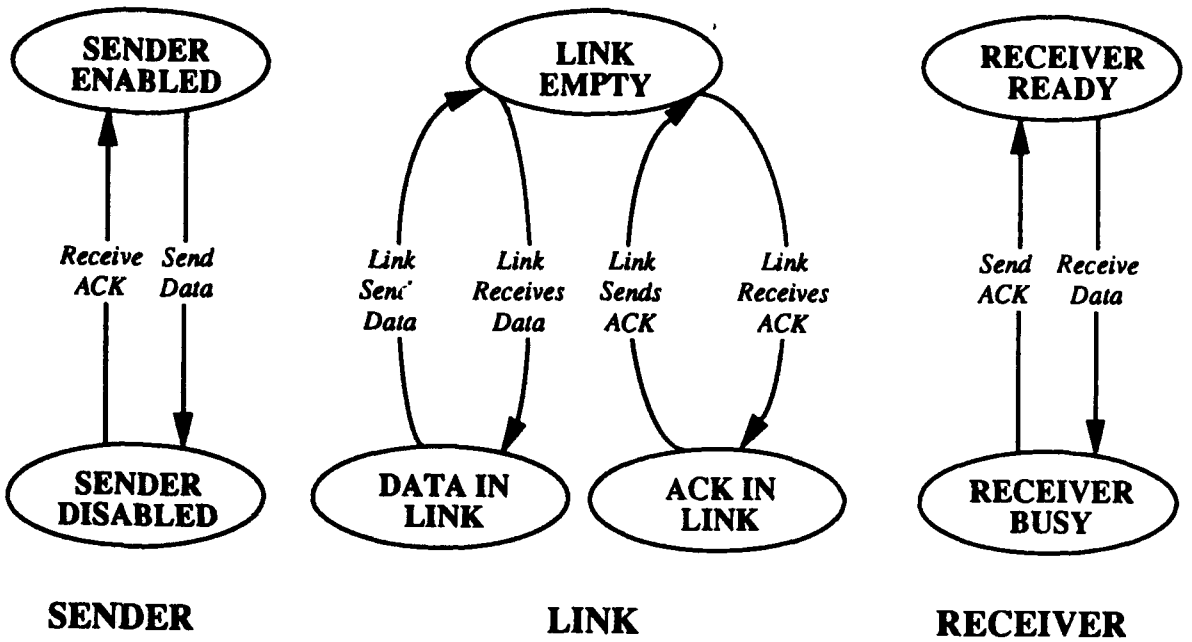


FIGURE 5.3-2. COUPLED STATE MACHINES
(Merlin 1979)

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.

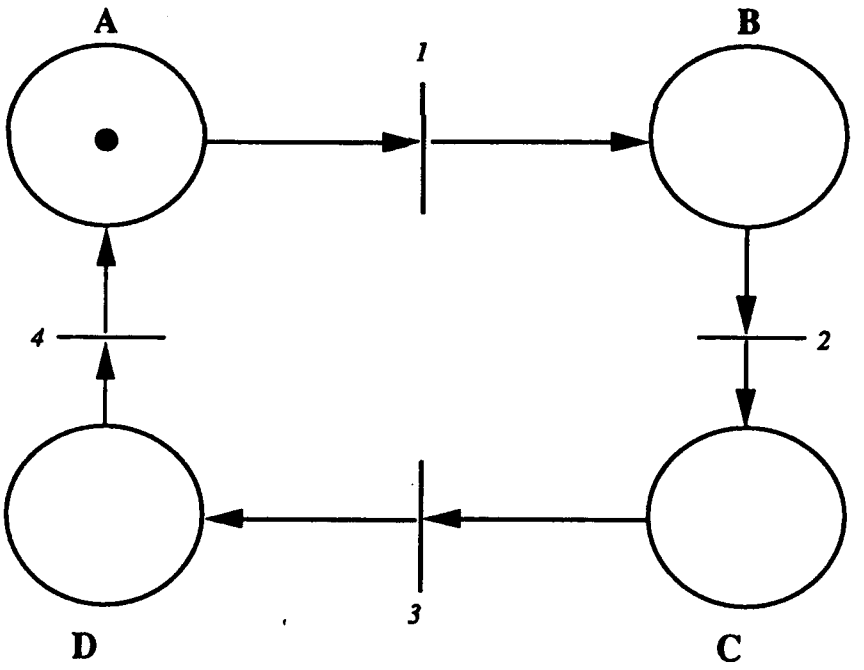


FIGURE 5.3-3. PETRI NET WITH FOUR STATES AND FOUR TRANSITION BARS

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.

