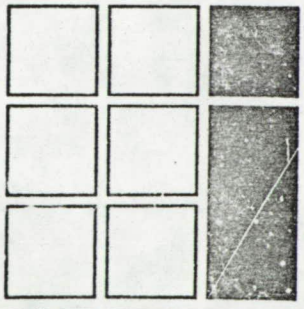


INTERMETRICS

1. NASA CR 115187

CR-115187

2. U/U

Final Report

4. Standard Interface Definition For Avionics Data Bus Systems, FINAL REPORT (16 Dec 70 - 16 MAY 71)

By: Alex L. Kosmala, Joseph A. Saponaro, and John P. Green, Jr.

5. May ~~1971~~

Prepared under Contract NAS 9-11477 by

3.

INTERMETRICS, INC.
380 Green Street
Cambridge, Mass. 02139

Foreword

This document is the final report on the definition of a Standard Interface Unit for avionics data bus systems. The study was sponsored by the Manned Spacecraft Center, Houston, Texas, under Contract NAS 9-11477. It was performed by Intermetrics, Inc., Cambridge, Massachusetts, under the technical direction of Mr. Alex L. Kosmala.

The study program covered the period from 16 December 1970 through May 16 1971. The Technical Monitor for the Manned Spacecraft Center was Mr. Cline W. Frasier.

The publication of this report does not constitute approval by the NASA of the findings or recommendations contained therein.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Background to the Study

Design concepts for the next generation of manned space vehicles have been formulated over the last few years and are currently being evaluated to establish specifications for the vehicles, their subsystems and the operational procedures. Common to all the proposed concepts has been an integrated approach to the avionics system, in which all subsystems communicate with, and are coordinated and controlled by the onboard computer system. A shared data bus has been proposed as the common communication link between subsystems and the computer(s).

This study has been concerned with the major factors that influence the design of a data bus for the avionics system of the proposed NASA Space Shuttle. Although the various design approaches to the Shuttle data bus developed to date have differed in many aspects, all recommend the concept of a multiply redundant data bus, a data bus control unit, and a bus interface unit for connecting the avionics subsystems to the common bus. Designs for the bus interface unit have stressed commonality and standardization, because a standard interface unit is estimated to minimize the number of types and the complexity of hardware and vendor interfaces required for each subsystem. Key issues in the design of the bus system are: the functions and role of the interface unit as a part of the bus system, error detection and recovery, redundancy, and bus control philosophy. Since the interface unit is an integral part of the data bus system and cannot be viewed as a "stand alone" element, its definition and design must be considered with respect to the total approach to the bus system design.

1.2 Objectives of the Study

The central effort has been to identify those factors that form the major design drivers for the bus system, and to define the functional interfaces between the data bus, the bus control unit, and a standard bus interface unit. It was not the objective of this effort to determine the specific requirements for, nor to develop a detailed design of the total data bus system. It was to review already defined requirements and identify those key design features of the bus that affect the nature of the computer-to-subsystem communication and consequently, the specification of the standard interface unit. Although this study analyzed the communications between the computer and standard interface unit, it did not include detailed evaluation of the bus control unit design, nor the detailed design of the standard interface unit-to-subsystem interface. The objective of this study was to analyze the various approaches to an integrated Shuttle avionics organization, and to make recommendations on general characteristics of the data bus system with emphasis on the definition and functional specification of a standard data bus interface unit.

1.3 Approach to the Study

Since the scope of this study did not include a derivation of the primary performance and operational requirements for the Shuttle data bus, initial reviews of Shuttle data requirements and existing studies of proposed data bus designs were undertaken. It was intended to gather information on the various approaches such as design objectives and functional requirements; elements of the bus system and their functions; bus control method; configuration management technique; number of communication paths; command format; error detection and recovery scheme; modulation technique; physical considerations; and redundancy interfacing. Some difficulty was encountered in obtaining this information in sufficient quantity and detail. This was because of the preliminary nature of the studies, the continually changing requirements, and often because the desired information was proprietary in nature.

It was realized early in the study that in order to define the functions of the elements of the data bus it was necessary to evaluate a number of higher level aspects of data bus design. A significant effort was expended in analyzing the problems and possible solutions associated with the following areas:

a) Computer configuration. The communication of data and control over the data bus was found to be greatly influenced by the configuration of the control computer(s) with respect to the data bus.

b) Failure tolerance and reliability. The effect of the Shuttle failure tolerance criterion on the degree of redundancy in the data bus system, the techniques for the detection of failures and eventual reconfiguration of the bus were found to impact the design of the bus elements directly.

c) Data bus management. Several data bus control techniques were analyzed for relevance to the Shuttle. The type of control was found to impact the nature of the computer I/O with the bus, and the design of the standard interface unit.

The definition of a functional specification for a standard interface unit should only be made when a comprehensive list of requirements for it is known. Because of the early stage of development of the Shuttle vehicle concept this level of information was not available. The various designs under consideration at the beginning of this study were based on different (and changing) requirements and ground rules. It was consequently difficult to evaluate several design approaches on a common basis.

In the absence of a specific set of requirements for the Shuttle data bus it became apparent that no single approach to a design stood out as a clear candidate for implementation. A general impression was gained that almost any suggested approach could be made to do the job. Some criteria for evaluation had to be established in order to arrive at a set of specific recommendations. Chapter 2 provides a summary of the basic requirements, ground rules, and assumptions that this study used as a foundation. The evaluation and recommendations presented in Chapter 7 were guided by two basic ground rules, namely: 1) choose the simpler approach, and 2) choose the approach that solves one problem at a time, wherever the requirements or assumptions provided no clear decision path.

Finally, one other conclusion was reached early in the study. It was that the implementational details of a data bus design such as technology, transmission media, modulation techniques, etc. are of significantly less importance than the higher level questions above. This was reflected in the amount of effort apportioned to this aspect of the problem.

1.4 Overview of the Report

This report is divided into seven chapters. Chapters 3, 4, 5, and 6 are presented in no particular logical order or degree of importance of their subject matters.

a) Chapter 2 lists the requirements for the Shuttle data bus system that were adopted for the purpose of this study, and gives reasons for the less substantiated assumptions that were made.

b) Chapter 3 provides a discussion of the impact on data bus complexity and management difficulty of the assumed FO-FO-FS failure criterion. Various configurations for interconnecting computers and bus control unit, buses, and interface units are reviewed.

c) Chapter 4 presents a detailed appraisal of the problems of providing command, control, and data acquisition over a common data bus. Bus access methods, data formats and bus traffic are analyzed.

d) Chapter 5 reviews a variety of error control techniques. Error coding, transmission feedback, retransmission, and voting are included in the discussion.

e) Chapter 6 gives a brief treatment of selected hardware problem areas. Modulation techniques, transmission media, coupling and synchronization are discussed.

f) Chapter 7 reviews the material of the previous chapters and makes specific recommendations in the area of bus control, and the functions of a standard interface unit.

1.5 Summary of Recommendations

The recommendations fall into two categories:

a) those associated with the general problems of command, control, and data acquisition by a common data bus in an integrated avionics system;

b) the definition of the functional organization of a standard data bus interface unit.

The major points are very briefly summarized in the following paragraphs. For definitions of the terms used, and for further clarification of the recommendations, reference should be made to chapters 2 and 7.

1.5.1 Bus Control Summary Recommendations

a) One authority, i.e. the computer/BCU, should control all commands and data acquisition via the data bus. Any other computer must be interfaced to the bus via a standard interface unit, and must be regarded as any other subsystem.

b) The computer should initiate and control all bus communications. Control should be by the command/response address technique. No remote terminal may determine its own need to access the bus. Each terminal will respond only to the computer/BCU. A possible waiver of this rule may be made in the case of recording, telemetry, or display equipment, but a non-standard interface unit is then required.

c) The following error control transmission policy is recommended:

   1) path verification via feedback transmission by each SIU of at least its address bits upon being accessed by the BCU. No verification of echo check by the BCU is recommended prior to the release of the data to the LRU by the terminal.

   2) Message verification at the terminal and the BCU by horizontal and vertical parity bits. Message verification at the terminal is required before feedback of address echo.

   3) Error correction by re-transmission or re-request of message by the BCU.

   4) Higher security for "critical" commands to be achieved outside of the bus by software controlled multiple transmission of message; successful receipt to be determined by the subsystem.

d) A byte-serial data transmission is recommended. The data byte size should be a submultiple of the computer's memory word/byte length. The bus command byte should be determined by the nature of the bus traffic, when more exactly known.

e) A variable length message format with a limit of 32 bytes is recommended. A 2-bit field in the control format is sufficient to specify a 1,4, 8 or 32 byte data block.

f) A "busy" indication by the terminal is recommended to allow for terminal and LRU latency, and as a mechanism for expanded terminal capabilities.

1.5.2 Recommendation of Standard Interface Unit Organization

a) The communication of data to and from the subsystem should be the interface unit's prime design consideration. Additional functions may be added later, but must abide by the constraints of the bus control structure and data formats.

b) A standard interface with the bus is recommended for all terminals. For the subsystem interface a fixed maximum of 512 electronic interface channels is recommended, in order to size the channel address field. A maximum of 16 channels each of analog, parallel digital and serial digital input and output signals is suggested as a standard electronic interface, with modular expansion from 0 to 16 to suit the requirements of a particular equipment. An "invalid channel address" signal is recommended to indicate a less than maximum interface implementation. (This standard interface cannot be made a recommendation of the study, since specific equipment requirements have not been determined to date.)

c) The 512 channels could be assigned differently than above, but a non-standard electronic interface would be the result. A modular internal structure could alleviate the problem of terminal diversity.

d) A unified terminal with combined SIU and EIU facilities is recommended, although a separate input channel for each bus line should be provided to the point of address comparison.

e) Redundancy interfacing should be performed at the bus-to-terminal interface by special line coupling elements, which provide 4-to-many and many-to-4 interfacing.

f) Line conditioning, signal amplification and noise discrimination should be provided. They may be conveniently accommodated in the line coupler.

Chapter 2

Space Shuttle Data Bus Definitions and Ground Rules

2.1 Introduction

The definition of a standard interface unit for the Shuttle data bus depends directly on, and is constrained by, such factors as: the avionics configuration requirements, avionics subsystem and equipment data requirements, redundancy techniques and centralization versus decentralization of functions. The objective and the scope of this study did not include a detailed analysis of the avionics system requirements. These were assumed to be as formulated and derived by others.

2.2 Data Bus Definition and Terminology

2.2.1 Definition

The space Shuttle data bus system is the principal medium of communication between components of the integrated avionic system and the computer complex. It is a multiply redundant, common, shared interface to flight electronics and mechanical equipment providing remote acquisition and distribution of commands, data and other information. The Shuttle data bus system illustrated in Figure 2.1 is composed of three major elements:

a) a bus control unit,

b) a redundant set of transmission lines, and

c) a number of remote terminals.

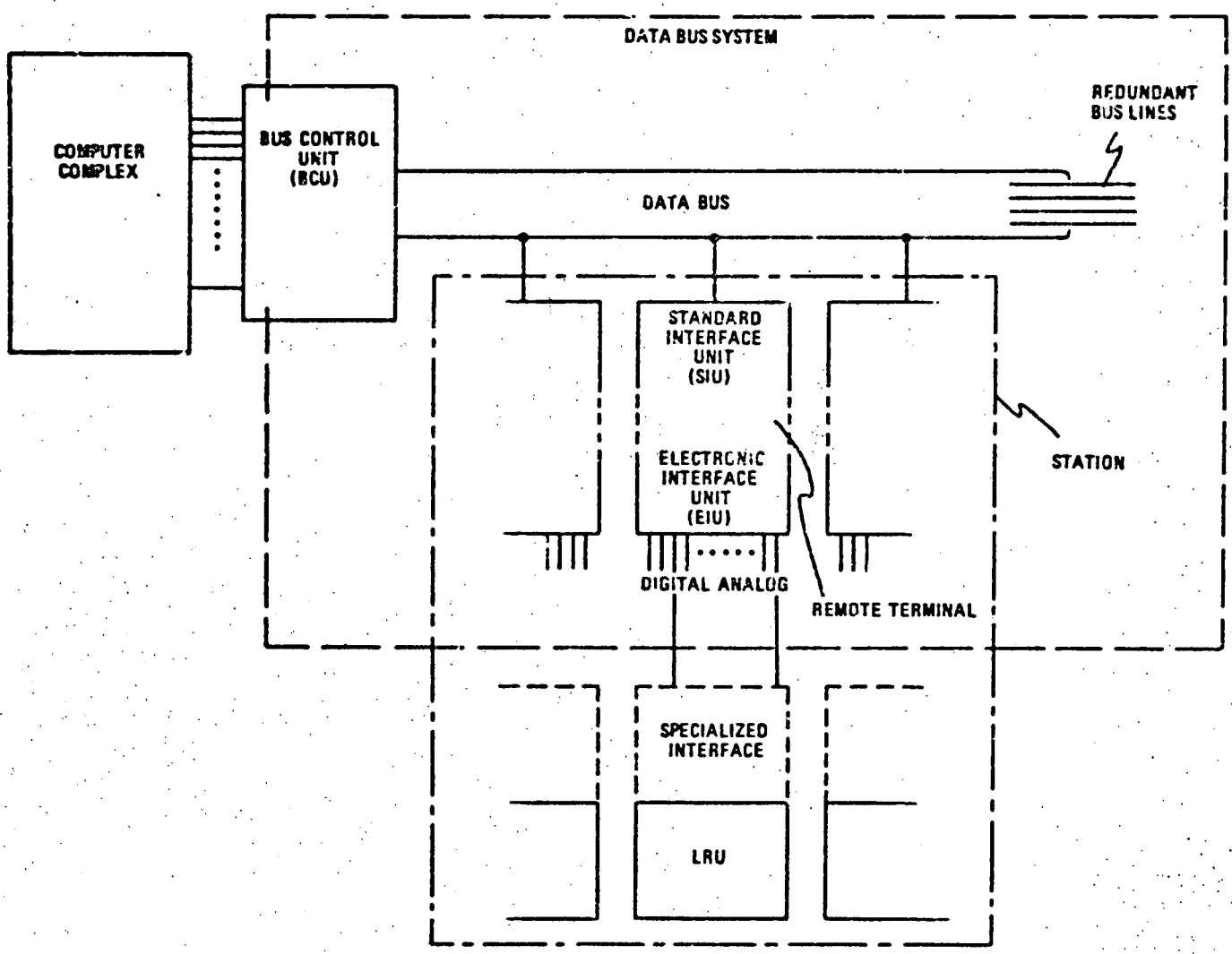


Figure 2.1 Data bus system elements and terminology

2.2.2 Terminology

The nomenclature utilized throughout the report attempts to utilize, as much as possible, existing terminology and acronyms rather than defining new ones. However, to avoid confusion and preconceived associations the following definitions are provided.

a) The data bus. The data bus is composed of a set of redundant bus lines. Each bus line is a single communication channel, capable of two-way transmission of serial digital information between several remote terminals and the bus control unit. Physically a bus line may be a single coaxial cable, or balanced line such as a twisted shielded pair, on which information is transmitted by time or frequency division multiplexing.

b) BCU (Bus Control Unit). The BCU is the control element of the data bus system. It provides the primary interface of the bus system to the computer complex. It is the primary I/O peripheral to the computer complex, containing a parallel interface with control and shared access to the computer memory. The functions of the BCU are discussed in more detail in Chapters 4 and 7.

c) Terminal. The terminal is a remote unit which interfaces between a bus line and a remote avionics equipment. A terminal is addressable by the computer/BCU for the input or output of data to the equipment. It consists of two basic elements: a standard interface unit (SIU) and an electronic interface unit (EIU), which may or may not be physically separated.

d) Standard Interface Unit (SIU). The SIU is that part of a terminal associated with the functions necessary to interface with the bus. It contains line termination, signal modulation and demodulation, transmitter and receiver control, and terminal address decoding logic.

e) Electronic Interface Unit (EIU). The EIU is that part of a terminal associated with the functions necessary to interface avionics equipment. The EIU inputs and outputs information to a number of analog and digital I/O interfaces in response to I/O commands. Data and commands received or acquired by the EIU are routed through the SIU portion of the terminal for communication on the bus.

f) Lowest Replaceable Unit (LRU). In this study LRU is defined as the smallest piece of avionics equipment recognized and addressed by the bus system. More than one LRU may be connected to an EIU. The following could all be categorized as LRU's: a single beacon, a VHF transceiver, an inertial measuring unit, a remote processing computer.

g) Station. A station is defined as a collection of terminals associated with a functional avionics subsystem.

h) Subsystem. A subsystem is defined as a collection of LRU's which constitutes a function recognized within the integrated avionics hierarchy. Examples of subsystems are the reaction control system (RCS), the electrical power system (EPS), the environmental control system (ECS), the inertial subsystem. A subsystem may be geographically distributed about the vehicle.

2.3 Review of Shuttle Data Bus Requirements and Ground Rul.

Several organizations, including the Phase B contractors, have conducted analyses of data bus and other Shuttle system requirements. The results of many of these studies have been published for the NASA Manned Spacecraft Center. A review of the information made available during the course of this study was conducted. Several problems arose in obtaining a common set of data and communication requirements for the bus, principally due to the continually changing nature of Shuttle operational requirements, and the differences in system design approaches and objectives. A summary of the major requirements of the Shuttle obtained from this review are listed below. Although there are many detailed system requirements for the Shuttle avionics system, only those pertinent to the functional specification of interface unit were used.

a) The study has assumed that the Shuttle data bus system meets the failure tolerance requirement specified for all electronic subsystems; namely, that it shall "fail operational" after the failure of two most critical components, and "fail safe" after the third failure. Accordingly, the failure tolerance specification has been interpreted as requiring quadruply redundant bus lines. A more detailed interpretation of this failure tolerance requirement is presented in Chapter 3.

   Although this failure tolerance requirement has been assumed and not analyzed or justified in detail, it is clearly of significant impact to the organization of the bus elements, in particular the remote terminal. The necessity of interconnecting an avionic subsystem of m-level redundancy with a quad redundant bus has been a key design driver in formulating the SIU requirements.

b) The concept of a central shared data bus with standard remote interfaces to avionics equipment is assumed to be the most

cost effective concept for both the Shuttle orbiter and booster vehicles.

c) The primary function of the data bus system is to provide a communication path between the avionics equipment and the prime computer complex. No general requirement for terminal to terminal communication which cannot, or need not be routed through the computer complex has been identified.

d) Subsystem interfaces. The data bus must provide a capability of interfacing to redundant electronic subsystems. The exact number and type of such subsystems has been changing as the operational requirements evolve. The representative list provided below was assumed to indicate the scale of the system.

1) Primary propulsion subsystem: this system consists of two orbital insertion engines and one orbital maneuvering engine.

2) Reaction control subsystem: at least 20 RCS jets located in the nose, wings and tail for effecting rotation and translation in space.

3) Hydraulic system: hydraulic power generation, distribution, control, and conversion of mechanical energy. It consists of supply lines, gimbals, pumps, aerodynamic surfaces, flaps, wheel controls, etc.

4) Electrical power generation and distribution system: fuel cells and battery, and the auxiliary power units located throughout the Shuttle.

5) Navigation aids/air data: a collection of equipment providing navigation and landing capabilities (ALS, radar altimeter, TACAN, DME, etc.).

6) Environmental control system: the environmental control system provides temperature, pressure, and humidity control of equipment, equipment bays, and personnel compartments.

7) Cryogenic system: contains the hydrogen and oxygen for the primary propulsion, the reaction control system, the fuel cells and the auxiliary power units.

8) Displays and controls: this system is assumed to have local processing capability and accepts dynamic data through the bus for updating of display parameters.

9) Telecommunication: this system consists of various transmitters and receivers including S-band, C-band, VHF, telemetry encoder, EVA communications, air traffic control communications, etc.

10) Guidance, navigation and control: this subsystem is composed of elements necessary to control, stabilize and navigate the Shuttle vehicle during all phases of the mission. It interfaces to the reaction control system, jet engines, aerodynamic control surfaces, and landing gear, etc. It has access to sensors which include the inertial subsystem, horizon and star trackers, approach landing aids, rendezvous radar, radar altimeter, etc.

Although this list of subsystems may not be complete for the final organization of the avionics system it is meant to be representative. It is estimated that approximately 150 to 250 LRU's are associated with the subsystems listed above.

e) Data requirements. The following is a summary of the data requirements abstracted from the various studies of Phase B contractors.

1) Speed. Peak load estimates of data rate for both the Shuttle and orbiter have ranged between 100,000 and 250,000 bits per second, including overhead. Considering an average overhead of approximately 50% for each bus transaction and allowing for a minimum of 100% expansion to the maximum speed, a capability of 10^6 bits per second has been assumed to be an adequate requirement. This speed should allow the computer to acquire data at a rate of approximately 10,000 average transactions per second.

2) Measurements. Estimates have ranged between 4000 and 6500 unique data points to be sampled from the total complement of avionics equipment by the central computer. Data types include:

digital parallel
digital serial
analog
discrete

The majority of these data points are measurements input to the computer, and are estimated at approximately 60% to 70% of the traffic on the data bus.

3) Response time/sampling frequency. The maximum sampling frequency of measurements is estimated at fifty samples per second. The average sampling frequency for status information is between 2 and 5 samples per second. Very little information was made available on response requirements and load distribution of subsystems.

4) Number of terminals. The number of terminals estimated varies considerably depending upon the degree of redundancy, interfacing policy and the design of the terminal. The number of independently addressable terminals is assumed to be somewhere between 50 and 200.

f) Physical requirements. Each bus line was assumed to be physically separated aboard the vehicle for reasons of reliability. It was assumed that bus lines will be run down each side of the vehicle, and that the bus will be capable of transmitting over distances of 300 to 500 feet. It is assumed that the equipment will be located in several equipment bays located throughout the vehicle. Terminals must be capable of being separated from the bus by distances of up to 50 feet.

PRECEDING PAGE BLANK NOT FILMED

Chapter 3

Reliability, Redundancy and Organization

3.1 Definition of Failure Tolerance

Before a detailed evaluation of the Shuttle data bus can be undertaken it is important to consider the requirement for failure tolerance, since it is this factor that introduces the greatest complexity into an integrated Shuttle avionics system. As usually stated, the avionics system must remain fully operational after the first and second failures, and must fail in a safe manner after the third. In a practical system failure tolerance implies that each major element in the system must possess internal functional redundancy, and a highly effective technique for failure detection to allow quick reconfiguration in the event of a failure.

The high level of redundancy that is required for a multiple failure criterion allows application of voting and comparison techniques to systems which generate output data; for example, the computer, and sensors such as the IMU, radar, pressure and temperature transducers. Voting of passive elements such as actuators requires the feedback of information which indicates the element's response to the command.

The penalty that must be paid for voting as an approach to failure protection is that all redundant copies of a given piece of equipment must be powered up, functional, and operating identically.

The level of redundancy and the technique of failure detection depend on the interpretation of the failure criterion. The greatest difficulty attaches to the definition of the "failed safe" condition. Two interpretations are possible:

a) To treat the "failed safe" conditions as "graceful degradation". In this concept the failure results in a reduced system capability which, nevertheless, retains

certain functions critical to the safety of the crew and the vehicle. The security of these functions must, therefore, be treated as part of the system design specification. This approach allows the greatest economy of equipment, but suffers from a difficulty of definition and of sophistication of system design, especially in the area of software.

b) To treate the "failed safe" condition as "operational". This obviates the need to specify a diminished set of critical functions, and avoids the difficulty of their implementation. It suffers, however, from the need for full redundancy of equipment to allow performance in an undegraded fashion after the failure to the "safe" condition.

The second approach has been assumed during this study, because the detailed definitions required for graceful degradation cannot be undertaken at this early stage of Shuttle development.

A clarification of the "operational" condition prior to the "failed safe" is necessary to establish the degree of redundancy required by the second definition above.

a) It may be defined as a fully operational condition in which there will be a 100% certainty of failure detection, with a near-instantaneous reconfiguration.

b) It may be less strictly interpreted, as a fully operational state with a small but finite probability that certain failure modes may pass undetected, or remain unresolved, and that a small, but finite time may be required to recover from a failure transient.

The first interpretation virtually demands that sufficient redundancy among the unfailed elements remains for majority voting to take place even in the penultimate failure state. Majority voting provides almost perfect error detection when errors occur relatively infrequently in an uncorrelated random fashion. An added attraction is the capability for immediate error correction upon determination of the dissenting vote. The penalty is that at least triple redundancy is required prior to the failure to the safe condition. For the full FO-FO-FS tolerance this implies that five levels of redundancy must initially be available.