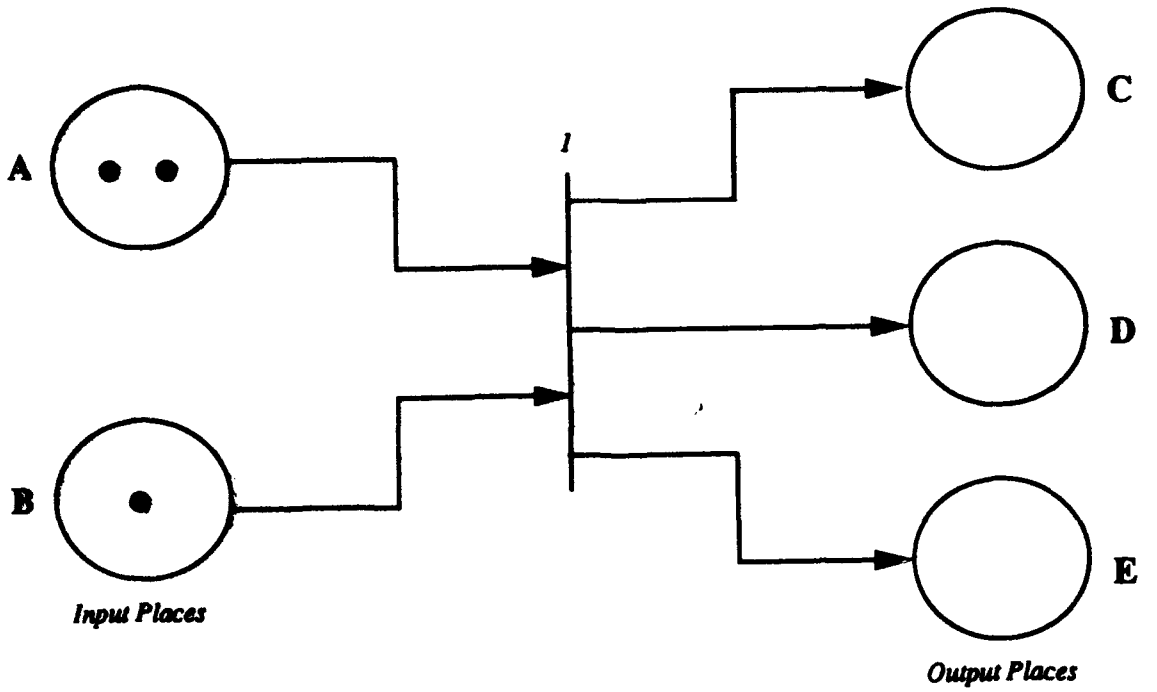
At the lower levels, the access protocol is implemented in hardware. The protocol at this level may be analyzed by an FSM or a Petri net method. Included in Attachment 7 of the ARINC 629 bus specification is a state diagram of the overall access protocol. The purpose is to give an overview of how a terminal 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.

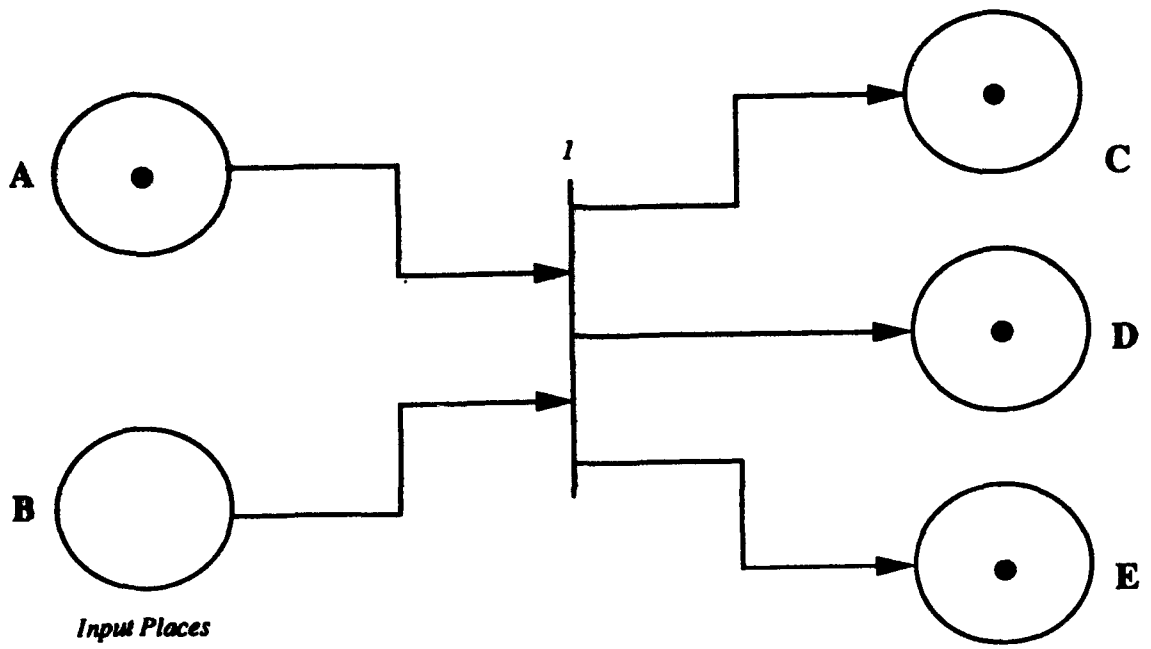
Figure 5.3-4 shows, in a general manner, how a terminal acquires the bus for a transmission when using the CP. The three levels of access and the various timers are explained in section 5.1 of this chapter.

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 3×10^8. 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.

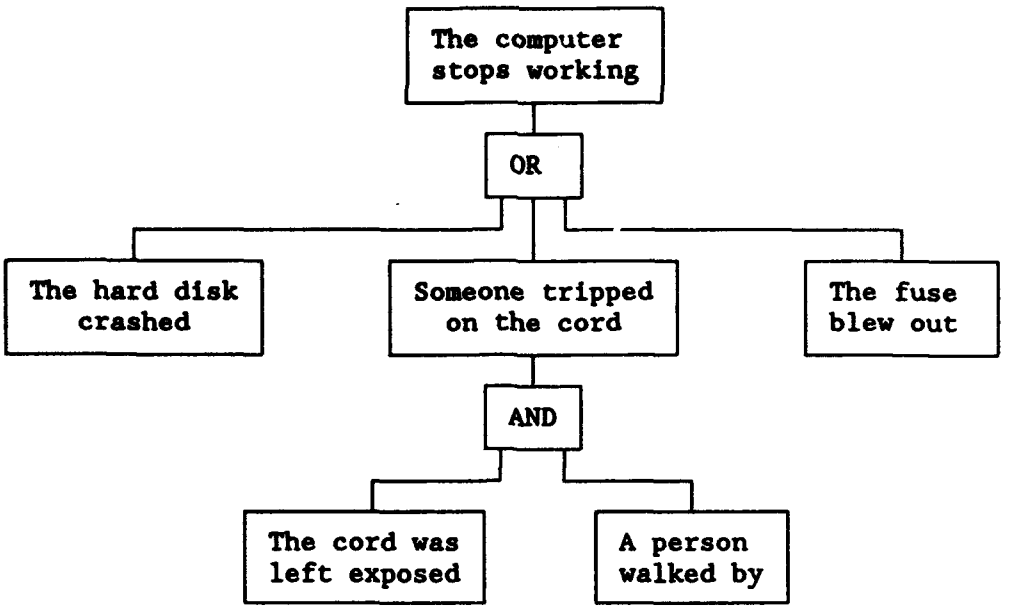


FIGURE 5.4-1. TYPICAL FAULT TREE

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.