The second interpretation above allows the requirements of the failure detection technique to be relaxed, and a lower degree of redundancy to be used. At least dual redundancy is required by the "operational" definition of the "failed safe" state, and allows comparison to be used to trap the final failure. For the

failure characteristics described above comparison provides for almost certain detection. Consequently, FO-FO-FS tolerance can be provided by four levels of redundancy, rather than five. Comparison, however, does not provide an indication of which of the redundant signals or equipment has failed. A supplementary method must be employed to identify and isolate the failure:

a) Self-diagnosis. In the case of computers, a degree of assurance can be provided by special self-test software which exercises most of the basic operations in the processors, memory, and I/O. However, the volume and complexity of the diagnostic routines required for near certain probability of fault isolation precludes their use in parallel with the operational software. These routines must replace the operational software for as long as it takes to track down the ailing element. During this period the computer can obviously not provide the full complement of capabilities to the system.

   For the less "intelligent" systems in the avionics less extensive diagnostics are possible, and in fact, less are needed.

b) Built-in test equipment (BITE) can be designed into the avionics equipment, to provide measurement data not normally utilized in operation. When a failure is indicated these measurements are sampled, either by the computer in the system, or by special sequencing and comparison circuitry to determine the malfunction. Hardware complexity limits the degree of isolation by BITE.

c) Functional Testing. The suspected equipment is cycled through a functional sequence of operations of which it is required to be capable by specification, again either by the computer, or by special equipment. This is the least exhaustive technique, but it demands the least amount of diagnostic hardware and software. In conjunction with failure detection by comparison it may provide the most cost effective approach.

It is obvious from the foregoing discussion that a less-than-100% certainty of detection and recovery for the final failure must be included in the definition of the "safe" condition, as must the finite response time to resume an operational status.

3.2 Failure Detection and Isolation

Having discussed the impact of a given failure tolerance criterion on the degree of redundancy in the system, some of the finer points of failure detection and recovery by voting, comparison, and other diagnostics will now be reviewed. In a well designed system, with high signal to noise ratios and a minimal likelihood of widespread or catastrophic failure modes errors will be random, uncorrelated, and infrequent. In this environment comparison of redundant, independently-computed output data provides a near certainty of failure detection. Current estimates are that the Shuttle avionics system will probably be characterized as such a system. Comparison is being proposed for error detection in the Shuttle data bus system in the areas of:

a) the computers

b) the data bus

c) other sensors.

These areas will now be examined in turn.

3.2.1 Computer Failure Detection and Recovery

Although the computer operates in a highly involved and complex fashion, it is deterministic and exact: a given operation will always yield the same result if repeated with the same input data. The major problem for computer comparison in a real time environment such as the Shuttle data bus is the synchronization of computations which involve time dependent functions and input data. Synchronization can be achieved by:

a) central control of the computer clocks;

b) careful gating and distribution of input data;

c) strict identity of hardware and software operation.

A comparator/voter mechanism adds to the hardware and software complexity. It also incurs operational delays, because time is required:

a) to wait for synchronization of clock and data;

b) to perform the comparison;

c) to decide on the results of comparison;

d) to take corrective action.

To minimize overhead, the comparison should, therefore, take place at a fairly high level of operation, rather than instruction by instruction. Comparing the operation of the computers at the point where they influence their environment, i.e., at the computer/bus interface, is a logical choice, provided that outputs occur frequently enough.

Comparison and voting can be done in varying degrees, with varying hardware and software complexity:

a) majority voting on the output data of three or more computers, reducing to comparison with diagnostics when less than three good computers remain. The bus receives only the data derived from the majority vote. Failure isolation and correction is automatic as part of the voting process. The complex voter that this requires must be sufficiently redundant and possess adequate error protection to meet the failure tolerance criterion, because it is an in-line element in the data bus.

b) Majority voting on the indications of health, but not on the output data. One computer is selected to be "active", and its outputs control the bus directly. The other computers are used as standards to provide independent checks on the operation of the active computer. A voting mechanism decides on the basis of a majority of comparator results whether the active computer is operating correctly. It may also determine which of the inactive computers has developed a failure (see Figure 3.1). In the event of a failure of the active computer one of the others is made active. The voter mechanism may be considerably simpler than the data voter of the previous paragraph, since it only operates on binary values; its response time need only match the reconfiguration dynamics, not the transmission frequency of the bus. Furthermore, since it is not an in-line element of the system, it may not have to meet the same stringent failure tolerance requirements. Each comparator can be considered a part of a computer's I/O section and is thus naturally redundant. In fact, the comparison could be performed, by software, internal to each computer.

As a consequence of voting binary, rather than many-valued byte or word data, the simplicity of the second method pays a penalty in the lower inherent certainty of correctly interpreting failure conditions. There is a greater possibility for split vote situations to arise with binary variables, and a greater likelihood of identical multiple failure. However, these conditions will only arise when failures in the comparison and voting logic itself produce erroneous indication of computer health; the lower complexity of this voter will aid the achievement of the necessary reliability.

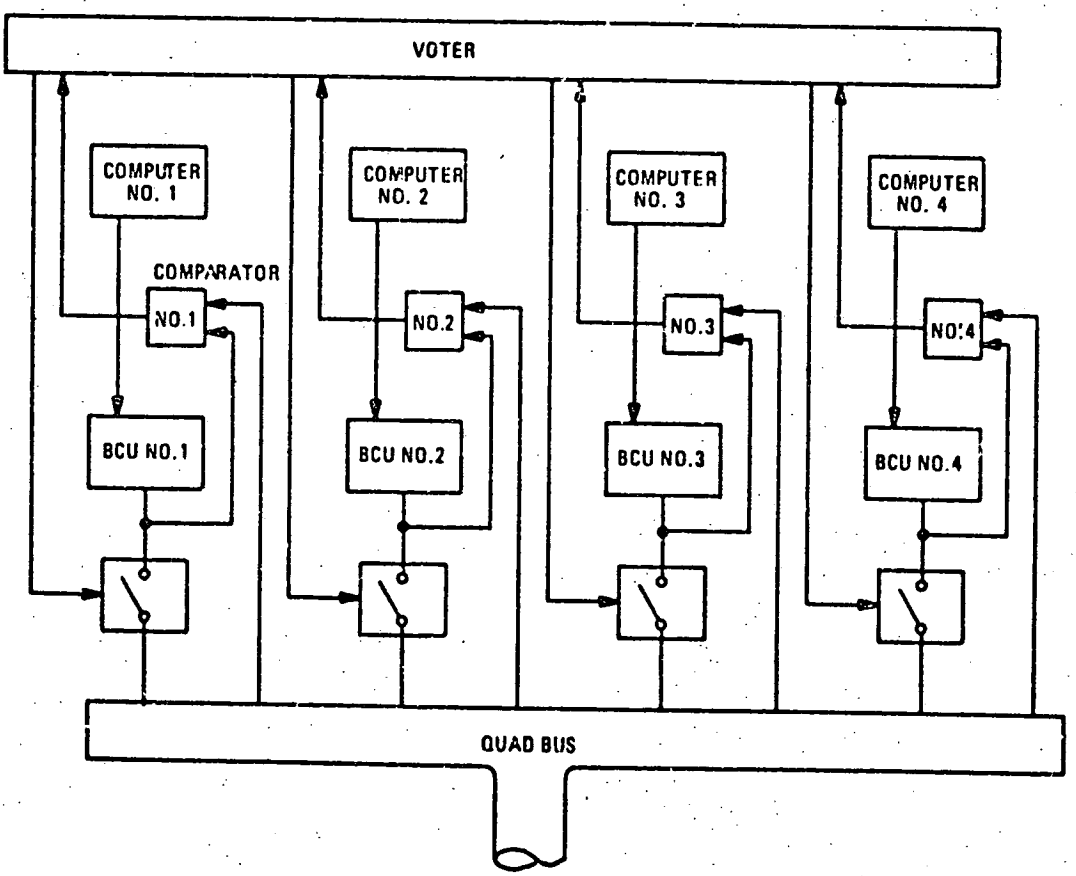


Figure 3.1 Computer configuration with external comparator and voter

For either voting approach, once less than three good computers remain, reliance must be placed on self-diagnosis to determine the faulty computer. No self-diagnostic technique can be infallible; a disagreement between two computers could yield the following conditions:

a) one computer determines itself to be faulty, the other finds itself healthy. This is the expected result.

b) Neither computer detects a malfunction. This may be because the fault was transient, or because it was a borderline case beyond the capability of the diagnostic method.

c) Both computers detect malfunctions. This event is highly unlikely in the case of uncorrelated random errors, but may easily occur for common mode problems such as physical environmental transients (e.g., power supply and thermal variations).

One insidious possibility for a processing failure that may not be trapped by any of the techniques discussed so far is that of the software error. The software in each of the redundantly operating computers must, for the purpose of comparison and voting, be virtually identical. It is, therefore, inherently non-redundant. A software fault will produce data which, being identically erroneous, will appear to compare correctly. This condition must be classed as a design error which, along with the similar logical hardware fault, must be prevented by careful design and adequate verification, rather than by complicating the system in an effort to make it immune to conceptual errors.

3.2.2 Data Bus Error Detection and Recovery

Although there are two causes of failure in the bus system, namely hardware failures and transmission errors, these may not be separable in cause or cure. Usually, the same error detection and recovery procedure handles both. Of the two main approaches, error correction coding or voting on multiple transmission, the first is treated in some detail in Chapter 5 and the second in Appendix A.

The repeated transmission of a message over a single path is a well known form of coding and can be used for error detection (by comparison, requiring all messages to be identical) or error correction (by voting, and accepting the message that is made up of the most often received bits). It is easy to

implement, but as coding systems go, it is relatively inefficient. In order to get a Hamming distance four code for three error detection, the message must be repeated four times. The same error detecting capability can be obtained with many fewer bits using other coding schemes.

The transmission of the message over multiple separate paths is in many ways similar to the multiple transmission over a single path. It is true that the message is received and verified at the output with less delay than is associated with the sequential transmission scheme, but on an overall basis, there is no improvement in the utilization rate of the available channel capacity. In analyzing the probability of an undetected error, for random independent errors, there is no difference between the two schemes. For errors caused by external influences, such as EMI, the probability of having all of the channels affected in the same way by an external occurrence appears to be quite low, especially if the channels are physically separated. In severe cases, it is possible to offset the multiple transmissions from each other by a small number of bits, so that the same information bit will not be affected on each line. However, the probability of having some number of sequential transmissions over a single channel be altered identically by sequential external occurrences, is of very low probability. Therefore, unless the reduction in throughput rate becomes unacceptable, there appears to be little advantage, from the point of view of error detection and correction, to be derived by parallel transmission. The redundancy of the bus should be determined by the need for hardware failure protection alone.

Voting on multiple transmissions from the computers implies a complexity at the receiving terminal:

a) storage must be provided to hold each transmission, if sequential;

b) a majority voter to act on the redundant transmissions is required;

c) for parallel transmission, a back-up error detection policy is required in the event of hard bus failures since voting is not feasible with less than three good lines;

d) the results of decisions on transmission validity made at the terminal must be communicated back to the control computer to maintain an up-to-date configuration status. This prevents the use of a pure command/response bus control policy (see Chapter 4).

e) For parallel transmission each terminal must access all buses to perform voting. The catastrophic failure of a terminal could incapacitate the whole bus system. In addition, for physically separated bus lines (e.g., port and starboard cable routing), extra weight is incurred for cross connections.

This complexity discourages the use of transmission voting as an error detection and correction scheme for all communications, since other techniques such as echo checking, described later, offer as much security without the overhead. However, critical communications can be conducted with a higher degree of confidence by repeated serial transmissions. The buffering and voting of these should properly occur outside of the bus, in the particular subsystem involved. It should be noted that in the absence of a comparison mode of SIU operation, it becomes almost imperative that bus transmissions should only occur on one line at a time to avoid the confusion that would result at the subsystem from the receipt of several simultaneous messages. This is especially true for configurations that require only one SIU connection to each bus line.

3.2.3 Subsystem Output Voting

The voting of data received over the bus by the computer has less value so far as validating the transmission of the information, since the computer has considerable flexibility in determining correct system operation (error coding, echo checking, etc.) that does not involve the complexity of voting, or multiple transmission by terminals. However, comparison of data from redundant subsystems provides a powerful fault detection and isolation capability which would be difficult to match by other diagnostic techniques (e.g., BITE). This capability should, however, have little influence on the design of the bus system: it is more a problem of data management by the software. Each computer in a redundant configuration must be able to access the data from each redundant element of the subsystem, which imposes a constraint on the interconnection of computers, bus lines and terminals. This will be discussed in the next section.

Multiply generated data from transducing subsystems, such as the inertial reference, present other problems for voting in addition to those of time synchronization described earlier. Such data is generally derived from analog quantities and is subject to drift, scale factor errors, etc.

3.3 Redundancy Interfacing

This section will discuss the interconnection of the basic units of the Shuttle data bus system; i.e., the computer, bus control unit, bus, bus interface unit, and subsystem. Although five levels of redundancy could be required to meet the FO-FO-FS criterion, a maximum of four will be assumed in the discussion. It is expected that strict adherence to FO-FO-FS throughout the avionics systems will not be demanded, nor will it, indeed, be practical. It has been assumed for this study that some systems, because of non-criticality or extreme reliability will not be required to demonstrate four-fold redundancy, but will, nevertheless, demand the full failure tolerance from the bus. To interface these to the quad-redundant bus presents a redundancy matching problem; cross connection, or cross-strapping of the different levels becomes necessary. That an overall increase in system reliability may be a by-product of cross-strapping is indicated in Appendix A. Another reason for cross-strapping is to provide a greater flexibility for the application of comparison and voting among the redundant levels, and for the management of system reconfiguration. Finally, in a real Shuttle environment with geographically divided bus lines, cross-strapping prevents a situation in which the left side computers could not run the right side equipment. The question of cross-strapping occurs mainly at the computer/bus and at the bus/subsystem interfaces. These areas will be examined in turn.

3.3.1 Computer to Bus Interface

This interface involves the computers, BCU's and the bus lines. Only the expected case of quad redundant computers and four buses will be considered. The several options are illustrated in Figures 3.2 through 3.5 and are now compared in turn.

a) Configuration 1 (Figure 3.2)

This is the simple approach of no strapping at all. The computer, BCU and bus constitute a single string unit. A failure of one of the elements in the string fails one entire bus line. Communication of data between computers must be done via the bus terminal or by special purpose cross links between the computers. Error checking of the bus by comparison is impossible at the computer end. Each bus terminal must access all bus lines, which makes the whole bus system vulnerable to a catastrophic terminal failure. The bus terminal is complicated by having to bear the brunt of bus system failure detection and correction, reconfiguration, and status monitoring. If comparison

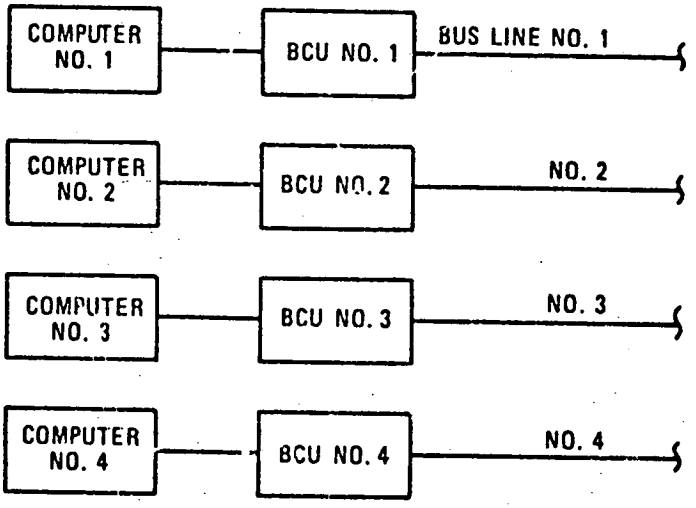


Figure 3.2 4 single strings without cross connection

or voting of the computers and the BCU's is to be done in this configuration, it must be accomplished by the terminals. Voting of sensor outputs is plainly not possible, even at the terminals. Certain subsystems must have all LRU's powered up and running; e.g., the inertial measurement unit. These subsystems require all the string elements of the bus to be up and running: partial bus operation is not feasible without cross-strapping.

b) Configuration 2 (Figure 3.3)

In this configuration the computer and BCU are a single string unit: cross-strapping occurs at the BCU/bus line interface, enabling each computer/BCU to access any bus line. The access may consist of either

1) transmit and receive on one line and receive-only on the others

or

2) transmit and receive on all.

Configuration 2 provides:

1) the capability for each computer/BCU to monitor the bus line outputs of the others, thereby enabling the comparison/voting of computer performances discussed in Section 3.2.1. Note that this arrangement includes the BCU in the error detection loop.

2) The ability to reconfigure the arrangement of active computer/BCU and bus lines in the event of failure. In this configuration it is not possible to transmit onto the bus the majority voted output from the several computer/BCU's, since data for voting is only available after the bus has received the active computer's output. Furthermore, a failure to the active computer/BCU is detected only after the erroneous message has been transmitted into the bus system. Since the terminal in this configuration need not access every bus line, as was necessary in Configuration 1, it has been relieved of the burden of checking on computer operation.

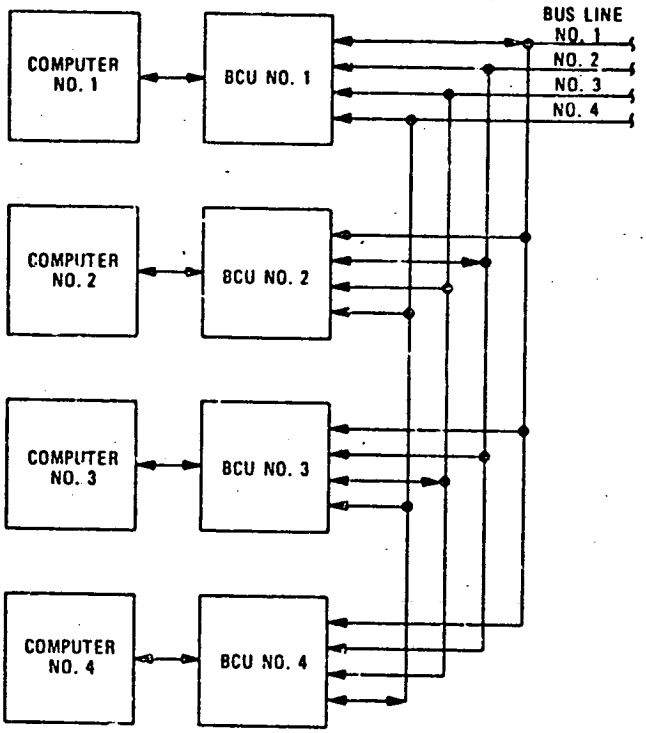


Figure 3.3 Cross connection between BCU and bus

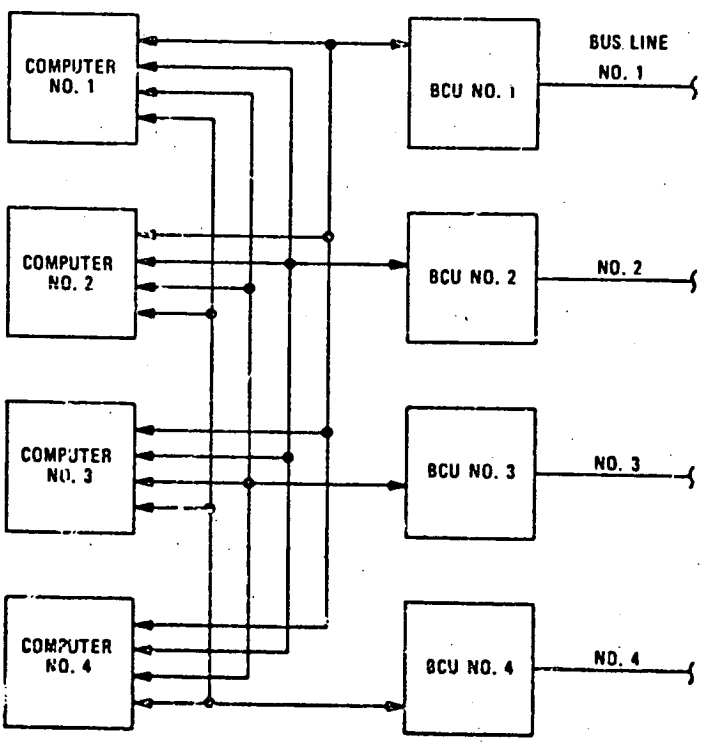


Figure 3.4 Cross connection between computer and BCU

c) Configuration 3 (Figure 3.4)

This is an alternate cross-strapping arrangement which, as far as the bus lines and the terminals are concerned, has similar characteristics to that of Configuration 2. However, each BCU/bus line is now the single string element, rather than the computer/BCU. Cross-strapping occurs between the computers, which makes possible the parallel transfer of data with a great increase in the speed of comparison. This may allow comparison to be performed before erroneous data is transmitted to the bus, and a measure of majority output voting, even with only one active computer. Comparison and voting may be performed with software, with consequent flexibility.

However, since the computer to BCU is likely to be a parallel data interface, it will be complex. Cross-strapping at this point will therefore present considerable difficulty. The complexity can be limited if a computer's access to the other BCU's is restricted to receive-only, and if it transmits to the bus only through its own BCU, which is in turn dedicated to a particular bus line. This restriction means, however, that a bus line failure will, in effect, disable a computer.

d) Configuration 4 (Figure 3.5)

In this configuration the computers, the BCU's and the bus lines are all separately reconfigurable. Cross-connection exists between the computers and the BCU's and between the BCU's and the bus lines. This arrangement can be considered:

1) as a combination of Configurations 2 and 3 above. The reconfiguration flexibility thereby achieved is severely off-set by the complexity of the interconnections and the magnitude of the configuration management task;

2) As the only one that allows full majority voting of computer outputs by the BCU before transmission over the bus.

In the second of these two roles, illustrated in Figure 1.3.4, Configuration 4 suffers from a number of disadvantages:

1) Since the BCU functions as a majority voter on all the computer outputs, it is necessarily a single point element of the system. It must, therefore, be internally redundant to meet the FO-FO-FS criterion, and in addition must possess its own failure detection, isolation and reconfiguration mechanism.

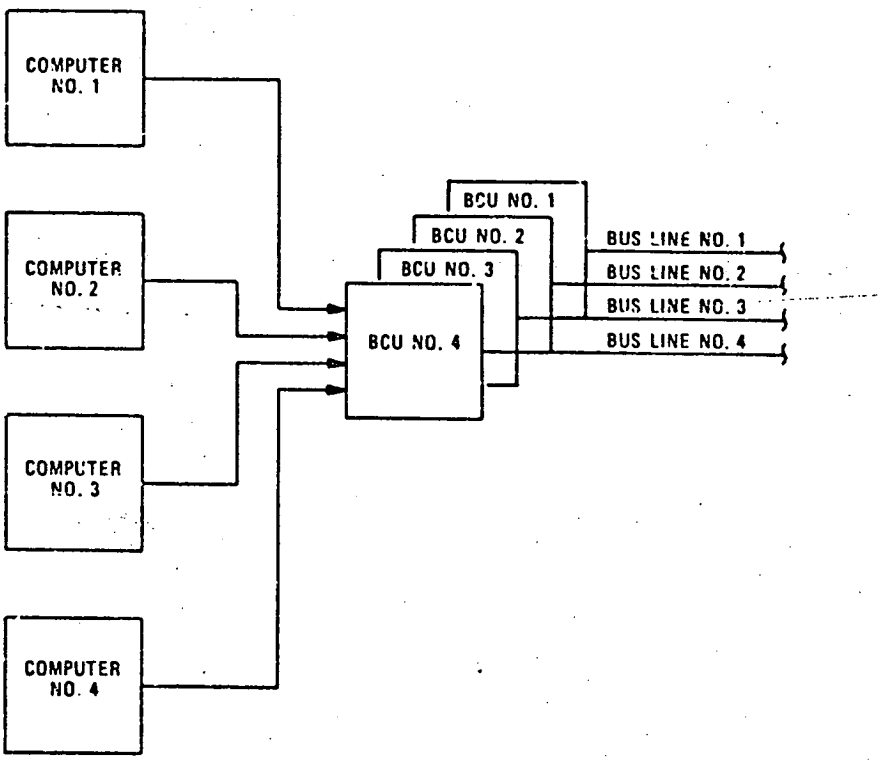


Figure 3.5 Quad redundant BCU configuration

2) It is not possible to drive the bus lines with non-identical data (as might be desired in a future expansion of the capabilities of the bus system), since there is no clear dedication of a BCU to each bus line. For the same reason parallel input voting would be difficult.

The foregoing discussions have assumed that the computer is performing a central function in the control of the bus, and that replication is purely to achieve failure tolerance. It is possible to consider more than one computer in the operation of the bus system, as described in Chapter 4, but these cases will not be discussed here. Suffice it to say that the complexities of cross-connection, failure detection and isolation, and reconfiguration management increase steeply with each multiply redundant computer that is added to the system.

3.3.2 Bus-To-LRU Interface

The configuration of this interface is influenced by the configuration of the computer/bus interface and by the redundancy levels existing in the subsystem to be serviced by the bus. The elements of this interface are shown in Figures 3.6 through 3.10 which illustrate the major options for cross-strapping the connection from a quad redundant bus to a triply redundant subsystem. From the point of view of redundancy interfacing these elements are characterized as follows:

a) SIU - connects to each bus line by a single, serial data path. May be simplex or internally redundant.

b) EIU - connects to each SIU by one or two serial data paths, and several control lines. Need not be physically separate from the SIU (or from subsystem, depending on configuration).

c) LRU - connects to each SIU by a complex interface that may consist of various signal types: serial, parallel, discrete and analog, of up to 50 or 100 signal paths.

Taking each cross-strapping option in turn the following observations can be made.

a) Configuration 5 (Figure 3.6)

This is the trivial case of no cross-strapping at the terminal. The bus, SIU, EIU and LRU constitute a single string element. A failure of any one component fails the whole string. Reconfiguration consists of isolating the faulty string and switching to a good one. This has the advantage of simplicity. LRU's may be geographically separated without involving local cross-connection penalties. However, the subsystem is serviced by a bus that does not meet the FO-FO-FS failure criterion, since the fourth line is not connected.

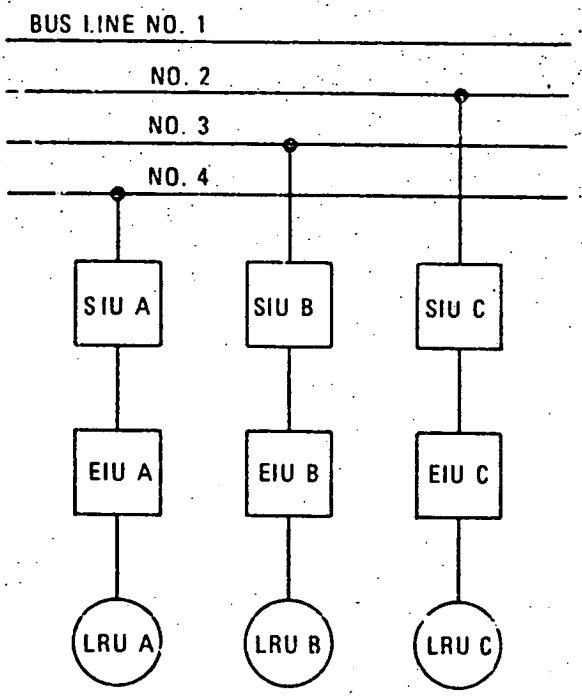


Figure 3.6 Single string no cross connection

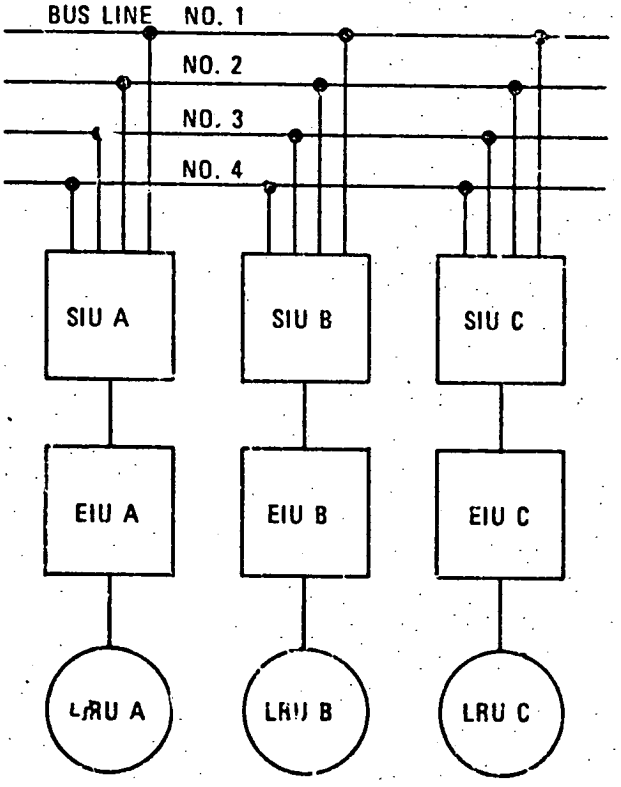


Figure 3.7 Cross connection between SIU and bus

b) Configuration 6 (Figure 3.7)

Cross-strapping occurs at the bus/SIU interface. The SIU, EIU, and LRU are here a single string element, and each LRU can be accessed by all bus lines. The full failure tolerance of the bus is therefore available to the subsystem. Reconfiguration is, as for the first example above, still a matter of switching to a good string. Disadvantages of this approach are:

1) The SIU and EIU are considered a part of, and their redundancy is determined by, the subsystem. The complexity of the SIU/EIU combination may force the use of a higher level of redundancy in a subsystem of superior reliability than would otherwise be considered.

2) Each level of subsystem redundancy requires four connections to the bus, in this case a total of 12. Bus connections should be minimized, as is discussed in Chapter 6.

3) A catastrophic failure of one SIU can disable the whole bus system. (The use of bus line couplers can limit this to failure of the subsystem only.)

4) Comparison of the outputs of LRU's cannot be accomplished locally. (This is not a serious drawback, since local voting may be undesirable in any case on grounds of complexity.)

5) It is not possible for one terminal to compare, or vote on, multiple, parallel bus transmissions.

6) A separate address in the bus control word format must be provided for each SIU to allow for reconfiguration.

7) Only one EIU may be serviced by each SIU.

c) Configuration 7 (Figure 3.8)

Here cross-strapping occurs at the SIU/EIU interface. The SIU becomes an element of the bus, and the EIU is considered a part of the LRU. The degrees of redundancy of SIU and EIU are, therefore, determined by the bus and subsystem respectively. The advantages of this arrangement are:

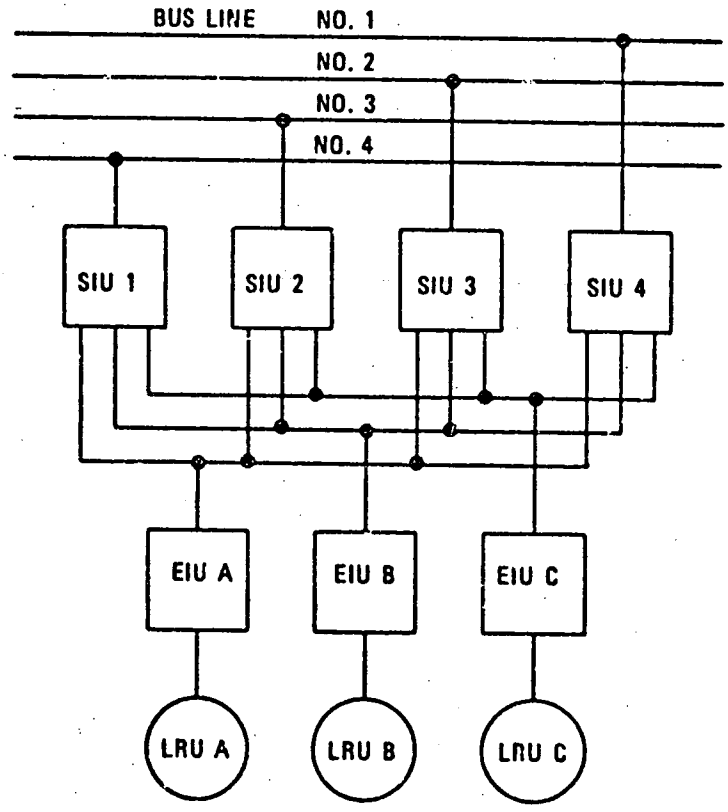


Figure 3.8 Cross connection between SIU and EIU

1) No single SIU failure can disable the bus system, although it could disable all EIU's (and therefore the subsystem).

2) LRU comparison could be done at the SIU level (although at considerable cost in complexity).

3) If bus configuration is controlled by the computer/BCU (which is probable), then a separate address is not required by each SIU at a terminal.

4) Cross-strapping involves only a few physical paths, since data at this point is still largely serial.

The disadvantages are:

1) Four SIU's are required even if the subsystem redundancy level is lower (this is only a real disadvantage for reliable but critical subsystems which require the full FO-FO-FS bus tolerance).

2) The bus system can only be run in a simplex communication mode (e.g., command and data cannot be assigned separate paths).

3) The cross-strapping complexity, though much lower than Configuration 8, may exceed that of 6. This is of concern if the bus lines are separated by considerable geographical distance.

d) Configuration 8 (Figure 3.9)

In this configuration the bus redundancy is carried to the EIU level and cross-strapping occurs at the LRU interface. This approach allows the SIU and EIU to be considered as one unit, an advantage if one standard bus-to-subsystem interface is specified. However, its overwhelming disadvantage is the complexity of cross-connecting an interface that may consist of 50 to 100 separate paths to each LRU. Such an arrangement should be considered only for very simply connected LRU's.

e) Configuration 9 (Figure 3.10)

This is not strictly a different configuration, but rather a version of configuration 7, designed to cope with the problems of widely separate LRU's. Such separation may be specified to reduce the vulnerability of Shuttle avionics elements to catastrophic occurrences such as meteorite

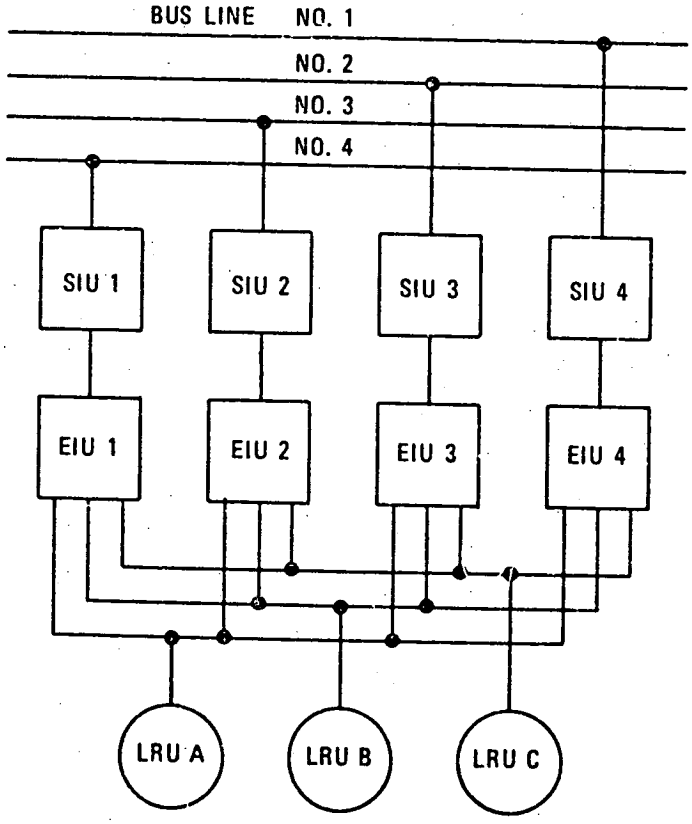


Figure 3.9 Cross connection between EIU and LRU

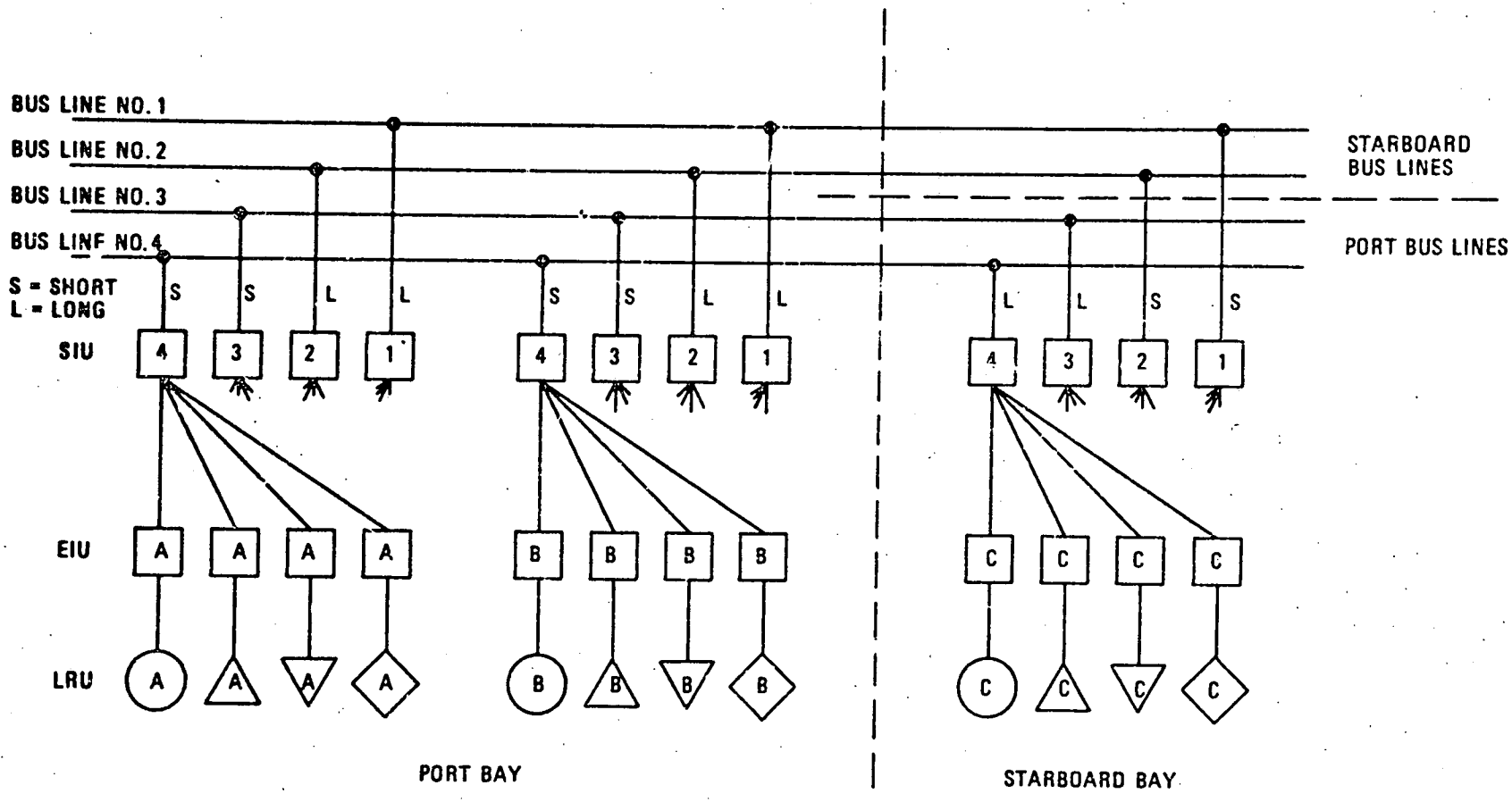


Figure 3.10 Bay oriented configuration

collision, or the explosion that crippled the Apollo 13 service module. Rather than associating a group of SIU's with a specific subsystem, Configuration 9 associates a group of SIU's with a particular level of redundancy. This approach is particularly suited to the clustering of equipment into bays located at strategic positions in the Shuttle vehicle. For example, a group of four subsystems each with triple redundancy would require only three groups of SIU's, two in one bay and one in the other. The long cross-over leads from port to starboard and vice versa (indicated by L in Figure 1.3.9) can be formed by T-junctions into the bus lines; i.e., each need only be a single data path. In the mechanization of Configuration 7 on the other hand, this arrangement would require four groups of SIU's, with vehicle cross-overs occurring at the more complex SIU/EIU interface. In general, Configuration 9 scores whenever equipment can be so grouped that the number of subsystems serviced by one bus terminal exceeds the subsystems' degree of redundancy.

3.4 Relationship of Computer To Bus

The relationship between the computers and the rest of the data bus system is a complex subject. The choice of a configuration that best meets the requirements of the Shuttle is impacted by many factors, most of them outside the scope of this study. However, one important consideration of direct consequence to the design of the bus is the computer's role as the controlling element in the system.

The following discussion will take a more general view of the integrated avionics system than that assumed so far, in order to compare and contrast the various approaches. The only assumption made is that all or part of the complement of avionics equipment will be interconnected by common data buses, rather than by dedicated interfacing. The configurations then involve the use of one or more computers, and one or more data buses. The options may be summarized as:

a) single (central) computer, single bus

b) distributed computers, single bus

c) distributed computers, multiple buses

There are several variations of these, depending on the requirements specified for the system and the functions of the elements.

3.4.1 Central Computer, Single Bus

Topographically, this is the simplest arrangement (see Figure 3.11, although it imposes a higher level of sophistication on both hardware and software than the distributed approaches. The central computer is the sole authority on the bus and all bus communications are initiated and directed by it. In addition, it exercises control over subsystem operation (for example; IMU, radar, environmental control, power distribution, etc.) and performs all required computations and data processing. Only under certain conditions, described later, may processing be performed elsewhere. All the avionics subsystems and their equipments interface with the one common bus, and the resulting bus traffic data structure must accommodate the full complement of addresses, commands, and data interfaces. Since there is only a single functional copy of the computer and the bus, the replication of equipment to achieve the required level of redundancy is minimized; i.e., the amount and complexity of reconfiguration hardware and software, and the number of interfaces to be controlled. The main drawbacks to this approach are that the central computer must

a) accommodate all Shuttle avionics functions;

b) possess a high level of performance;

c) contain a greater volume of software, of greater complexity, than any computer in a distributed arrangement.*

The software burden of the completely centralized computer configuration can be alleviated, without degenerating the autonomy

---

* It should be pointed out, though, that the total volume of software in a distributed set of computers, performing an equivalent job, will be higher than in a central computer, for the following reasons:

a) each computer must carry its own set of executive and system routines

b) bus control routines must exist in each computer (although some hardware/software trade-offs can be made with the bus control unit);

c) certain bus system data, such as status information, must be maintained in each computer to enable it to use the bus intelligently;

d) extra software is required to enable the computer-to-computer communications.

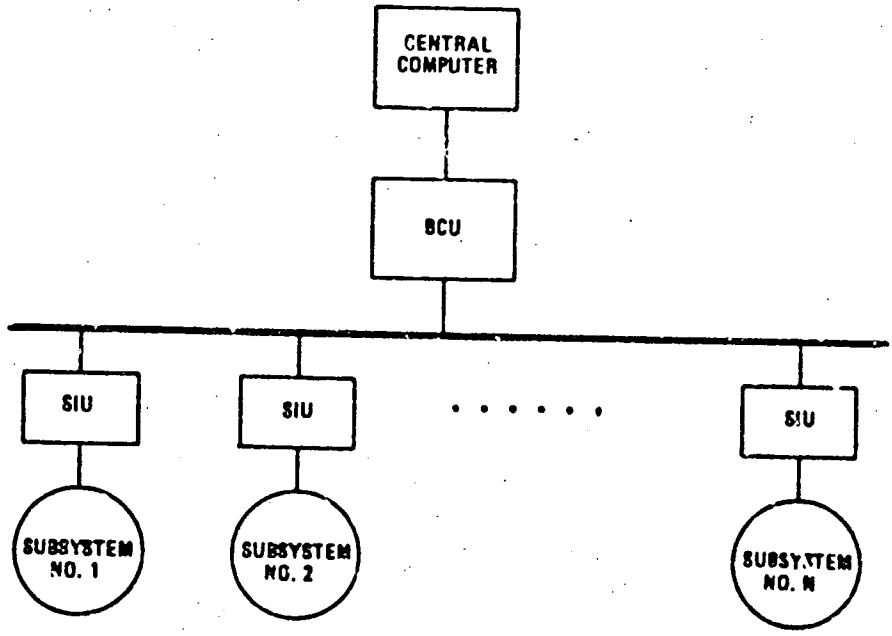


Figure 3.11 Central computer control

of control that the central approach offers, by dedicating to a subsystem its associated processing task. A separate computer may then be introduced into the subsystem to take over the task from the central computer. The condition, however, being that it forms a part of the subsystem, and that it makes no demand to control or directly access the bus. Any data requirement will be communicated through the regular subsystem-to-bus interface. The central computer need not then be aware of the other computer.

Two examples of such an arrangement could be the display processor, which provides stored display formats and refreshing to the crts, and the main propulsion system processors. A configuration which treats these as subsystems in a centrally managed bus system is shown in Figure 3.12.

The software in the central computer services all equipment connected to the bus and performs all the functional requirements of the system. The boundaries of a functional subsystem tend to disappear, and exist only as shared software in the computer. For example, the stabilization and flight control system will consist of redundant sensors connected to the bus operated by programs which are allocated a portion of the central computer resource. The total flight control subsystem (including software) is therefore not a visible separate entity but becomes, in fact, part of an integrated avionics system.

The centralized system promotes standardization of approach to system design problems. Centralized software and a central bus provide the means to impose this standardization.

A centralized system does not, however, provide isolation or localization of changes. Changes made to a particular system such as electrical power distribution, may not be easily or absolutely isolated from the rest of the system. Nor does it provide a hardware independence of functions. That is, security of subsystems is only achieved through the bus system design.

3.4.2 Distributed Computers, Single Bus

The sharing of the resource of the single bus between the several control computers characterizes this configuration. Two different versions of this configuration allow each computer to gain access to its subsystems:

a) Time-slotting. In this approach (Figure 3.13) a time-based sequencer, external to the bus and the computers, grants exclusive use of the bus to each computer in turn.

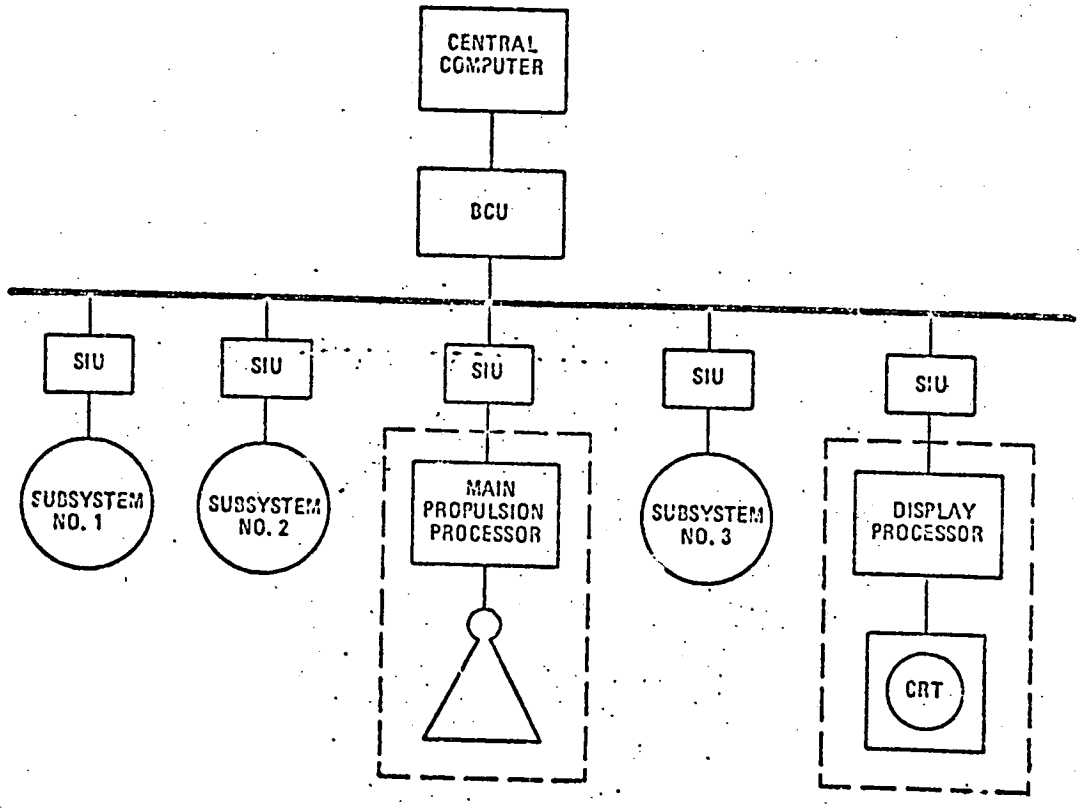


Figure 3.12 Central bus control with distributed processing

During its interval of control, each computer has sole access to all subsystems. The intervals are pre-set in the sequencer and of fixed duration, giving this approach an apparent simplicity. A more flexible, but more complex, variation on this technique places control of the access interval in the hands of one of the computers, so that the assignment of computers and subsystems to the bus may be varied in response to the changing requirements of different operating modes and mission phases.

b) Master-slave relationship. In this arrangement, the responsibility for the overall management of the system rests with the master computer. The control of the bus is transferred to a slave computer at instances of time and for periods determined by the master computer in accordance with the needs of the system. This has very similar characteristics to the time-slotting technique with variable time slots, but is not constrained to a cyclic control sequence.