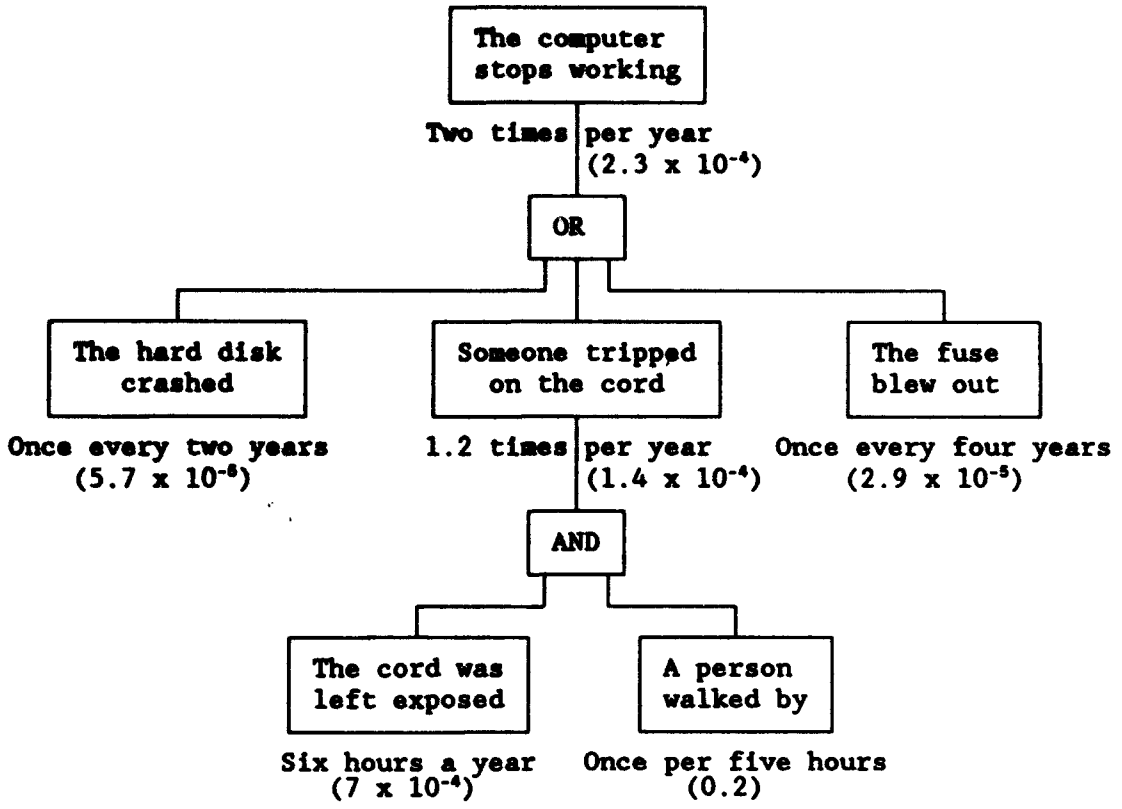


FIGURE 5.4-2. QUANTITATIVE FAULT TREE ANALYSIS

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. 2. Bus impedance is changed.
	Short Circuit	1. Bus transmission disabled until driver timeout. 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 2×10^{-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	1×10^{-3}	Open	90		x
		Short	5	5×10^{-5}	
		Other	5	5×10^{-5}	
B	1×10^{-3}	Open	90		x
		Short	5	5×10^{-5}	
		Other	5	5×10^{-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.