A computer-to-computer communication link becomes an unavoidable requirement of both these arrangements for two main reasons:

a) configuration status, operation mode, and other operational information about the bus must be communicated from one computer to the next to prevent uncontrolled or inadvertent interaction between the different activities within the computers.

b) It is very likely that a degree of communication and control overlap will occur in which a common item of equipment on the bus will be accessed by more than one computer. In such cases a difficulty arises if the transaction of control and data between computer and equipment is not complete by the time the computer relinquishes the bus. The equipment is then obviously not prepared to be accessed by the next computer in the control sequence. (Conflict over its allocation could be tackled by providing a locking mechanism in the equipment itself. Such a mechanism would, however, violate the principles of a command/response bus control policy, since the equipment would be capable of exercising a control initiative.) Conflict between two or more computers over the sharing of a common resource is a well known problem, and requires cooperation between the computers.

Computer-to-computer data transfer can take place either:

a) over the common data bus, by providing a special time-slot and/or message format for this purpose, or

b) over dedicated data paths between the computers (through I/O or memory access channels).

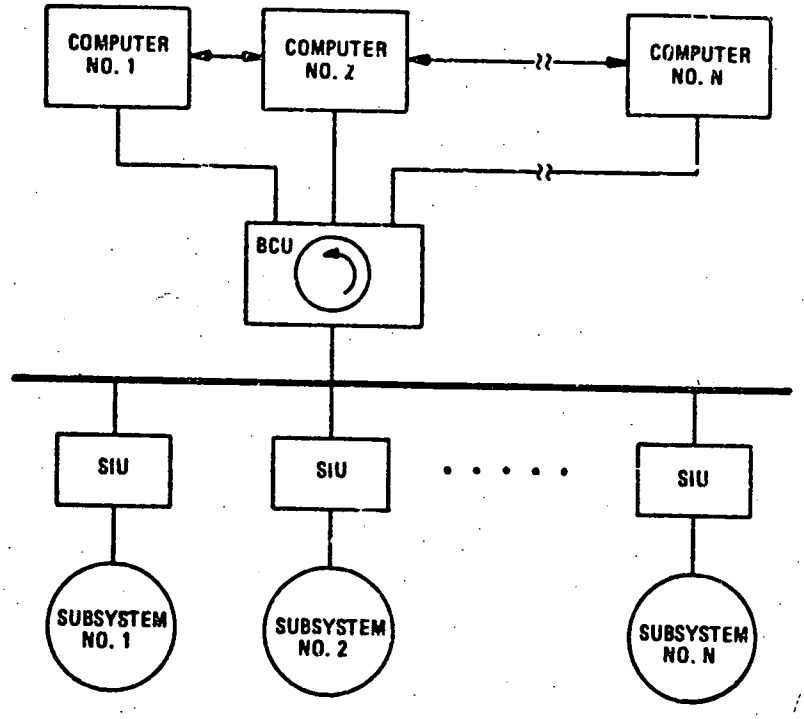


Figure 3.13 Bus control by several computers

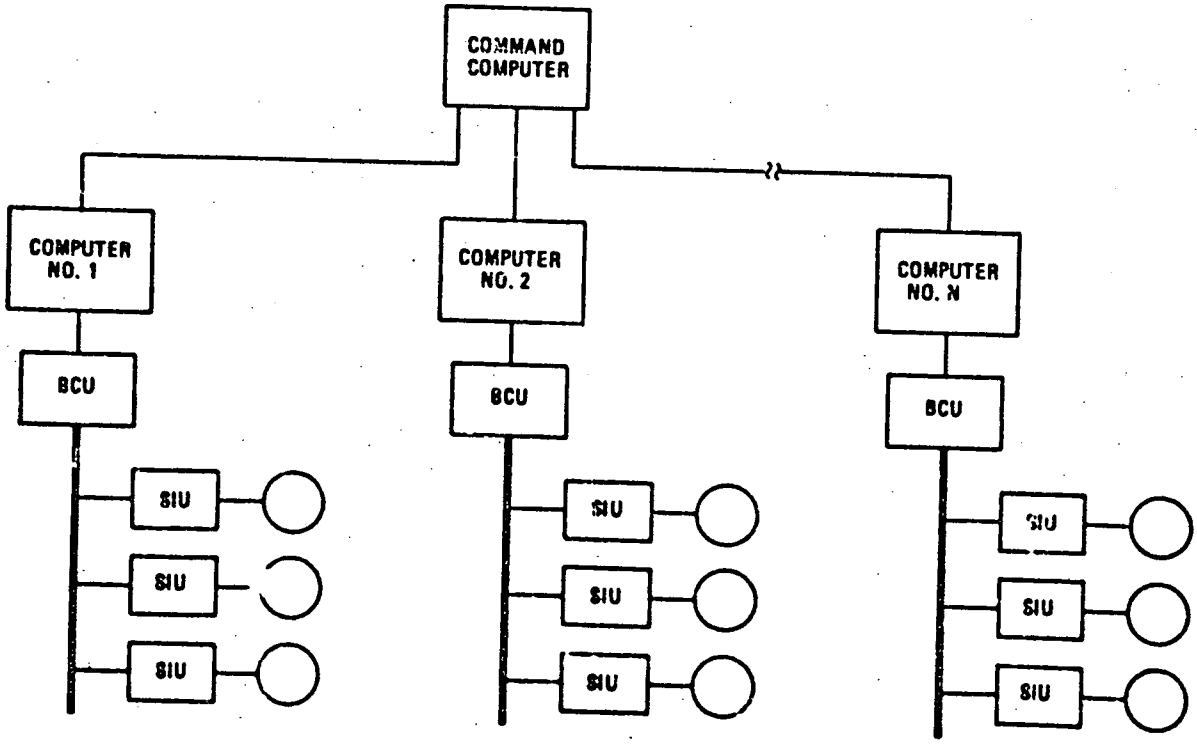


Figure 3.14 Distributed computers and multiple buses

The need for such communication breaks down the independence of the various computers, and the characteristics of the distributed configuration tend to assume those of the central. The key issue here is whether the separate computers can be uniquely identified with independent functions and dedicated subsystems or whether interaction of control and data between the computers is unavoidable. If the independence of the computer functions cannot be assured, then a physical distribution of processing introduces problems of communication and control that may create greater difficulty than if the job were accomplished in a single computer.

3.4.3 Distributed Computers, Multiple Buses

In this configuration, (see Figure 3.14) a subsystem that can be identified as independent, i.e., not requiring the services of equipment in other subsystems, is administered by a dedicated bus and control computer. A central command computer coordinates the activities of the various subsystem computers. Its functions are:

a) to perform most mission oriented computational tasks (such as targeting, navigation);

b) to perform high level system decisions (such as mission mode selection);

c) to coordinate and control communication between the distributed computers of high level, but low rate and low volume data (such as crew commands).

The local computers perform all high speed operations required by the local buses and their equipment. They provide a fully operational subsystem capability to the command computer. This implies a high degree of autonomy in the areas of:

a) status monitoring,

b) fault detection and isolation,

c) reconfiguration in response to failure indication,

d) all processing required to implement subsystem functional capability.

In essence, the local computers provide a bandwidth compression by removing from the central computer the burden of processing at high repetition frequencies and the control of activities requiring fast time responses.

The provision of separate buses encourages the formation of functionally related sets of equipment into subsystems. Since the subsystems are no longer tied together by the common control mechanism of a single data bus, they can achieve the degree of independence required for a distributed computer configuration.

The configuration management function can be more easily distributed than in the case of the single bus configurations, because of the degree of local autonomy possible with dedicated buses.

The penalties of this approach are:

a) multiple buses imply a weight and power disadvantage over the single bus (although the penalty is far less than for a dedicated wiring approach).

b) In common with the other multiple computer approaches, there is a replication of hardware and software that is avoided in the central computer configuration.

"Page missing from available version"

page 46.

PRECEDING PAGE BLANK NOT FILMED

Chapter 4

Operation and Control of the Data Bus

4.1 Data Bus Access and Control Philosophy

Since the Shuttle data bus constitutes a central communications resource shared among multiple terminals and a central controller, a fundamental feature of its design is the method by which it is allocated to a particular communication path. The data bus system is essentially a "party line" shared by all terminals: when access is granted, the bus is dedicated to a single communication path between a transmitting and receiving station.

Selection of the bus access method is a basic decision because it constrains the design of both the remote terminal and the bus control unit. A general description of candidate approaches and a comparative evaluation is provided in this section.

4.1.1 General Description of Bus Access Methods

Several approaches to accessing a communication line have evolved out of the design of digital data acquisition and telecommunications systems. Four categories of line access and control have been identified: contention, polling, sequential time slot, and addressing.

4.1.1.1 Contention Access

A contention access method is one in which the remote terminals that desire to transmit "bid" or contend for the use of the bus. The first terminal to initiate contact on the line, not currently in use, seizes the line and prevents its use by other terminals until it has concluded transmission. A contention method must provide a means for resolving conflicts between contending stations. The queue list of "contending

requests" is either examined in a prearranged sequence or allocated via priority. The contention access method allows random accessing of the bus by terminals which have determined their own need to transmit,rather than by planned allocation via a central controller. The method requires a degree of control intelligence at the terminal.

An interrupt controlled bus represents a form of contention access in which the central controller receives an interrupt signal from a terminal requiring service. The controller allocates the bus to the queue of interrupt requests according to a pre-defined allocation algorithm.

4.1.1.2 Polling

Polling is a systematic, centrally controlled method of permitting terminals to transmit without contending for access to the bus. Polling is accomplished by the central controller which periodically contacts the remote terminals and requests each if it requires the bus. The controller will continue to poll the remote terminals in some order until one is found to require the bus.

The most straightforward polling technique involves a polling signal sent in a "round robin" sequence, and each terminal returns a positive or negative response. More efficient schemes exhibit sophistication in the polling sequence, or the order in which units are polled. One such scheme reported by MIT involves a polling signal recognized by all terminals. Each terminal requiring access attempts to transmit its unique number by transmitting a '1' onto the bus corresponding to each bit position containing a one in its wired-in identification. During transmission periods corresponding to zeros in its identification code, the terminal only monitors the bus. If it detects a '1' in this interval it ceases transmission and awaits another poll message. If a station succeeds in transmitting a complete number it has won the poll. It is important to note that a terminal cannot access the line in a system controlled by polling; however, utilization of the bus is random depending on terminal need and the outcome of the poll.

4.1.1.3 Sequential Time-Slot Format

Time slot data acquisition is essentially a method of granting access through a commutator such that data is transmitted in a prearranged order and is stripped out by a decommutator. In such a system all units on the line must be synchronized because of the rigid timing requirements of the structure. This approach has been used successfully in data acquisition, monitoring, and telemetry systems. The method involves allocating the bus to a particular path for a fixed time interval within a time frame. A time frame is organized

into fixed time slots and is initiated by a sync pulse. Each remote station requiring access begins counting clock pulses and at predetermined count starts to transmit its message. The sampling rate for each terminal can be varied by changing the interval of time allocated to it.

A variation on this approach incorporates a format code into the time slot structure defining the format of the data. The format code can be changed during the mission by the controller as communication requirements change.

The term "time slotting" has also been used in another context to describe a method for sharing the control of a bus system among several controllers. It was examined in Section 3.4.2.

4.1.1.4 Command Response Addressing

In a command response addressing scheme access to the bus is centrally managed by the controller. Under this concept, the controller transmits an appropriate command to the terminal including: synchronization header, terminal address, function to be performed (transmit, receive), data, and parity coding. Upon recognition of its address, the terminal interprets the command and begins transmitting or receiving the appropriate data.

Using command response access, a terminal does not initiate any communication unless it is commanded to by the controller. Terminals only "speak" when "spoken to".

In contrast to the polling scheme a terminal is not "polled" as to whether it wants the bus or not but rather is "commanded" to send or receive a message. Command/response addressing is similar to a polled system in that a terminal responds only when addressed.

A fundamental characteristic of command response control is that the "intelligence" of when, what, and how often to communicate is in the controller (i.e., computer software). There are consequently no access conflicts to resolve as in the contention method, or local decisions required as in the polled system.

4.1.2 Qualitative Evaluation of Access Methods

The advantages and disadvantages of the several bus access methods are discussed next. They are summarized in Table 4.1.

Access Method	Advantages	Disadvantages
Contention	1. Efficient allocation with a small number of terminals. Bus communication only when needed. 2. Effective for a random input/output environment at the terminal. 3. Provides flexible structure for adding terminals.	1. Requires local intelligence at the terminal to determine need to access. 2. Requires resolution of access conflicts. 3. Non-deterministic bus traffic. 4. Failed terminals difficult to detect
Polling	1. Centrally controlled allocation. No conflicts in access. 2. Comparatively efficient in bus utilization. Access granted only for positive responses to polls.	1. Terminal access is random, I/O loading difficult to predict. 2. Polling frequency for individual terminals must achieve system response requirements. 3. Terminal errors may be undetected for negative response to poll.
Time Slot Format (Fixed and Variable)	1. Least complex terminal. 2. Efficient data transfer. Little or no overhead in message format. 3. Simple to test	1. Inflexible. 2. Rigid timing and synchronization.
Command Response Addressing	1. Access and utilization are determined in advance and the load balanced in accordance with computer requirements. (No backlog of I/O).	1. Inability to efficiently accommodate random input streams.

Table 4.1 Bus Access Method Comparison

Access Method	Advantages	Disadvantages
Command Response Addressing (cont'd.)	2. Relatively simple terminal design. 3. Flexibility achieved via software I/O command bits. 4. Error control procedures more positively controlled. 5. Simple to reconfigure via software.	2. Overhead in traffic required for addressing and command data. 3. BCU complexity is greater than other approaches

Table 4.1 Bus Access Method Comparison (cont'd.)

4.1.2.1 Contention: Advantages and Disadvantages

One advantage of the contention bus access method is that it could be the most efficient from a system point of view. Utilization of the bus occurs only when it is required; terminals do not have to be polled, commanded, or preprogrammed into a time slot. This approach is most effective when the events which result in bus communication are random; for example, inputs from the crew. It allows equipment to be added very simply, even after the system structure has been established.

However, contention access requires each terminal to possess enough intelligence to know when it needs to communicate. This increases the complexity of the SIU in the Shuttle bus system. Another difficulty arises in assigning fixed priorities to each unit such that conflicts in access can be resolved, particularly in the case of the Shuttle with a large anticipated number of terminals.

Since bus access is random, the load on the bus is not balanced and at peak loads a backlog of access requests for service will accumulate. This could affect the response time for certain subsystems with high frequency bus utilization requirements, such as the flight control system.

The non-deterministic allocation of the bus creates difficulties in system and software verification, particularly in testing for operation under all traffic conditions. Another problem with an interrupt controlled system is that a failed terminal cannot immediately be recognized by the controller. It may not be able to bid for the line, and consequently the fault may exist without being detected. To avoid this situation the controller would be required periodically to sample the status of each terminal and to determine its status. The frequency of this status check could be high for terminals connected to time critical subsystems, diminishing the efficiency of bus utilization provided by the approach, and increasing its complexity.

If the Shuttle data bus consisted of a small number of intelligent terminals this approach would be a reasonable candidate.

4.1.2.2 Polling: Advantages and Disadvantages

The major advantage of a polled scheme is that it eliminates random access to the bus but maintains a moderately efficient data transfer; i.e., communication only occurs when a positive response to a poll is received. The efficiency of bus utilization will depend on the selection of a polling sequence and frequency which minimizes the number of negative responses. It is effective with a large number of terminals because it eliminates bus conflicts, and with a communication

requirement that is largely non-random.

The polling sequence and frequency must be consistent with system response and data requirements. If the polling sequence is implemented via hardware, it must be determined in advance of system development, and becomes an integral part of the communication system. Software control of the polling sequence would, of course, provide greater flexibility.

Disadvantages of this technique are:

a) although bus conflicts are avoided, the utilization of the bus is random, and at any time depends on the polling sequence and the terminal responses.

b) A faulty terminal may respond negatively or not respond at all to a poll request, making error detection difficult. The design of the system must allow the status of each terminal to be determined without requiring a response to the poll.

c) It requires intelligence at the terminal.

4.1.2.3 Sequential Time Slot: Advantages and Disadvantages

This approach is of course the simplest, imposing the least complexity on the terminal. Each terminal is prewired to recognize its time slot by a synchronized time delay in the frame. It is extremely efficient if the same information is communicated during the entire mission with very small changes. The only overhead in communication is the synchronization pulse, parity and coding bits (and possibly a format type). It enables full central control and is most effective for data acquisition. Since the structure of a frame format is relatively fixed it requires a minimum of software and is simple to test and verify.

The principal disadvantage lies in its inflexibility. The rigid timing and message structure is built into the hardware. It allows very little variation in bus communication requirements from one Shuttle mission phase to the next, and it is questionable whether it can satisfy all system demands, without losing its simplicity.

4.1.2.4 Command/Response: Advantages and Disadvantages

The command/response addressing scheme offers several advantages. First, there are no access conflicts. Communication requirements are predetermined by the designer and then implemented via software. There is no overloading of the bus.

Utilization of the bus is balanced by allocating the I/O servicing requirements to the available bus time. Correspondingly, computer processing may be interleaved with a predetermined I/O structure. As a result, the traffic on the bus, and the processing in the computer become deterministic, and may be more easily tested and verified.

The system imposes no significant complexity or intelligence requirement on the remote terminal. It provides certain error control characteristics. Since it is centrally controlled, a remote terminal only "speaks" when commanded to via a unique address. The terminal can be designed to include its own address in the response. Although this "echo check" is not an integral feature of a command/response system, the enhanced error detection control it provides cannot be as easily obtained in a polled or contention approach.

Finally, command/response provides reasonable flexibility in accommodating to a variety of I/O requirements, since this is accomplished by changing the software controlled I/O command sequences.

A basic disadvantage of the command/response addressing technique is that it does not allow randomly occurring events to be transmitted to the computer as they occur. Rapid response can only be achieved by commanding the terminal for the information at a high enough frequency. A command/response bus access method decreases the efficiency of bus utilization, since time critical or high frequency events impose a proportionately high bus I/O rate.

The addition of the command and address bits required to obtain or send data contributes to a higher bus traffic overhead for the command/response access method than any of the other approaches.

4.2 Control and Operation of the Data Bus by the BCU

Once a particular access method is selected, the communications procedure established to perform a single I/O transaction impacts the design of the bus system elements. The following steps, illustrated in Figure 4.1, must be taken in order for a single computer to send and receive data from a set of avionics equipment.

a) In a command response access concept, the computer directs all I/O requests in the system. It indicates along which

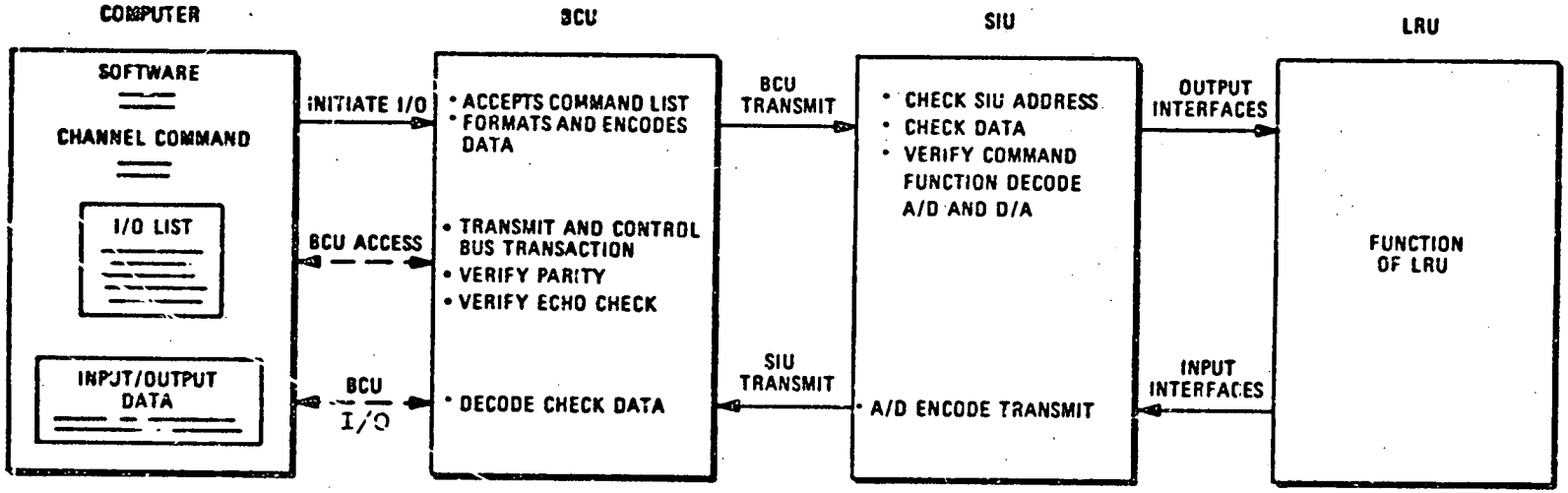


Figure 4.1 Basic functions during a bus transaction

bus line and to which remote terminal the message is routed, and if data is requested, where to put it when it has been obtained.

b) The BCU must encode the message and transmit it to the proper remote station over the selected bus line.

c) The remote terminal responds to the command, selects the appropriate channel to the LRU and executes the appropriate functions to obtain the data.

d) Signal conditioning and conversion takes place at the terminal, which then encodes and transmits the data back to the control unit.

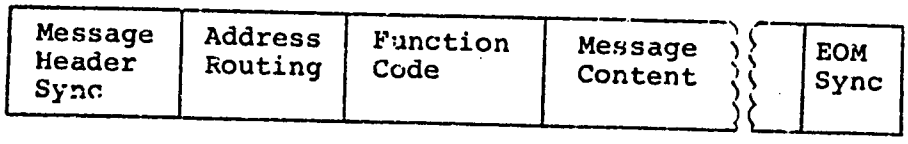
e) The established error-control scheme is maintained throughout the transaction.

f) The BCU transfers the data to the computer and informs it of the completed request or list.

The details of this transaction influence the bus message format, the functions of bus elements,and communication security. The message format and structure must satisfy the data acquisition and distribution requirements, without unduly complicating the bus hardware design. A level of transmission "security" must be established to minimize the probability of an undetected error,without significantly increasing the equipment complexity or message overhead. The following sections provide a general discussion of bus operation and the bus format and structure. The error control scheme is discussed in Chapter 5.

4.2.1 Bus Message Format

In general there are four basic parts to the structure of any communication message: the message header and terminator, the address and routing information, function code, and message content.



The first three parts of the message are associated with the communication system.

4.2.1.1 Message Header and Terminator

Message synchronization is required to enable terminals to recognize the start of a message and is usually a unique control signal recognized by the terminal. It is essential that the synchronization signal be different and clearly distinguishable from data to avoid misinterpretation. The characteristics of the sync signal will depend on the modulation technique selected. It is usually assigned a pulse width or phase change different from the standard data bit.

There are four possible sync signals: at the beginning and end of the BCU to SIU message and at the beginning and end of the SIU to BCU message. However, from a communication point of view they are not all necessary. The end of the BCU to SIU message can be distinguished by the "idle bus" when the BCU stops transmitting; similarly for the end of the SIU to BCU message. However, detection of an "idle bus" may cause circuit difficulties in either the BCU or SIU. The use of different sync signals for BCU to SIU messages and SIU to BCU message rules out inadvertent SIU to SIU communications, since the SIU need only respond to a BCU sync.

In any case, the only positive requirement for any address system is that there be a sync signal, clearly distinguishable from data, so that each terminal can begin to look for its own address in synchronization with the message. The need for other sync signals for end of message, accept, knowledge, etc., is a function of the communication procedures and the details of the implementation.

4.2.1.2 Address and Routing

The address portion of the message identifies the sender and receiver by "to X" "from Y". In a centrally controlled system, where there is no terminal-to-terminal communication, there is no requirement for the "from" part of the address. All communications are initiated by the BCU with transmitting/receiving occurring only between BCU and one SIU.

The "to" part of the message identifies the path to the LRU via an SIU address and an EIU address. A separate EIU address is necessary when the bus terminal communicates with more than one EIU. If the SIU and EIU were combined into a single unit, then the address could be combined.

4.2.1.3 Group Addressing

A group addressing capability would be required to send a single message to more than one SIU or EIU, as might be required to enable a passive flight recorder on the line to receive data intended for other terminals. Group SIU addressing could be an advantage in transmitting the same data to every element of a distributed subsystem, such as the individual quads in the RCS system. Group addressing would be useful in the central management of a redundantly configured subsystem, particularly if identical commands are issued by the computer to every redundant unit.

Group addressing on the bus requires the SIU to recognize more than one address. However, there is the problem of coordinating the return transmissions of echo or data messages. Coordination could be implemented in several ways: by sequential time slotting of the SIU responses, by ignoring the echo in the passive device, or by a contention access method. The SIU, EIU address and function codes would need to be coded in a way which would have group meaning. The tradeoff here is between the added complexity of the SIU and BCU hardware,and the additional software and memory to store multiple commands instead of one. A modification to the computer/BCU message to provide a routing indicator and a list of SIU addresses, which would enable the BCU to send multiple messages, could alleviate the computer software burden.

In summary, however, it is felt that group addressing is probably not worth the additional complexity in bus system design if, as has been estimated, there is adequate capacity in speed to accommodate the inefficiencies encountered.

4.2.1.4 Function Code

The function code field of the bus command specifies the action to be taken by the interface unit in acquiring or distributing data or signals to the LRU. The structure and format of this field is directly impacted by the requirements of the electronic interface portion of the remote terminal. In order to provide the capability of interfacing the majority of electronic equipment, the following types of interfaces would be required:

a) digital parallel,

b) digital serial,

c) analog data,

d) discrete.

The function code does not have to be in a standardized format for all terminals. More parallel digital signals may be required for a particular LRU, but less analog. The electronic interface itself need not be standardized. The function can be decoded and interpreted by specially tailored function controllers at the terminal. Alternatively the function code could represent the address of a location in a control memory which stores special control sequences within the interface unit. There are several ways of organizing the function code field, which are discussed in the following paragraphs.

a) Channel Addressing

Under this concept, each interface is assigned a channel address, and the function code becomes part of the address structure. Group addressing is possible only if channel addresses are in sequence (e.g., 2 through 6, not 1, 3, 5, etc.). Input or outputs may be implicit in the channel address number, or specified via a format. The interface unit is required to distinguish between input and output channel addresses, to determine if data is to be sent back.

Channel addressing is the simplest function code to implement and allows the greatest flexibility. However, it can be very inefficient if channel addresses are not assigned in a way which can be effectively utilized.

b) Functional Classification of Interfaces

In this method interfaces are functionally classified and a code for each class or subclass is defined. For example, all communications can be functionally organized into the following categories: commands, moding, functional input, functional output, and others. The functional categories are assigned a coded number and all interfaces are assigned to a category. A function code would then involve input or output of all data in the corresponding category. Obviously each major category can be further subdivided into subclasses by extension of the function code field. A significant advantage of this method is that the efficiency of information transfer can be much higher if information is generally transferred in a block. It can also be useful from the computer's point of view, since all data in the "functional group" may be desired at the same time (e.g., all status information).

c) Memory

The final approach involves a small memory, of a few hundred words. The function code specifies a location in the memory which contains instructions for data input and output. The memory could store channel addresses or sequences corresponding to an interface function. A memory with a read/write capability could be altered inflight to accommodate changes to a subsystem's operation demanded by different mission phases.

A small high speed memory of the read/write or read only type described above is well within the state of technology. This concept provides the most general and flexible capability, although it obviously increases the complexity of the EIU. Memory size could be expanded to accommodate increases in equipment requirements,or to extend the terminal capability to provide functions such as limit checking of data, or the monitoring of LRU status. Ultimately the terminal becomes a small computer capable of providing a local service to the equipment and thereby reducing bus traffic.

4.3 Operation and Control of the Data Bus by the Computer

Viewed from the computer the data bus is a single, relatively high speed, asynchronously operable, peripheral I/O device, capable of performing data gathering and data distribution. Under the command response access concept, the computer initiates and directs I/O operations on the data bus. It directs I/O by commanding the bus control unit with a set of I/O requests. The BCU then controls and synchronizes the data bus system to carry out these requests. Most likely, the bus system will be mechanized in a way which allows the bus to operate independently of the CPU once an I/O command is issued by the computer. This means that the data bus system and computer operate asynchronously.

4.3.1 Overview of Computer I/O Operations

There are two basic approaches to the design of the computer software for controlling the activities of the bus. These are identified and treated in greater detail in Appendix C than is required for the purpose of this section. The first is the synchronous, fixed I/O method, in which I/O control is based on a predetermined execution sequence and a fixed time cycle. The second schedules I/O operations on a demand basis. The characteristics of the two are summarized in the following sections. To a large extent the computer executive and I/O

control structure can be considered independently of the control structure chosen for the bus.

4.3.1.2 Computer I/O Operation in a Synchronous Structure

Fixed sequence structured software requires I/O operations to be interleaved with processing tasks in the minor cycle. The inputs required by processing tasks in a minor cycle must be available prior to execution of the minor cycle.

The concept requires commanding the BCU (or dispatching I/O), each minor cycle to input data required for the "next minor cycle", and output data from the "last cycle". I/O software for controlling the data bus is operated in each minor cycle. For example:

Bus Activity	Inputs for processing during N Outputs from N-2	Inputs for processing during N+1 Outputs from N-1	Inputs for processing during N+2 Outputs from N
Computer Activity	Process inputs from N-2 for output during N	Process inputs from N-1 for output during N+1	Process inputs from N for output during N+2
Minor Cycle	N-1	N	N+1

The dispatching of an I/O command list to the BCU can occur at the beginning of each minor cycle. However, it is necessary that the list of I/O be completed by the bus system prior to the start of processing the next minor cycle. Thus, the bus will be operating for only a portion of the minor cycle at a percentage of its speed. For example, the BCU may be commanded for 16 ms of I/O every 20 ms. In this case there would be 4 ms idle bus time unless the BCU were commanded again to perform some additional I/O on checkout functions.

At the beginning of each cycle I/O commands are checked for errors. If no errors have occurred, the next I/O list is sent to the BCU and computer commences its processing sequence. If I/O errors occurred, an error recovery and fault isolation routine must be operated and the sequence of processing tasks re-scheduled accordingly. Prior to the end of the minor cycle I/O scheduling is operated to set up the I/O command list for the next dispatch to the BCU.

Since much of the Shuttle data bus design conducted to date has postulated this philosophy of software operation, it will be assumed for the description of BCU activities in the following sections.

4.3.1.3 Computer I/O Operations in a Demand Structure

The alternative approach to fixed sequence I/O is scheduling I/O operations on a demand basis. Typically, this is accomplished in asynchronously controlled software structures as follows:

a) when an I/O request is made by the computer software, control is transferred to an I/O scheduler, and a command is inserted into an I/O queue.

b) The task requesting the transfer is placed into a "wait state".

c) Upon availability of the I/O device, the queued I/O requests are processed via the dispatcher which uses an algorithm, e.g., first in/first out (FIFO), to determine which I/O request to service next.

d) The I/O requests are sent to the BCU one at a time, or in a list for bus execution.

e) When the I/O request has been serviced, the issuing task is informed and allowed to continue.

This approach is used on large ground-based systems, particularly where I/O requirements are not known or impossible to predetermine. The demand I/O concept does not appear consistent with command response or fixed sequence scheduled processing tasks. However, if a distinction were made between computer input and output requests, output requests because of their independence of processing tasks may lend themselves to demand scheduling.

4.3.2 Computer to Bus Operations

An evaluation of the requirements of the interface between the computer software and BCU is directly dependent on the design of the BCU. There are obviously tradeoffs between complexity in the BCU hardware design and the computer software. The BCU in an extreme case could become a computer itself, dedicated to communications functions, supplying all communication of data in and out of the bus system. At the other extreme, it could simply perform time synchronization, transmitting and receiving control, and error coding. Somewhere in the middle, the basic BCU capabilities can be extended by providing the BCU with a limited set of registers and logic, and a

direct memory access (DMA) interface to the computer's memory. By cycle stealing from the computer, the DMA can supply commands and data to the BCU directly from the memory. Commands and data are sent to the BCU either by incorporating a starting address and the number of commands into the channel command word, or by chaining commands and instructing the BCU via the operation code in each bus command. A limited capability will be assumed for purposes of this discussion, although comments are made on areas where an expanded BCU capability may lessen the software problems. The basic computer-to-BCU operations are the following:

a) I/O dispatching - involves commanding and controlling the BCU with I/O to be performed.

b) I/O scheduling - involves scheduling bus commands to be issued the next minor cycle.

c) I/O error processing - checking previous I/O commands issued for errors and taking appropriate action.

4.3.2.1 Dispatching I/O: Computer/Bus Interface

The BCU is provided with a list of I/O commands by loading an I/O channel with a command word from the computer (see Figure 4.2). The channel command word must contain sufficient information to enable the BCU to execute all the appropriate I/O commands in the list. Once this channel is loaded, the computer and BCU may operate independently. The channel command word contains an address of the first BCU command, and the number of BCU commands to be processed. (BCU commands may also be linked by address chaining.) The BCU commands can be stored in sequential memory locations, and the list operated on in sequential order by the BCU. Upon completion the BCU can be instructed to interrupt the processor with an I/O complete signal. (Alternatives, more in line with a "no interrupt" policy, can be devised, such as a "BCU busy" signal accessible to the computer enabling it to determine status of the BCU.) In either case, it is necessary to coordinate the asynchronous operation of the computer and BCU so that the computer is aware of the status of the BCU.

4.3.2.2 BCU Command Format

The BCU command format must contain instructions for the BCU to execute the computer's I/O request. A single command will contain four parts: control information for the message, status information, skeleton bus message format, data linkage addressing information.

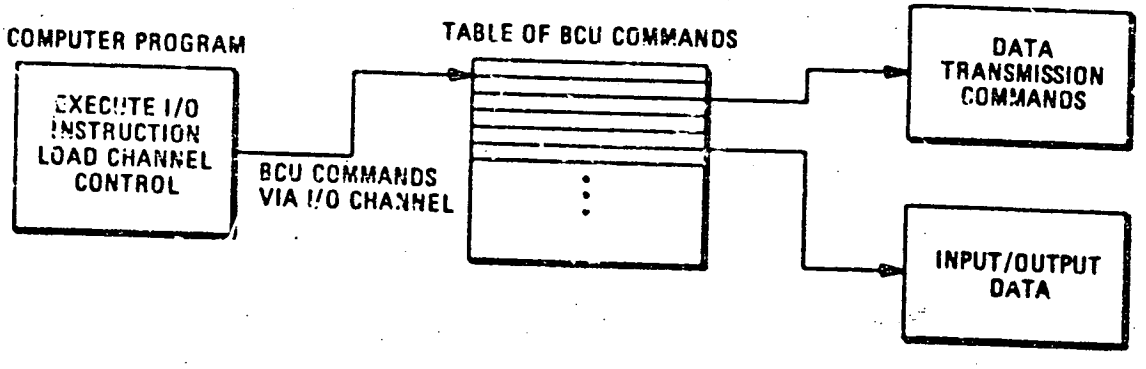


Figure 4.2 Computer to BCU I/O command operation

BCU control op code	I/O status	Bus command			Linkage to data
		SIU #	Function code		

a) Control

The control part of the BCU command contains information pertaining to the type of operation requested of the BCU. Examples of individual BCU operation codes are Read, Write, Skip, Linkage. With fixed I/O tables in the computer's memory, a "no-operation" code may be desirable to skip commands at certain times such as unrequired jet on commands in a fixed I/O schedule. If the BCU contained memory, and was more of a communication processor, this part of the BCU command may contain a pre-programmed BCU memory address for execution.

b) Status Bits

Status bit(s) are required to enable the computer to determine if the bus command was completed successfully. The computer must be informed of bus errors so that it can reconfigure and reschedule accordingly. An incomplete I/O transaction will result in rescheduling the processing tasks. An "incomplete I/O" status indication may also be desirable.

c) Skeleton Data Bus Message

The skeleton bus message contains the actual bus command associated with the I/O transaction. The contents of the bus message format were discussed in Section 4.2.1. It contains information which is both fixed and variable during the course of the mission. Specifically, the terminal addressing will vary with the status of the avionics configuration; a specific communication path must be chosen prior to execution of the command. For example, a request for data from a redundant subsystem (e.g., radar) requires information as to which LRU is active, and which data path to use. It is reasonable to assume that configuration management is a computer software function, and therefore this information must be supplied to the BCU in some form. The degree to which the computer will need to modify the bus message format at run time will depend on the extent and capability of the BCU.

In order to establish fixed I/O command tables required by the synchronous I/O method it may be useful to define a symbolic and "physical" relationship similar to that

used with tapes, disks, etc.,in a conventional facility. In this case a symbolic assignment, such as ISS_A or ISS_S for inertial subsystem active and standby respectively, will be associated with the subsystem. The symbolic identification is then associated via configuration tables to a physical unit such as ISS#1, ISS#2, etc. Predetermined I/O bus commands would be generated using symbolic identification and their physical identification determined at run time by the computer or by the BCU via the transfer tables of the computer. Path identification for a specific physical unit (i.e., which SIU/EIU address) must also be determined dynamically.

If each physical unit had a single path, i.e., a unique address (BUS#, SIU#, EIU#) the problem is solved. However, there is more than 1 path to each unit; the address must be determined from the status of buses and SIU's. The complexity of this problem will, of course, depend on the redundancy interfacing and cross-connections established in the system. For example, consider a system configuration of a quad-redundant bus, 4 SIU's, and up to 4 EIU's per SIU. There could be up to 64 possible paths depending on the cross-strapping.

Physical Unit	Bus	SIU	EIU
LRU #1	1	A	X
	2	B	Y
	3	C	Z
	4	D	W

If the SIU is an extension of the bus such that SIU_A cannot be addressed via bus #2, then there are 16 possible paths to a specific LRU. If the SIU were cross-strapped to the bus and interfaced to a single LRU, then there are only 4 paths to it.

The function of inserting addresses could be allocated to the BCU, assuming it had memory, by sending it a table of physical equipment codes, and the current path. The current path would be updated by the configuration management task as configuration switching occurred.

d) Data Linkage Addressing

This part of the bus command identifies the computer memory location of the data to be output, or the destination of the data input from the bus. If the bus format allows

block transmission, then the number of words is variable, and must be obtained from the bus message itself.

4.3.2.3 Computer I/O Error Processing

An unsuccessful I/O transaction detected by the BCU during bus operations is eventually communicated to the computer, using the error control bits in the bus command table. If the BCU is commanded with a list of I/O requests, an I/O error will not be detected until the start of the next minor cycle. At the beginning of each minor cycle, the error status of all messages is checked. If errors occur, the minor cycle task schedule is modified accordingly, and the I/O error recovery procedures are initiated. Design of the I/O error recovery software procedure is not within the scope of this study; some of the alternatives are:

a) the I/O request could be rescheduled via an alternate path. A reconfiguration of equipment may be required.

b) Fault isolation tasks could be initiated to determine what to reconfigure (the BCU, SIU, or subsystem may have failed).

c) The sequence of tasks contained in the following minor cycle must be altered, delayed entirely, or allowed to continue with "old" data.

4.3.3 I/O - Processing Memory Conflicts (Buffering and Interlocking)

Independent operation of the bus and computer can result in a conflict over the access to common data. This problem occurs when a processing task is using data while the bus control unit is at the same time attempting to input or output the same data for the same memory locations. The problem is more likely to occur for data that is sampled at a high frequency, when use of the data cannot be easily synchronized. It is also more likely to occur in a block of data rather than a single word because of the inherent interlock of a single word access. For example, attitude angle information from the inertial unit may be in use by the digital autopilot task when the BCU inputs new values via the DMA. In this case the autopilot is operating on partly new and partly old values. This problem can be avoided by several approaches:

a) the I/O input and output in this category can be buffered into different memory locations. It may be transferred to other locations, or a pointer can be switched between two sets of registers for the data item, one set for I/O, one for processing. Input data may in any event require to be smoothed or compensated prior to use. This is the general concept of "double buffering" of input or output.

b) The data could be interlocked via a control indicator or busy bit, during the time either the BCU or the computer is using it. However, this would require the BCU to access, test, set and release the indicator with a consequent increase in its complexity.

c) I/O can be planned by predetermining and adjusting the sequence of I/O commands to avoid the conflict. I/O commands can be designed to occur at the opposite end of the cycle from the conflicting processing task. This approach, although consistent with synchronous bus control and I/O philosophies, appears risky due to the inaccurate estimates of timing. It is, in fact, similar to the approach used to solve the memory conflict problem in Apollo. This was only partially successful, and it could only be verified by extensive testing.

4.4 Description and Analysis of I/O Transactions

4.4.1 Definition of an "I/O Transaction"

An "I/O transaction" is defined as the complete sequence of operations performed by the BCU in carrying out a single I/O request from the computer. Once the BCU has received and interpreted a command from the computer, it synchronizes the terminals on the line, transmits a message to the specified terminal and receives the appropriate response. A transaction occurs between the BCU and a single terminal. It is the basic bus communication activity. It is independent of any other transaction over the data bus system. There are two types of I/O transactions that are performed by the data bus: read and write transactions.

a) A read transaction is the sequence of steps performed by the bus system in acquiring data from the avionics equipment. It can be termed a "get" command, to sample a specified LRU equipment interface.

b) A write transaction is a sequence of steps to send data to an LRU interface. It can be described as either a "receive" command, or a "do" command. The SIU receives the data or command and delivers it to the specified equipment interface.

A third type of transaction may be required, termed an "SIU Event Status Command", in which the BCU transmits a command message to an SIU, requesting it to return its event status register.

This transaction enables the computer to determine if random events (interrupts) have occurred at LRU's connected to a particular SIU station. A rescheduling of processor tasks and read/write transactions may be necessary as a consequence of the event.

4.4.2 Functional Description of Bus Transactions

A discussion of how the bus system performs a transaction provides another step towards a specification of the bus/SIU/EIU hardware design. In order to describe the operation of the bus during a transaction, an assumption must be made with regard to a specific bus to SIU to EIU configuration, and an error control approach. It is important to emphasize that this section is intended to describe the functions required at each bus element, and not to select a final design. Several configurations of a standard bus terminal were considered, but a detailed bus command format was only designed for one.

The example configuration assumes a physical separation of SIU and EIU. Each SIU is connected to only 1 bus line and may service up to 8 EIU's. Each EIU provides analog and digital interfaces to equipments. The other terminal configurations assume no logical separation of the SIU and EIU, and are cross-strapped to all four buses. These two approaches to the standard interface unit are discussed more fully in Chapters 3 and 7.

The error control method selected for analyzing the transaction is transmission error detection through vertical and horizontal parity, and path verification by address echo. A detailed discussion of error control philosophy is given in Chapter 5.

A variable number of 8-bit data bytes was selected as the basic transmission format. A 3-byte command format is

selected since 16 bits are considered inadequate to provide the range of addressing and function codes. A minimum of 18 bits are required for the command word in this configuration (7 for SIU address, 3 for EIU address, and an 8 bit function code).

Figure 4.3 illustrates a representative format designed around the 3 byte command message with a variable data message. The asterisked fields are mandatory. Representative use for the other bits in the 3 byte command are discussed below:

*a) SIU address (up to 128 since only one terminal address per station is required. See Section 3.)

b) SIU transaction bit. This bit may be used to command an SIU station to send an event status message. This is a two byte response from an SIU containing the status of 16 events or conditions that are assigned among EIU's at a terminal. Each is set in an EIU by the occurrence of a local random event such as a hand controller movement, display input, or fault occurrence.

*c) EIU address (up to 8 EIU's per SIU)

d) Error control bits. These are sent in an echo message from SIU to BCU when an error occurs associated with the LRU. Typical of the possible error response conditions are:

1) Parity failure at EIU

2) EIU/LRU busy

3) No response by EIU

4) Improper channel

This information could be provided by a special request to the SIU. Making it part of the command format simplifies SIU/EIU logic. If the information were not provided to the BCU, a "no echo" response for all the above conditions will be treated in the same way.

e) I/O control. This control bit determines whether the specified channel address is an input or output operation.

f) Block. This field of the command message identifies a single or multiple channel address group. It is used in conjunction with "Block size" to specify the size of the message block.

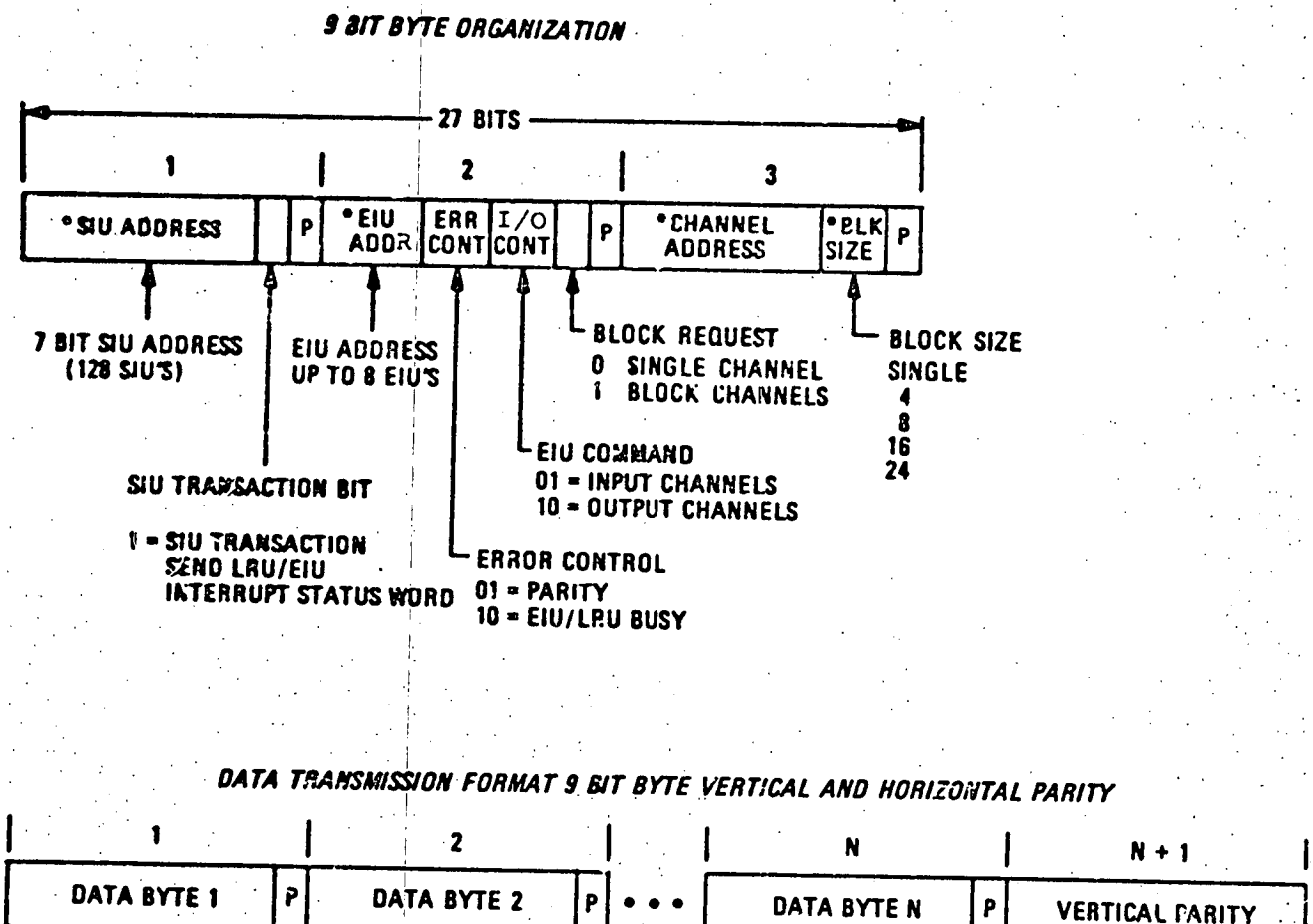


Figure 4.3 Representative bus command message organization

*g) Channel Address. This specifies the EIU interface by one of the methods listed in Section 4.2.1.4.

h) Block Size. The block size identifies the number of channels to be sampled.