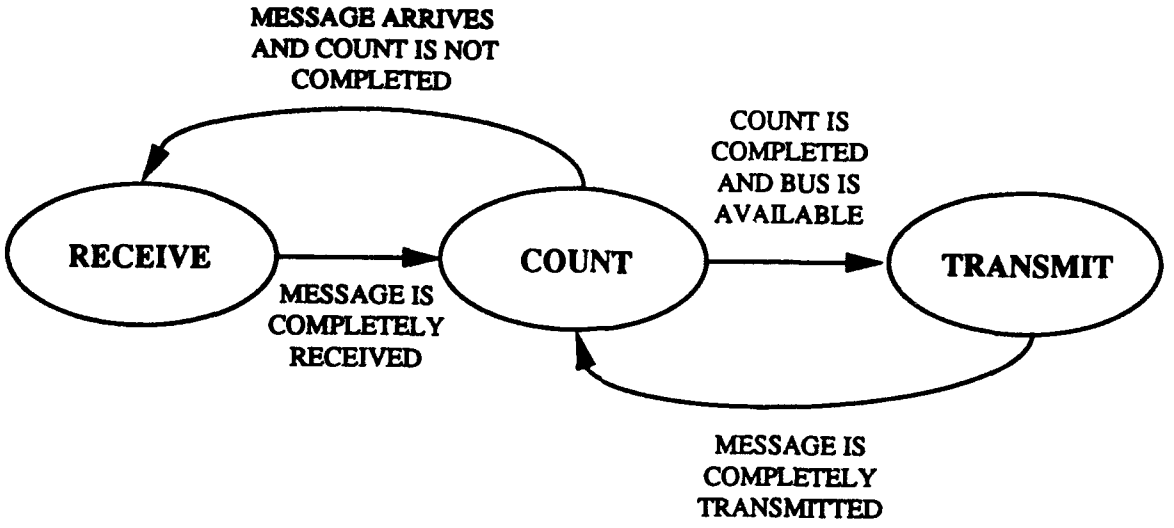


FIGURE A-1. DTSA ACCESS PROTOCOL STATE DIAGRAM

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 $n - m$ if $n > m$ $N + n - m$ 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).

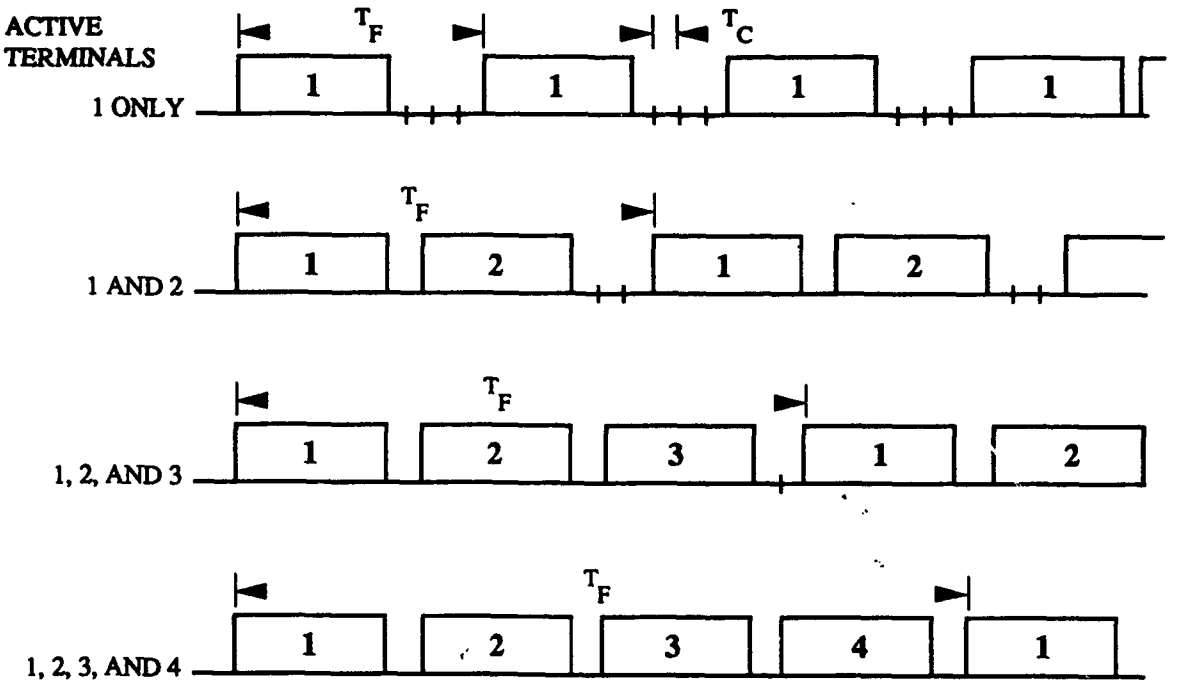


FIGURE A-2. DTSA ACCESS PROTOCOL USER TIMING

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