4.4.3 Description of the Transaction Sequence

The steps involved in read and write transactions using this format are illustrated in Figure 4.4. A brief description of the transaction is as follows:

a) A read transaction begins when the BCU initiates a sync signal on the bus, followed by transmission of the bus command word. The BCU then waits the response.

b) All "up" receivers on the line receive the sync signal. Each compares the SIU address in the message with its own prewired address. If no match occurs the rest of the message is ignored, and then each SIU monitors the line for the next BCU sync.

c) If the address check shows agreement, the SIU decodes the EIU address and then routes the message to the specified EIU over a serial channel*, while checking for horizontal parity in each byte.

d) The SIU awaits the parity check signal from the EIU to insure that the message was received properly, and upon its receipt, transmits an echo message to the BCU. If the EIU does not accept the message, the SIU transmits its address echo with the appropriate error control bits set in the second byte of the command word.

e) During the time the SIU is transmitting the return echo, the EIU decodes the function code (channel address or memory), multiplexes the requested input channels, performs A/D conversion if required, and sends the requested data to the SIU. A time lag is incurred by this process, termed the LRU latency. It is discussed below.

f) The SIU verifies parity and continues transmitting the data message to the BCU.

---

* Serial transfer is considered advantageous in minimizing the number of interconnections.

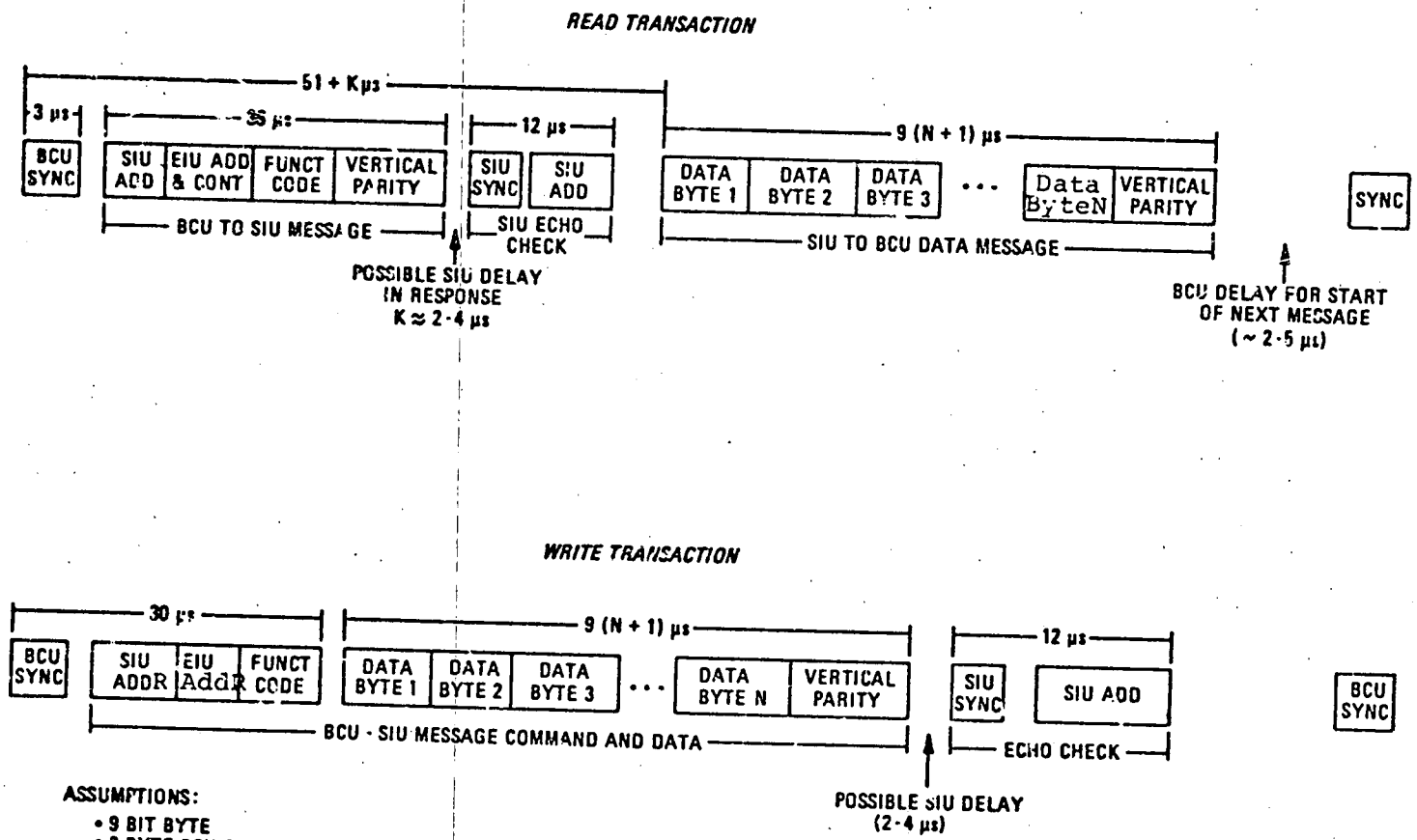


Figure 4.4 Sample read/write transactions

The BCU, after transmitting the initial command, monitors the line for the return echo. If no echo is received within a fixed time interval, a transmission error is deemed to have occurred, and the computer is informed via the I/O error control.

When the BCU receives the echo check, it accepts the requested number of data bytes, verifies parity, and transfers the data to the requested locations in computer memory, after which the read transaction is completed.

Write transactions are performed using similar procedures as illustrated in Figure 4.4. A total time to complete an I/O transaction using this command structure and error control procedures has been estimated for a block of size N bytes to be approximately:

$$\text{WRITE transaction} = (59 + 9N)\,\mu s$$

$$\text{READ transaction} = (69 + 8N)\,\mu s$$

4.4.4 Bus Efficiency and Latency

4.4.4.1 Efficiency

The bus utilization efficiency can be computed by the ratio of information bits in a transaction to the total number of bits in the transaction. If we consider the total number of bits in a transaction to be the total transaction time (including delays, etc.) times the bus speed (assumed to be 1 MBPS) we obtain a worst case estimate of bus efficiency. Information transfer efficiency estimates for a 3-byte command format are illustrated in Figure 4.5.

The bus system will operate at about a 50% efficiency for transfers of 10 or more bytes. This illustrates the obvious fact that to maintain efficiency the software should be structured to obtain information from LRU's in blocks. For example, status data should be obtained in functionally related groups, such as all temperature readings.

A significant factor is the number of I/O transactions that the bus can complete in a minor bus control cycle. Figure 4.6 contains a plot of the I/O transactions, consisting of a given number of data bytes, which can be completed during a fixed interval of time. Based on an average block of length 8 data bytes, approximately 70 transactions can be completed during a 10 ms interval. It is apparent that even though the

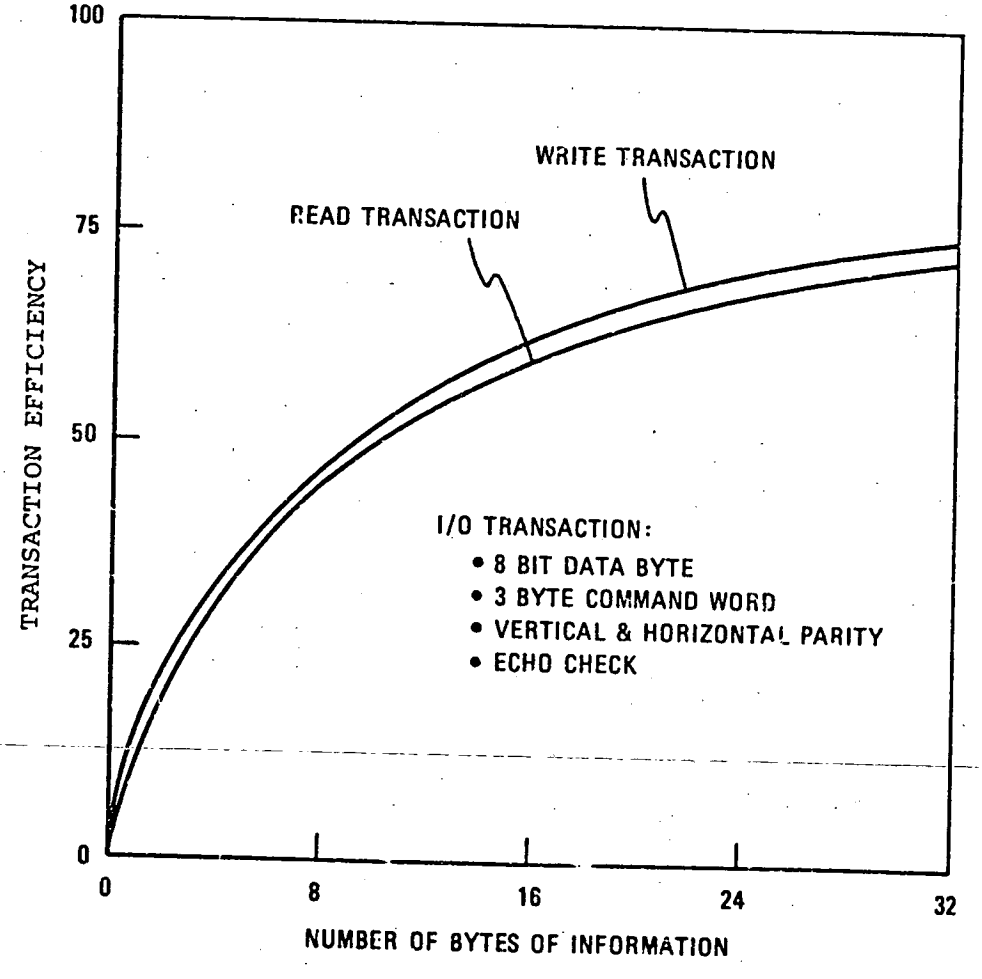


Figure 4.5 Bus I/O transaction efficiency

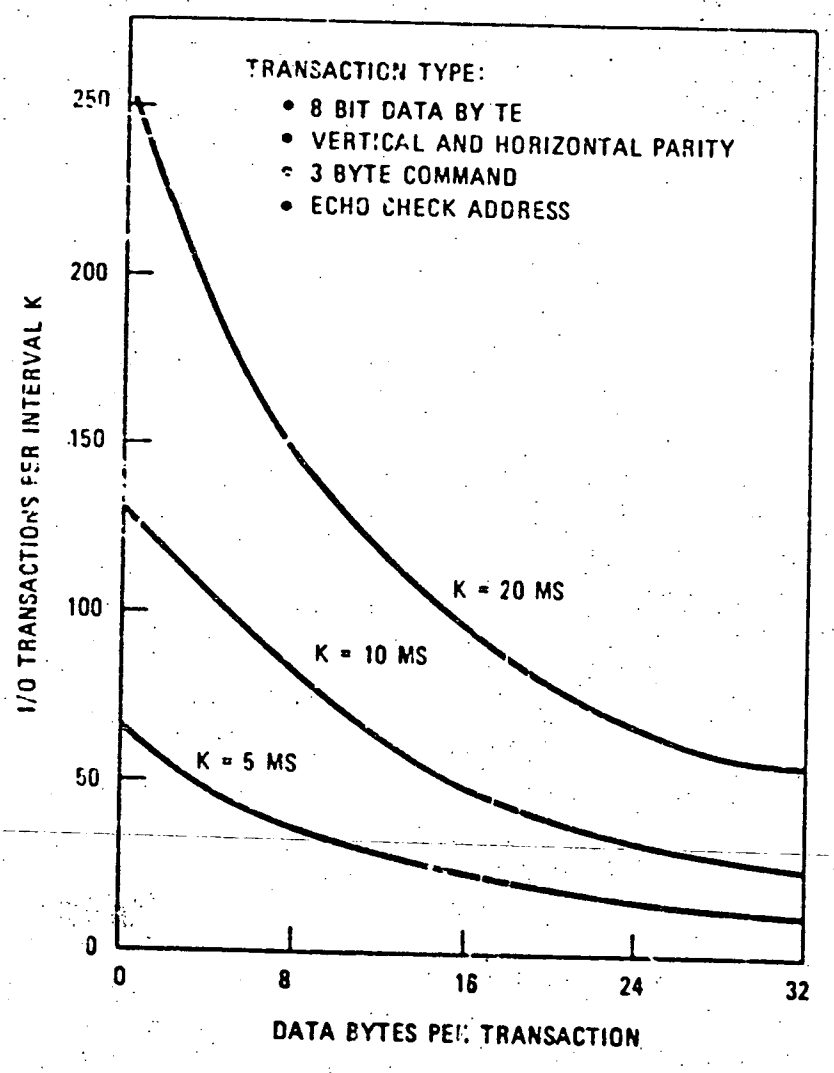


Figure 4.6 Frequency of I/O transactions versus number of data bytes

efficiency of information transfer may be less than 50% in most cases, the actual number of transactions completed during an interval of time should be adequate to service the expected Shuttle I/O requirements. Figure 4.6 illustrates that careful scheduling of the bus during any minor cycle will be required, particularly if the size of blocks vary.

4.4.4.2 Subsystem Latency

When a read transaction command is received by the EIU, an interval of time is required, called the latency time, for the EIU to interpret it, to carry out the command, and return the data. A delay can be caused by analog-to-digital conversion, serial/parallel conversions, inherent equipment dynamics, etc. If an I/O request from the computer has a latency time exceeding a certain fixed interval, it must be organized into two or more transactions. An example is the computer request for DME transponder range. The inherent characteristic of the DME (see Appendix B) is that to obtain range to a specific point, the DME measures the time a signal takes to traverse the distance to that point and back again. The latency time required for this operation is intolerable in the I/O transaction structure described above. This type of transaction must be divided into two transactions: one to command the range to be read, and the other for reading the range. Coordinating these inter-dependent transactions so that they occur at the right time, presents problems to the I/O scheduling software design.

A form of latency occurs for certain types of block data transfer from computer to subsystem. Error control that depends on horizontal and vertical parity cannot provide verification of the correct receipt of a data block until the last byte has been received (the last byte is, in fact, the vertical parity byte). To prevent erroneous data from being transmitted to a subsystem, the complete block must be buffered at the terminal until it is verified. It is subsequently transmitted to the subsystem for which it is intended. However, this second transmission may take a considerable time, by bus standards: a 32 byte block will take over 0.25 milliseconds at 10^6 bits per second. This is enough time for several other transactions to take place.

For both kinds of latency, it is essential to allow no inadvertent interference with the terminal from other transactions. For this reason it is desirable to provide for the indication of an EIU/LRU "busy" condition via the status bit(s) associated with the SIU echo return. This bit can be interrogated by the BCU to provide an I/O error indication to the computer whenever another command is addressed to the busy terminal.

4.5 I/O Timing Difficulties

A class of system problems exists in the operation of a time shared bus which is associated with the correlation of data and commands with "time". For example:

a) Correlation of data and absolute time. Several system computations demand the acquisition of data from separate subsystems at the same time. For example, a navigation measurement combines sensor data with attitude information, correlates both to the same absolute time, and updates the navigation data. With a synchronously controlled data bus, in which sampling is performed only at fixed minor cycle intervals, time may only be established with a granularity of the sampling period. That is, all samples taken during one minor cycle are associated with the same time tag. If a finer time reference is required it must be provided by a local clock. In an asynchronously driven bus system a finer reference time quantization may be obtained because a specific I/O command may be serviced within approximately 100 μs (depending on the I/O queue backlog).

   A related processing problem arises in the derivation of a rate of change by differencing two measurements. In this case a difference in time must be either assumed or computed for two measurement samples. For high frequency samples obtained with a synchronously driven bus, the order of the I/O command in the list may be important, particularly if a fixed delta time is assumed in the calculation.

b) Local precision timing. Another problem that may arise concerns the precision timing of events at geographically separate and remote subsystems, for example, the timing and coordination of firing commands to the RCS jet thrusters. From a system point-of-view, it is desirable to design such subsystems to receive a message which contains not only the command but also the firing interval. The impact on I/O complexity, bus traffic and response, of separate transmissions to command the thruster on and then off could be considerable if this type of bus activity predominates. The capability for local precision timing may be incorporated into the subsystem or terminal.

Chapter 5

Data Bus Error Control

5.1 Introduction

Since the Shuttle data bus provides the sole communications for onboard avionics equipment, an important design requirement is that it provide a reliable transfer of information in the presence of both permanent and transient failures. Permanent failures are caused by equipment failures and are a direct function of the simplicity and reliability of the data bus system elements (i.e. BCU, bus, SIU, EIU, and LRU). Transient failures are caused by such effects as electromagnetic interference, which must be anticipated in the Shuttle environment. The characteristics of the interference are anticipated to be predominantly impulsive, and primarily caused by coupling to the line of transients and noise from switches, motors, relays or other sources. "Burst errors" involving multiple errors close together are to be expected in this environment. A major task of the data bus design will be to incorporate an error control approach which provides "security" of communication in the presence of noise of largely unknown characteristics.

Several error control techniques have been applied in communication systems to reduce the probability of undetected errors. The techniques generally attempt to satisfy a probability goal within the system design constraints of cost, weight, power, or bandwidth.

There are two basic objectives of the shuttle data bus error control scheme to be satisfied in the presence of potential permanent and transient errors:

a) To maximize the probability that a transmitted message is correctly received by the correct terminal;

b) To minimize the probability that an incorrect message is received.

Most commonly a particular error detection scheme has been coupled with retransmission or forward error correction. Various

forms of information coding to obtain an error detection and/or correction capability have been used. Numerous codes have been devised to satisfy a particular communication channel error probability. Prior to discussing the specific error control approach appropriate to the shuttle data bus, a review of information coding schemes is presented with a discussion of their advantages and disadvantages.

5.2 Information Coding Review Discussion

5.2.1 Coding Theory

Coding modifies the message to be transmitted by adding redundant bits to the transmitted message. These extra bits are examined at the receiving terminal to determine whether an error has been introduced and in some cases to locate the error bit within the message so that it can be corrected.

The methods of detecting and correcting errors can most easily be explained with the aid of the concept of Hamming distance. Briefly, the Hamming distance between two strings of binary symbols (of equal length) is the number of positions in which the symbols in the string are different. Thus, the symbol strings 1100 and 1000 are separated by a Hamming distance of 1, while 1100 and 0011 are separated by a distance of 4.

In the study of codes, one of the parameters of interest is the minimum Hamming distance between any two valid code words in the set (for codes in which all the code words contain the same number of bits). Thus, if a code has a minimum Hamming distance of two between any code words, at least two symbols must be changed in order to change one valid code word into another valid code word. With such a code it would be possible to detect any single symbol error, and also many but not all, possible errors affecting more than one symbol.

5.2.2 Single Parity

A common example of such a code is the single parity check, in which the code word is generated from the binary message string to be transmitted by adding a single bit such that the total number of "1's" in the code word is even (or odd). The choice of even or odd parity has no effect on the random error correcting properties of the code, and is usually made to facilitate the detection of certain equipment failures which can produce all "1's" or all "0's" in the received message.

In particular, errors affecting an odd number of bits will be detected but errors affecting an even number of bits will not. The single parity bit is extensively used for error control, principally because of its simplicity in terms of hardware. It is effective against random independent noise.

5.2.3 Error Correcting Codes

For some applications, the mere detection of an error is not sufficient. It is necessary to determine from the received symbol string the nature of the error, or, to be more precise, to determine the message that should have been received in the absence of noise. This can be achieved by error correction codes.

5.2.3.1 Hamming Single Error Correcting Code

The well-known Hamming single error correcting code is an example. This is a code having words of length 2^m-1 where m is any integer. There are m parity bits and 2^m-1-m information bits. The construction of the code word from the message bits will be illustrated for m=3.

Bit Position	B_1	B_2	B_3	B_4	B_5	B_6	B_7
Parity-Message	P_1	P_2	M_1	P_3	M_2	M_3	M_4

The parity bits are determined from the equations:

$$P_1 + M_1 + M_2 + M_4 = 0 \text{ (or 1)} \quad \text{(modulo 2 additions)}$$

$$P_2 + M_1 + M_3 + M_4 = 0 \text{ (or 1)} \quad "$$

$$P_3 + M_2 + M_3 + M_4 = 0 \text{ (or 1)} \quad "$$

At the receiver, the three parity equations are checked to give three error states E_3, E_2, and E_1. (A "1" denotes that the equation did not check, and a "0" indicates that it did.) These three error bits are ordered as a binary number $E_3E_2E_1$, called the syndrom, which equals number of the message bit that should be changed.

If two or more errors occur in the transmission, then either the received word passes the parity tests and is incorrectly accepted by the decoder, or the decoder recognizes that an error has occurred but incorrectly identifies the nature of the error and incorrectly "corrects" the received message.

The Hamming codes that are discussed here have the interesting property that every possible received word is within the error correcting distance (in this case a "sphere" with a "radius" of a Hamming distance 1) of some valid code word. A code having this property is called a perfect code or a close packed code [1]. In general, most codes do not have this property. In fact, for codes capable of correcting more than one error, only a few such codes are known.

5.2.3.2 Augmented Hamming Codes

In the case of non-perfect codes, several strategies can be used when the received message is not within the specified correcting range of any valid code word. On one hand, the distance to each valid code word can be determined and the nearest valid code word selected for the decoder output. If two valid code words are equidistant, outside knowledge of the message probabilities could be used to resolve the tie. At the other extreme, any received message not within the assured error correcting range of the code could be labeled as a detected but uncorrectable error.

An example of a code for the latter strategy is the augmented Hamming code generated from the Hamming code described earlier by adding one additional overall parity bit. This code has a minimum distance of four, and, while it is not a perfect code, every possible received sequence is within a Hamming distance of two of one or more valid words. This code can be used as a single error correcting, double error detecting code.

It is worth noting that a particular code can be used in a number of different ways, depending on how the decoder is mechanized. The extended Hamming code will detect some but not all higher order errors (and will "correct" some other high order errors to produce a wrong message). The same code could also be used as a triple error detecting code. In this case, the code will also detect many more of the higher order errors. In fact, it will detect any error pattern that does not convert the transmitted code word to another valid code word.

It has also been shown that this same code can correct all single errors and also all double errors in adjacent bits, provided

the parity bit is not in error [2]. Using this decoding procedure very few if any nigher order errors will be detected.

5.2.4 Higher Order Error Correcting Codes

Codes are known which have sufficient Hamming distance between valid words so that they can correct two or more errors in a block. In general, these codes are either trivial (repetition of each message bit an odd number of times with majority voting, called a binary repetition code), or are too complicated to describe in detail here.

Among the better known of the constructive (non-random) codes are the Reed-Muller codes [3], and the Bose, Chandhuri and Hocqueughem (BCH Codes). BCH codes are a generalization of Hamming codes for multiple error correction. The correction procedures are, however, fairly complicated. The technique for BCH error correction consists of solving the roots of a N degree polynomial and a set of N equations, where N is the number of correctable errors. The complexity of the correction process forces BCH codes to be considered only for error detection. Correction becomes feasible if a processing capability is available, and a delay in the receipt of the message is acceptable. BCH codes are cyclic codes and have the disadvantage of being sensitive to loss of synchronism since shifted cyclic code words are also valid code words.

5.2.5 Burst Errors and Burst Codes

In many instances where coding has been employed to detect or correct random errors in a data transmission system, the improvement in system performance has not been as great as expected. The reason is often that the assumption of additive white gaussian noise, or other mechanisms which generate independent bit errors, is not valid. Generally, in a real environment the errors occur in groups or bursts. Electromagnetic interference of duration longer than one bit transmission time would be an error source with this characteristic.

A simple example is provided below to illustrate such a problem. Consider the case of a system operating at one million bits per second, and using coherently detected amplitude modulation at 15 db signal to noise ratio. We will assume that the system is perturbed by gaussian noise so that errors are random and independent. The probability of a bit error for this condition can be calculated to be one in 1.26×10^8 bits. The code is a three error correcting code having 23 bits, with 12 of them information. The example is a special case known as the

Golay code. This code is close packed, and we can, therefore, neglect all of the possibilities of detecting higher order errors as they always result in a word error. The following observations are made:

a) a single bit error in a word is expected with probability $23 \times 7.9 \times 10^{-9} = 1.8 \times 10^{-7}$ per word, or once every 126 sec.

b) a double bit error will occur with probability 1.6×10^{-17} or once every 47.5 years.

c) the probability of three or more errors and consequently the probability of an undetected error in a word is vanishingly small.

If, however, the mechanism of the disturbance is such that for 10 consecutive bits the probability of error is 0.5, there will be an average of 5 errors in the burst of ten bits, so error bursts will occur every 630 seconds. Since .17 of these bursts will have three or less errors, and neglecting the fact that in some cases a burst laps over the division between two blocks, a decoding error will occur approximately every 25 minutes.

The description of the burst error channel given above is obviously a very simple case. Yet it illustrates the significant difference in conclusions which can be drawn about the expected performance of a control approach.

Some general observations can be made on the performance of error control codes in the presence of burst noise. If a code with a minimum Hamming distance of h is used as an error detecting code, any burst causing up to (h-1) errors will be detected. For bursts causing more than (h-1) errors, most, but not all, will be detected. The exact percentage of errors of various lengths that will be passed depends on the code used.

At the other extreme, if the burst is sufficiently long and severe, so that the received bits have no correlation with the transmitted message but are instead received with a probability of error of 1/2 for each bit, then an estimate of the probability of passing an error is again possible. If the coded word has n bits, k of which are information, the remaining (n-k) bits are redundant. The k information positions in the word can be filled by the random process with any bits, and there will then be one and only one set of values for the redundant bits that will result in a coded word. The probability of this particular set of values being chosen is $(1/2)^{n-k}$.

The assumption that a noise burst will result in bits being received as "1" or "0" with probability 1/2 is, however, not always

valid. Sometimes a noise burst (or hardware failure) is more likely to cause errors in one direction, such as turning "1's" to "0's", than the other direction. Such situations arise from the details of the modulation scheme used and the design of the hardware, and are very difficult to evaluate in a general way. When possible, it is usually good design practice to design the code so that the most likely types of equipment failures will not result in a valid code word. Examples of this would be elimination of all "1's" and/or all "0's" as valid code words.

5.2.6 Fire Codes and Other Burst Codes

Some special error correcting codes have been developed which are especially applicable to error correction in channels which are subject to burst errors. For a given level of redundancy, these codes are able to correct more errors in a burst than would be possible if the errors were assumed to be random. These codes require long blocks and complicated decoding procedures. Two examples of these codes are cited:

a) Fire Codes

Fire codes are oriented towards a single burst of errors per message. They are inefficient for short blocks, however, and are not particularly good for multiple bursts on a single block.

b) Reed-Solomon Codes

The Reed-Solomon codes are a special case of the generalized BCH codes, oriented toward multiple burst error correction. They are moderately efficient, and for the same block length are similar to BCH codes in decoding complexity.

5.2.7 Horizontal and Vertical Parity Coding

A coding technique which has been proposed for the Shuttle baseline data bus systems is vertical and horizontal parity coding. This coding scheme assigns a single parity bit to each byte or word of the message (horizontal parity), and an extra byte or word for vertical parity on the preceding bytes. This approach detects all odd numbers of errors. An undetected error can only occur when each byte and every bit position contains an even number of errors. The scheme fails to detect errors only when an even number of errors, equal to or greater than four, occurs with the errors paired in rows and columns. The efficiency of this approach is moderately high for messages of several bytes,

but is poor if the number of bytes of data in a message is small. For example, the effective information rate of an 8 bit byte of data would be computed by

$$E_{IR} = \frac{8N}{9(N+1)} \qquad \text{where N is the number of bytes}$$

It can be seen that for a small number of data bytes the efficiency is low (i.e. 44% for 1 byte, 59% for 2 bytes). When the block size increases, however, the coding scheme becomes more efficient (i.e. 79% for 8 bytes, 91% for 32 bytes). Although there are more efficient coding techniques, this scheme has a major advantage in that its implementation in terms of the encoding, decoding and detection logic required in the SIU, EIU, and BCU data bus equipment is probably the simplest.

5.2.8 Repeated Transmission

The repeated transmission of a data message over a single path is a well-known method for error detection. Detection is accomplished by requiring all messages received to be identical. The time diversity, or spacing of transmissions provides independence.

Implementation of this approach as the prime error control approach in the Shuttle data bus would require the BCU to transmit the (uncoded) data to the remote station, and vice versa, two or more times. The remote terminal would require a comparator or voter to determine an "acceptable" transmission. Retransmission for error correction is still required for ambiguous voting results.

The method is relatively simple to implement, but is very inefficient, particularly for block transmission. In order to get a Hamming distance four code for three error detection, the message must be repeated four times. The same error detecting capability can be obtained with many fewer bits using other coding schemes.

5.2.9 Transmission Over Multiple Paths

The transmission of the message over multiple separate paths between a single BCU and single LRU is similar to the redundant transmission over a single path. It is true that the message is received and verified at the output with less delay than is associated with the sequential trarsmission scheme, but

on an overall basis, there is no improvement in the utilization rate of the available channel capacity. The necessity of providing parallel channels to allow continued operation in the event of a permanent hardware failure would directly affect the Shuttle data bus if it were the prime error control method used. It would require independent paths to be maintained for the FS mode of operation, increasing the number of buses required for FO/FO/FS.

The approach would increase the complexity of the BCU and SIU units, since it requires transmissions over multiple paths to be synchronized, so that comparison or voting could be performed at the receiver, or storage for delayed receipt.

5.2.10 Data Feedback/Echo Check

In this method, uncoded data is saved in buffer storage at the transmitting element and sent to the receiver. The receiving element transmits back the entire message. The transmitting element then performs a bit-by-bit verification of the entire message. Upon verification by the transmitter, the receiving element is instructed to use the information on receipt of a "verify" message from the transmitter.

If an error is detected the transmitting unit can retransmit the entire message. If the error was caused by an external noise transient, the second transmission should be valid. This method is referred to as an echo. One of the problems with this approach is the probability of transmitter's verification being in error. An endless chain of echoes may result in requiring the receiver to echo the echo, etc. Complete feedback of all data requires twice the time to transmit a message. Its main advantage is the high degree of error detection it provides.

5.3 Detection and Retransmission Vs. Forward Error Correction

In the analysis of data transmission systems, two distinct cases have been studied. The first case is Forward Error Correction, in which the decoder at the receiver studies the received message and, if an error is discovered, attempts to deduce the correct message from what was actually received. The second case is retransmission, in which the decoder checks the received message for signs of error, and if an error is detected the decoder informs the transmitter. The transmitter can then retransmit the message or take whatever other action is indicated.

A forward error correction scheme is considered undesirable for the Shuttle data bus since it would require too much complexity

at the terminal and BCU, particularly for correcting more than 1 error in a message. The method preferred is to combine an error detection scheme with retransmission for recovery.

The advantages of the retransmission approach to error recovery are reduced complexity of the decoder and the reduction in the probability of an undetected error for a given level of coding.

The classic studies of retransmission systems were reported in two papers by Benice & Frey in 1964 [4]. In these papers, three cases were considered:

1. Idle RQ - in which the transmitter sends a message and then sits idle until the decoder indicates whether a retransmission is requested. Presumably, this includes a "no response" from the terminal.

2. Simple RQ - in which messages are sent continuously. When an error is detected and a retransmission requested, the source repeats the requested message.

3. Dual RQ - in which messages are transmitted as in Simple RQ, except that the requested message and all subsequent messages are repeated.

The Idle-RQ system appears to be most appropriate to the Shuttle data bus, since the bus traffic is expected to consist of a large number of relatively short communications between the bus controller and the many terminals along the bus. The advantages of the other schemes are achieved when full duplex transmission systems (simultaneous continuous transmission in both directions) is used. The Shuttle data bus is not expected to be used in this manner.

The conditions for which the Idle-RQ scheme becomes a poor candidate are not applicable to the Shuttle data bus. In many data transmission systems, the transit time of the channel is long compared to the length of a message. Thus, the transmitter wastes a lot of time sitting in the idle state waiting for the message OK or retransmit signal. In the Shuttle data bus, the round-trip time to the farthest subsystem will only be a few microseconds, or bits.

In the data presented by Benice & Frey, the computed probability of an undetected error for the Idle RQ system drops rapidly until a certain minimum probability is reached, and then no further improvement is possible. This behavior is traced to the failure of the retransmission request to be recognized at the

transmitter. The minimum error probability is the probability that some kind of error will be detected in the forward message, and then the retransmission request is changed to a confirmation that the message was OK.

In the other two retransmission schemes, the retransmission request was encoded as a part of a message moving in the opposite direction and was, therefore, protected by the same level of coding as the original message. The occurrence of any error in a returned message was construed to be a retransmission request for the forward message. This attitude results in a small decrease in throughput rate, and a large decrease in probability of an undetected error.

In the Idle RQ scheme, Benice and Frey postulated a one bit confirmation message for most of the work, and this results in a minimum probability of undetected word error of about 5×10^{-8} for a bit error probability of 10^{-5} and a 511 word message. By changing the returned accept retransmit request message to a 7 bit format, the minimum probability of an undetected error was reduced to 5×10^{-38}. The point to be made here is that the retransmit request must be suitably protected if it is not to turn out to be the limiting factor in the probability of error in the transmission system. The penalty for this is a slight reduction in the throughput rate of the system, which does not appear to be a prime consideration in the Shuttle data bus system.

References for Chapter 5

1. Berlekamp, E.R., Algebraic Coding Theory, McGraw Hill Book Co., New York, 1968.

2. Abramson, N.M., "A Class of Systematic Codes for Non-Independent Errors", IRE Transactions on Information Theory. PGIT5, No. 4. December 1969, pp. 150-157.

3. Peterson, W.W., Error Correcting Codes, The M.I.T. Press, Cambridge, Mass., 1961.

4. Benice, R.J. and Frey, A.H., Jr., "An Analysis of Retransmission Systems", IEEE Transactions on Communication Technology. PGCOM-12, No. 6. December 1964, pp. 135-145; and "Comparisons of Error Control Techniques", Ibid, pp. 146-154.

Chapter 6

Bus Implementation Factors

6.1 Transmission Problems

This section will briefly review some of the factors affecting the implementation of a Shuttle data bus. As mentioned in Chapter 2, it is a finding of this study that, although these factors can be of critical importance in determining whether the data bus provides adequate performance, whether it is vulnerable to environmental transients and hardware failures, and whether it minimizes the penalties of power and weight, they are not as significant as those of redundancy management, control policy and overall configuration.

The major factors that impinge on the hardware design decision are:

a) the specified performance requirement. This is, of course, the prime design driver. There appears to be a natural breakpoint in the design of the system at a data rate of about 10^6 bits per second, especially for the transmission medium and the choice of integrated circuit technology. MOS technology is currently unable to operate at circuit speeds below a few hundred nanoseconds, whereas bipolar IC's are faster by nearly an order of magnitude.

b) The expected signal-to-noise ratio of the environment. This is, after performance, probably the overriding factor in the design of the total system. The choice of signalling technique and type of communication medium are, for a given data rate and specified error frequency, driven by this factor.

c) The bandwidth and distribution of the noise spectrum. This factor is virtually impossible to define without real measurements in an actual shuttle vehicle. To formulate design guidelines attempts have been made to characterize it, e.g., as white with Gaussian distribution. From studies of the characteristics of aircraft environments, no simple

noise model is known to be satisfactory. Diverse sources of radiated noise such as actuator solenoids, motors, radars, and the effects of ground coupled current transients are very difficult to characterize in a general fashion. Experience has indicated that noise energy in a realistic environment is distributed as 1/f at the lower frequencies (below 10^6 hertz). Assessments of transmission techniques whose findings depend solely on the performance in the presence of Gaussian noise must be questioned. The approach usually followed is to ensure a high SNR, so that the system is not sensitive to the shape of the noise spectrum.

d) The complexity of the bus transmitting and receiving equipment. The more sophisticated signalling techniques, such as modulated carrier, demand a greater complexity in the modulation and demodulation circuitry. In a Shuttle system, with an estimated total count of upwards of 200 bus line terminals, a minimal complexity at the terminal is of obvious importance.

A real Shuttle data bus design will encounter many difficulties of implementation in coping with the above constraints, and trying to meet the operational requirements of the Shuttle.

Although this study specifically did not address itself to such problems, two areas that have received much attention were examined, namely the transmission technique and the transmission medium. These are now described.

6.2 Non-carrier (Baseband) Signalling Schemes

A great number of techniques for directly encoding a signal for transmission on a channel has been devised in the field of communications. Table 6.1 summarizes some of the major properties of baseband signalling techniques, and their characteristic waveforms are depicted in Figure 6.1.

a) Non-return to Zero (NRZ): Figure 6.1a

A '1' is represented by a fixed voltage level of one polarity, and a '0' by an equal level of the opposite polarity. Detection becomes a simple matter of polarity discrimination, e.g., by Schmitt trigger. A sequence of unequal numbers of '1's and '0's produces a non-zero average component which demands a DC channel capability, with DC coupling or some form of signal restoration at the receiving end. NRZ is vulnerable to noise whose energy is inversely proportional to frequency, as is typical in aircraft electrical systems. Since the signal contains a maximum of one transition per bit the channel bandwidth requirement is low. However,

Modulation Scheme	Advantage	Disadvantage
NRZ	1. Simple detection 2. Bandwidth equal to date rate	1. Requires DC coupling 2. Transmission to zero frequency 3. Requires separate sync
Biphase	1. Sync inherent in data 2. No DC component 3. Zero frequency transmission not required	1. Requires bandwidth of twice data rate
Bipolar RZ	1. Sync inherent in data 2. Simple detection	1. Requires bandwidth of twice data rate 2. Requires DC coupling 3. Requires transmission to zero frequency 4. Requires detection of zero signal level
Bipolar	1. Bandwidth equal to data rate 2. Zero frequency transmission not required 3. No DC component	1. Requires separate sync 2. Requires detection of zero signal level

Table 6.1 Comparison of Modulation Schemes

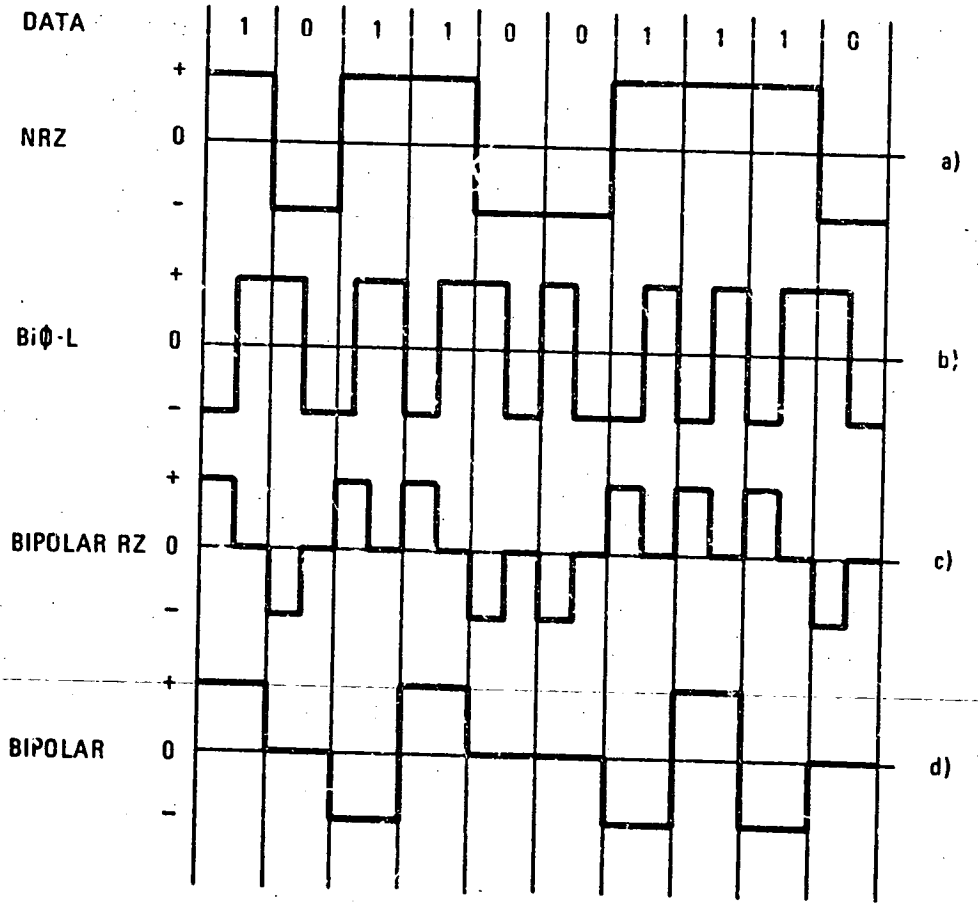


Figure 6.1 Modulation waveforms

because signal transitions occur only when the data changes from '1' to '0' (and vice versa), there is difficulty in obtaining bit timing for long sequences of '1's and '0's. Synchronization must be provided by a clock signal in addition to the data. Since the data frequency is a submultiple of the basic clock, its energy spectrum shows a minimum at the clock frequency (see Figure 6.2a). Effective clock and data separation can, therefore, be achieved by band-pass filtering the clock and low-pass filtering the data.

RZ is similar to NRZ in bandwidth and power spectrum, but has a higher error probability in the presence of noise.

b) Biphase (Manchester): Figure 6.1b

A signal transition occurs at least once during every bit interval, in one direction for '1's and in the opposite for '0's. Detection involves phase discrimination, which is a more complex procedure than level detection. Bit timing is inherent in the waveform, and there is no DC component. However, the frequency of transitions requires approximately twice the channel bandwidth of NRZ, although from Figure 6.2b it can be seen that most of the transmission energy goes into the data at a point in the spectrum which has a lower noise energy than for NRZ. Synchronization can be derived from the signal by detecting the zero crossings, filtering the resulting signal and using it to drive a local oscillator in a phase locked loop. This is considerably more complex than NRZ synchronization. Two alternatives are possible:

1) superimposing a separate clock signal on the data at a frequency well above the data passband, and then dividing it down at the receiver;

2) equipping each receiver with an independent clock of high enough frequency to provide fine grain strobing of the data signal transitions.

The first technique requires a higher channel bandwidth. Both of these techniques present difficulties in clock jitter due to uncertainties in the clock-to-data phase relationship, and in the frequency division logic.

c) Bipolar: Figures 6.1c and 6.1d

In Bipolar RZ modulation '1' and '0' are denoted by positive and negative going pulses, of usually a half bit duration. Between symbols the signal returns to zero volts, which provides the capability for bit synchronization. Detection is

as simple as NRZ. However, the scheme requires a bandwidth of twice the data frequency. Furthermore, it presents a difficulty of all RZ schemes: the necessity for adequately detecting the zero signal level. This is susceptible to baseline wander, and becomes error-prone in the presence of noise. Finally, it requires transmission to zero frequency and contains a non-zero average component.

A scheme that has been termed bipolar modulation but is more strictly a combination RZ and NRZ, is depicted in Figure 6.1d. It aims to achieve the modest bandwidth of NRZ without requiring a transmission to zero frequency, and without a non-zero average component. A '0' is represented by a zero level, and a '1' by a non-zero level, whose polarity alternates with successive '1's. Some logic is necessary after level detection, to extract data from the signal. Its energy spectrum is shown in Figure 6.2d. Like both NRZ and RZ, this bipolar scheme requires separate bit synchronization, and like RZ suffers from zero-level noise problems.

d) Other Techniques

Techniques involving multi-level signalling, and variations and combinations of the above schemes have been proposed to suit the special requirements of specific signalling environments. They will not be further reviewed in this report.

6.3 Carrier Modulation Techniques

In carrier systems the information is impressed on a carrier signal of a nominally fixed frequency, to occupy a band in the spectrum corresponding to a lower noise energy. A greater utilization of channel capacity is possible by operating at a high frequency, with a consequent increase in the transmission efficiency. Transmission media tend to possess better characteristics at higher frequencies; e.g., the variation of signal delay and attenuation across the operating band is less. This becomes an important factor at frequencies in excess of 10^6 bits per second. These are major advantages. However, the drawbacks of carrier modulation schemes are considerable, and must be carefully weighed, even if a severe noise environment indicates their use.

Modulation and demodulation are more complex than for non-carrier systems, since the data and synchronization must first be combined and then stripped from the carrier. It is not possible, as with non-carrier modulation, to use digital circuitry directly to drive and sense bus transmission.

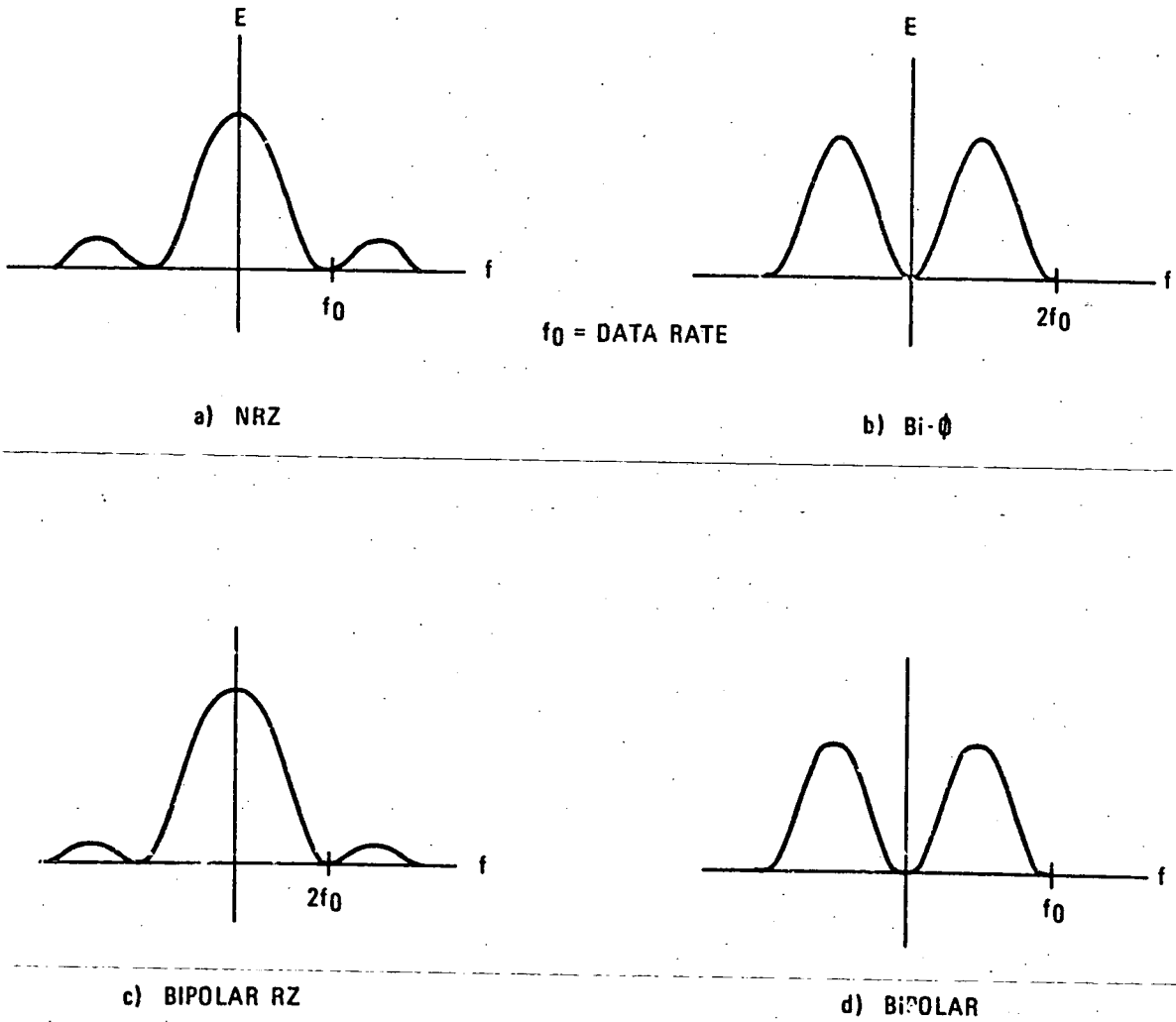


Figure 6.2 Modulation energy spectra

To take advantage of the noise immunity of carrier systems the carrier frequency must be high, up to ten times the data rate. The frequencies generally exceed the ability of the simpler shielded cables, which having less controlled characteristics, are less able to provide unattenuated and undistorted transmission.

A synchronization signal (clock) can be carried with the data, depending on modulation technique, either by combining modulation techniques, by modulating a separate carrier separated from the signal carrier by the bit rate (this facilitates filtering), or by combining the clock in phase quadrature with the data.

6.4 Bit Synchronization

The distribution of timing information is necessary to the operation of the data bus system, whatever the choice of modulation technique. Some of the problems of deriving bit timing from or with the data have already been discussed. In general, it is possible to generate the timing signal either centrally at the computer/BCU, or locally at the terminals. It can be transmitted inherently in the data, or over the same physical path as the data, or on a separate clock line.

a) Central Clock

The central clock signal is continuously distributed to all terminals, which then employ it to re-transmit their responses. The advantages of this technique are:

1) simplicity at the terminal;

2) provision of a common, synchronous clock to all systems on the bus.

Its disadvantages are:

1) a clock failure disables all operations that require timing information, until reconfiguration can occur;

2) data received by the computer/BCU from remote terminals will be phase-skewed with respect to the clock, requiring compensation logic at the BCU end of the bus.

b) Local Clock

The method suffers from the complexity of additional clock circuits at the terminal. However, most terminals will probably require a source of timing independently of their communication function. The advantages of a locally generated clock are:

1) no clock skew problems,

2) no dependence on central clock for local timing.

c) Separate Clock Lines

A separate clock line can be considered for either locally or centrally generated synchronization. The added penalty must be traded against the complexity of superimposing, and then separating data and synchronization. Although the extra line can be dedicated to continuous transmission of synchronization alone, a variation on this approach has combined computer command data and synchronization on a line separate from terminal responses. The resulting dual simplex communication mode has been claimed to effect savings in line coupling and drive and sense electronics, which offset the weight and power penalty of the additional bus line. However, to meet a FO-FO-FS failure criterion the arrangement demands at least five redundant paths, and switching between receive and transmit modes at each terminal, and at the BCU. If there is no switching, up to eight lines will be required.

This mode of operation creates an added degree of difficulty to the problem of error detection and bus reconfiguration. This must be considered along with the weight penalty in making an evaluation.

6.5 Transmission Media

Many techniques for transmitting data over distances of several hundred feet are available. Transmission media range from modulated light beams for data rates above 10^9 hertz, to the conventional wiring of current aircraft and spacecraft, which is limited to a 10^4 to 10^5 hertz data rate. As indicated in Chapter 2, most estimates that have been made of the Shuttle data rate requirement indicate that a 10^6 to 2×10^6 bits per second capability would be adequate. Accordingly, the choice

of transmission medium for the Shuttle data bus can be limited to the several varieties of miniature coaxial and balanced twin cable. Single wire conductors do not possess inherent immunity from spurious electrical and magnetic field disturbances. They are also emitters of interfering electromagnetic fields when carrying high-level signals, unless very heavily (i.e., doubly) shielded. Single conductor wire has generally been eliminated for all but very short cable runs. The basic electrical performance requirements for the transmission paths in the Shuttle are:

a) wide bandwidth

b) low frequency and phase (delay) distortion

c) low attenuation

d) immunity to conducted and radiated noise.

In addition, the Shuttle application makes the following considerations important:

a) cost and availability

b) low weight and volume per unit length

c) resistance to temperature, pressure, vibration, shock, etc.

d) physical flexibility and ability to share cable ducts with other wiring

e) ease of physical and electrical connection.

Since the Shuttle data rate requirement is modest many available cable types can meet the bandwidth and attenuation requirements. The candidates appear to be:

a) twisted shielded pair (TSP)

b) shielded coaxial cable, single and twin

c) flat cable.

Each type exists in various performance grades. The special flat cable is still under development. TSP is lightweight and usable to several megahertz, depending on specification. Coax can be extended to hundreds of megahertz, but becomes very bulky and unwieldy. The least known and least specifiable shuttle vehicle parameter appears to be the nature of the noise environment. A

low SNR environment, with noise energy concentrated into the low frequencies makes the use of the twisted shielded pair in a balanced mode mandatory with any of the baseband signalling techniques described previously, especially those with transmission to zero frequency. Figure 6.3 illustrates the relative noise immunity of TSP and coaxial cables. TSP is superior at the low frequencies where shielding becomes ineffective, because of the mutual cancellation of the individual conductors' induced fields. If the noise environment is severe, a high-frequency carrier signalling technique may become desirable, requiring use of standard single or twin coaxial cable.

A problem for any cable type is the geographical location of the equipments to be interconnected. It will in general be necessary to provide branching of the data bus. Such branches must not introduce undue reflections or attenuation. The total complement of equipment on the bus may not remain fixed. The addition or subtraction of terminals should not alter the basic characteristics of the line. These properties may be difficult to achieve with the less well-balanced and uniformly constructed cable types without line conditioning or the use of active couplers and repeaters. This naturally adds to the complexity and unreliability of the bus, which the use of simple TSP seeks to minimize. However, as will be discussed later, line couplers may be required anyway to effect redundancy interfacing.

The coupling of drivers and receivers to the bus lines has received a great deal of attention. Whether AC or DC coupling be used is determined by the modulation scheme. The noise problem favors the use of balanced line coupling: the inherent immunity of TSP to radiated noise is enhanced by driving and sensing the line in a balanced mode, since this rejects common mode noise created by ground return paths. AC coupling allows balancing through center-tapped transformers. The inductance of the transformer, however, affects the characteristics of the transmission line, which is a more serious problem for baseband than carrier signalling techniques, since the latter permits a degree of tuning to achieve noise rejection. Capacitative coupling has also been used to drive an AC line. DC coupling is achieved through resistive bridge networks.

A factor that may impinge directly on the choice between AC and DC coupling is that an AC coupling will not transmit DC. It can filter out continuous anomalous line conditions such as open circuits, shorts to ground or shorts to a DC supply rail, etc. Such conditions in the line driver of an AC-coupled bus terminal need not incapacitate the complete line to which it is interfaced. By suitable choice of driving and sensing impedances it may be possible to allow continued operation, even when violent impedance changes occur on either side of

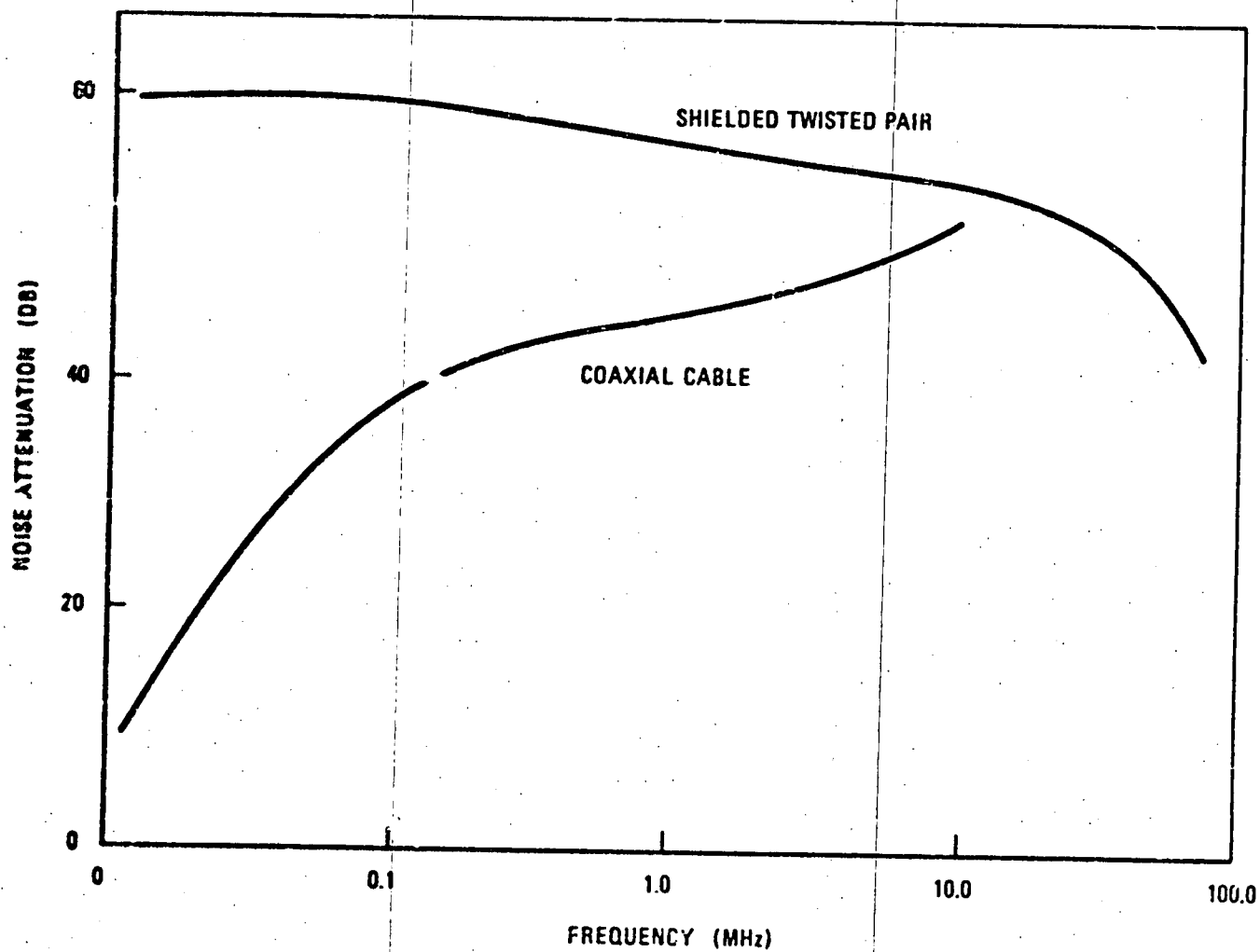


Figure 6.3 Noise attenuation for twisted shielded pair (TSP) and coaxial cable versus frequency

a coupling. The situation could be treated, not as a catastrophic failure, but as a condition of reduced signal to noise ratio. This would be much more difficult with DC coupled modulation schemes, for which the zero level represents an information state.

PRECEDING PAGE BLANK NOT FILMED

Chapter 7

Summary Review and Recommendation

7.1 Introduction

This chapter will review the material presented in the first part of this report and make more specific recommendations in certain areas of the Shuttle data bus design. Section 7.2 reviews the problems of management and control of the data bus, and Section 7.3 addresses the main objective of this study, namely the functional specifications of a data bus interface unit.

In order to make recommendations in areas where there was no clear criterion two general ground rules were established:

a) The simpler solution was preferred. This was interpreted very broadly. Complexity was considered a disadvantage not only at the circuit and logic level, but in control, equipment configuration, interconnections, software, program management, etc. Otherwise desirable attributes such as expandability, performance per se, high technology and so on, were discounted in favor of expediency and cost effectiveness. Cost was not measured in specific terms of dollars; but rather in the degree of difficulty of a given approach. The choice of the simpler approach was considered to minimize development risks and to achieve a higher probability of operational reliability.

b) Great weight was given to approaches that kept functional requirements separated. For example, the provision of capabilities at the data bus interface unit that were not directly concerned with bus communication, but influenced the bus control structure and data formats was not favored. Because of the central role of the data bus there is a temptation to burden it with functions that are more properly

the concern of the subsystem or the computer. For this study the bus was considered to be primarily a communications and data acquisition system, and not a tool with which to tackle general avionics system problems.

In addition to these general guidelines the evaluation of the bus system was directly dependent on the Shuttle system communication requirements and imposed design criteria defined in Chapter 2. In addition to the data requirements of speed, types of messages, number of units, etc., several other important criteria were used in evaluating bus design features:

a) Simplicity and ease in verification. Because of complexity and the degree of reliability specified for the integrated avionics, the data bus system must remain as simple and easy to verify as possible. The elimination of non-deterministic operation is desirable.

b) Efficiency in data transfer was not a prime objective. It should not be an essential requirement to maximize the efficiency of a data transfer or access scheme, provided that operation remains within a reasonable margin of the limit. If overhead is encountered to achieve design simplicity and "testability" it should be accommodated by the speed requirements.

c) Flexibility. In pursuit of simplicity care must be taken to avoid an inflexible system, especially since the Shuttle concept is at an early level of development. However, flexibility in design of the bus was considered only where a potential need was apparent. Speculation on the as yet unestablished requirements of future Shuttle activities was discouraged.

d) Standardization in bus design. A technique which maximized the degree of standardization possible in an implementation of the data bus, or one that involved established system design procedures was favored over an approach unique to the specific problem.

7.2 Command and Control of the Shuttle Data Bus

In this section the basic structure of the data bus system, the functions that it need, or need not perform, the type of control to be exercised, and the type of data structure required will be reviewed and evaluated.

7.2.1 System Configuration

The configuration of the bus with respect to the computers, the bus control unit, the terminal and the subsystem was discussed in Chapter 3. The computer's role as the controlling authority over the operation of the bus was identified as a critical system decision factor. Two aspects combine to make the computer's job difficult:

a) the scope of the centralized approach to Shuttle avionics;

b) the management of multiple redundancy implied by the assumed FO-FO-FS criterion.

Even though the processing task has been estimated to fall within the capability of a single computer of moderate size and speed, the question of dividing it among several computers has been raised. The arguments for division have included ease of management, less costly software production, and ability to develop the system incrementally. The arguments against division are based on the amount of duplicated hardware and software. Chapter 3 concluded that distribution of the computing tasks would be easier if each computer could be associated with a distinct, independent functional area.

In the Shuttle application strict separation of functions is not straightforward, because:

a) it is difficult to identify subsystems (for example, data management versus guidance and navigation) with independent functions and equipment. (Exceptions are displays, main engines, and perhaps the environmental control system, which does not contribute a major proportion of the avionics complement.)

b) The specification of a common data bus forces the need for cooperation between all computers and subsystems that interface with the bus.

c) The high level of redundancy imposed by the FO-FO-FS failure tolerance creates an additional degree of difficulty in the management of the bus and its subsystems. It becomes a major task to maintain the operational integrity of the system; it may be necessary to dedicate this function to to a system management computer. By the nature of its task, this computer is intimately related to all subsystems on the Shuttle, binding them operationally and preventing the establishment of strict independence.

If control of the bus is divided between several different computers, then the following problems are created:

a) the management of the bus system's configuration by more than one computer. Even if management is not a shared function, the monitored status of the system, and the results of all reconfiguration decisions must be communicated from one computer to the next via DMA, via the bus control unit, or via the bus itself.

b) The resolution of conflict over the use of shared subsystems by more than one computer, especially subsystems that require several bus transactions to effect a completed sequence of operations (e.g., the reading of range and range-rate from a navigation radar. See Appendix C.).

These two reasons dictate a single control authority for the data bus. It is a recommendation of this study that at all times only one computer be provided with the ability to directly access, control or otherwise influence the activity on the data bus. Another computer may be introduced into the system but only to provide added processing capabilities. It must be interfaced to the bus through the standard bus interface as just another subsystem to be serviced by the bus, and must operate within the constraints of the bus control policy. The recommended approach to computer bus control is as illustrated in Figure 3.12.

7.2.2 Bus Control Policy

In this section techniques for bus communication control, and error detection and correction are evaluated and recommendations are made.

7.2.2.1 Bus Access Method

Of the four approaches discussed in Chapter 4, the command/response addressing access method is considered to be more directly applicable to the Shuttle avionics data bus, principally because of its simplicity, its deterministic behavior, and its flexible addressing structure. Although the polling and contention methods result in a more efficient bus utilization, and provide more general and sophisticated solutions to the communication problem, the increased complexity required of the bus elements does not appear justifiable. Furthermore, the random access operation permitted by these techniques can result in an unpredictable I/O service and response rate, and an increased difficulty of testing and validation.

Command/response addressing is the simplest approach which can do the job. There is, however, a reservation to be made concerning its inability to service random events without incurring a high sampling frequency. An instance of this kind of bus activity is the monitoring of a large number of terminals for their operational status. A sampling of every terminal in a 250 terminal data bus system can be completed in less than 2 milliseconds, assuming a 1 megabit per second data bus rate. This represents about 10% of a typical 20 millisecond bus minor cycle. (In practice it would not be necessary to sample every terminal in every cycle. Random event indicators would probably be grouped into a few interfaces. These factors would reduce the actual duty cycle considerably.)

Although this example does not pose a major problem, if this kind of activity becomes a significant proportion of all bus communications the margin of advantage that command/response has over the polling access technique will narrow. A strict command/response structure can be modified by incorporating a form of group addressing for terminals servicing subsystems with random outputs. The problem lies in coordinating a number of simultaneously echoed responses from such a group. The Shuttle requirements as known to date do not justify the introduction of further complexity into the bus design to solve this problem.

A reservation of far less significance is that command/response allows transactions to occur only between computer/BCU and any terminal. It precludes the terminal-to-terminal communication required by telemetry or on-board flight recording equipment. However, the needs of these systems could be satisfied by a non-standard bus interface unit that:

a) had no transmit capability,

b) was able to receive transmissions from many other terminals.

With some processing capability, the telemetry or recording subsystem could extract the desired information from the stream of bus traffic.

7.2.2.2 Error Control Technique

Each of the error control codes discussed in Chapter 5 was devised to offer the best solution to the requirements of a particular communication system and its assumed channel error characteristics. The main difference between the codes was the type of error environment (e.g., random, burst, etc.). Since

accurate information about the channel error characteristics of the Shuttle data bus is not available, none of the basic channel models assumed for analysis (e.g., binary symmetric, binary erasure, burst) can be chosen with certainty. The anticipated noise environment for the Shuttle is such that it is advisable to assume a complex burst noise characteristic, which may not be amenable to analytical definition. The findings of Chapter 5 are summarized here.

a) Coding techniques which are designed for the control of random, independent errors are not satisfactory for the Shuttle bus. These include simple parity, Hamming codes and BCH codes. Fire codes and other burst codes capable of protection against burst errors are not satisfactory because of the required complexity at BCU and terminal, and because the characteristics of burst noise in terms of duration, intensity, and spectrum are not known in sufficient detail to judge their suitability and effectiveness.

b) A forward error correction method is considered questionable for the Shuttle data bus error control scheme, primarily due to the complexity required to correct two or more errors. An error detection scheme with recovery by retransmission is recommended as the basic control approach. Detection and retransmission have been shown to be superior to forward error correction when independent error rates are low and burst errors are expected.

c) The two dimensional parity check is considered a reasonable approach for the detection of Shuttle transmission errors. The coding of parity is simple and requires no predetermined knowledge of message length. It offers a high certainty of detection of random errors (the probability of an undetected error for an error probability of 10^{-6} is in the order of 10^{-30} for a 4-byte transmission). Its major drawback is the fairly low efficiency for short transmissions that results from the vertical parity byte: for single byte transfers it could fall below 50%.

Correction by retransmission requires error detection and the capability to request re-transmission at both ends of the communication link. In a command/response environment, the terminal may not initiate such a request. The need to retransmit a message incorrectly received from the BCU must be built into the bus control structure. It is recommended that there be automatic feedback from a terminal upon receipt of any BCU command of an identifying message (e.g., its address), which would serve to indicate that:

a) there was no hardware failure of the path to the terminal;

b) the correct terminal received the message;

c) the terminal correctly verified the horizontal and vertical parity within the message.

The BCU has, of course, the ability to request retransmission by a terminal of data which the BCU found to fail the check for parity. A "time-out" check in the BCU may suffice to determine the non-receipt of an echo.

This "address echo" technique can be implemented in more than one way:

a) the transmitted message is held at the terminal until echo verification is "acknowledged" via a signal from the BCU, or

b) the transmitted message is routed directly to its destination once parity has been verified by the SIU without awaiting an acknowledgement from the BCU.

A conceivable error that illustrates the difference between these methods is one in which a noise burst changes the address in the message so that it is received by the wrong terminal. For this to occur the terminal address must be changed by noise without affecting the two-dimensional parity bits. This requires at least four bits clustered at intersecting rows and columns in the data block to be in error. The probability of this is approximately $1/m^{th}$ the probability of one error in an m-byte message, i.e. less than 10^{-30}.

Method (b) above would complete the transaction before the error had been discovered by a failure of the "echoed" address to check back at the BCU. In method (a) the BCU would not send a verification to the SIU, and the transaction could be killed. However, method (a) suffers from some drawbacks:

a) it inserts another delay into the transaction;

b) the increased security may not be necessary for all messages;

c) the traffic overhead for a bus transaction is increased;

d) there is no confirmation by the BCU that its "acknowledge" signal was correctly received, or that the transaction was completed.

The probability of the kind of failure described above is judged to be too low to justify the added complexity and inefficiency of method (a). A single feedback of the terminal address is recommended to validate transmission.

For a very small class of messages a further increase in the security of communication may be required. These are the so-called "critical" commands, of which "main engine on(off)" is an example. If this class of command constitutes no more than 5% to 10% of all output commands from the computer (which in turn are expected to be only 30% of all bus communications), then only 2% or so of all bus commands are "critical". Such a low probability is considered not to justify an increased level of error control for general bus communication. This separate class of command should be handled by redundant, multiple transmissions, which are initiated by the software concerned with the critical activity and interpreted and acted upon by the specific subsystem hardware.

A final observation on the proposed error control scheme is that verification of the correct receipt of data (or command) by the LRU cannot be achieved by the data bus system itself. Although a command/response system has, by the return of data from the LRU, an implicit verification of input requests by the central computer, validation of the receipt of computer data outputs or commands cannot be verified positively without monitoring the status or mode of the LRU subsequent to transmission.

7.2.3 Bus Data Structure

This section reviews the size of the basic unit of data, the basic command format, and the data organization.

7.2.3.1 Basic Data Size

The selection of a word, byte, or bit data organization is influenced by several considerations:

a) the standardization of established components (e.g., shift registers, counters, multiplexers, etc.);

b) the word/byte organization of memory in the computer;

c) the bus message format and data requirements;

d) the requirement for minimum overhead.

The choice of an 8 bit data byte or a 16 bit word organization is attractive because of its standardization and availability in the industry. Several other organizations (e.g., 10 bit byte, 20 bit word) would provide more efficient packing but require non-standard sized circuit functions. Most available flight computers have memory organizations which are word-oriented. They are typically multiples of 8 bit bytes. The selection of a data organization not compatible with the computer memory will result in inefficient packing and processing by computer and BCU.

Another impact on the basic data size comes from the different scalings and formats of digital data from the various sensors in the avionics system, especially if there is an emphasis on the use of existing equipment. A byte rather than word organization minimizes packing inefficiency.

7.2.3.2 Command Format

In Chapter 4 a command format of three 9-bit bytes was assumed in the discussion of bus transactions. It is tempting to consider the possibility of a 2-byte format for the increase in utilization efficiency it offers. Two factors preclude the use of a 9-bit byte for a 2 byte command:

a) the assumption that the Shuttle will require more than 128 addressable terminals, requiring at least an 8-bit address;

b) the necessity for the indication of terminal status as a part of the echoed address established in Chapter 4, requiring at least 1 bit.

It is interesting to compare the transmission efficiencies of a 3-byte (9-bit byte) versus a 2-byte (10 or 11 bit byte) command format, if it is assumed that in all cases a standard 8-bit data byte is retained. The efficiency of a read transaction (see section 4.4) for block lengths of 1, 8, and 32 bytes is as follows:

Byte size \ Number of bytes	1	8	32
9-bit (3-byte command)	11%	47%	73%
10-bit (2-byte command)	13%	49%	69%
11-bit (2-byte command)	12%	45%	63%

Read Transaction Efficiency

It can be seen that for short transactions, involving less than 8 data bytes, a 10 or 11 bit 2-byte command is more efficient, even though the byte carries 1 or 2 redundant bits in addition to the data.

In summary the recommendation is that the basic data size be keyed to the word size of the controlling computer. For example a 32-bit or 24-bit control computer would dictate an 8-bit data byte, whatever the bus command structure. This minimizes main storage inefficiencies and reformatting logic in the BCU. The command byte structure is not as critical. It depends on the scope of the avionics system and on the nature of the majority of bus transactions. The marginally superior efficiency of the 10 or 11-bit byte would not alone justify the use of a different size command byte. However, the saving in register lengths at the terminals, coupled with the possibility of improved security by using the spare bits for parity, makes the 2-byte command structure with a 10 or 11-bit byte a strong candidate.

7.2.3.3 Block Size

The question is whether to provide for transmission of a fixed or variable number of data bytes, and if variable, should the number be continuously expandable from 1 to n bytes, or in steps. A variable data capability incurs a higher command word overhead to specify the block size. The choice between a fixed or variable length message format depends on the acceptable efficiency of transfer, and on the expected nature of the average bus transaction. The following qualitative observations can be made about Shuttle data bus communications:

a) control commands from the computer to a subsystem are probably single byte transfers. However, only 25% - 30% of all transactions are expected to be like this. The majority of transactions will be data acquisition by the computer.

b) It is the nature of a control computer to handle data in groups of functionally related types (e.g. inertial angles, state vectors, status measurements, etc.).

c) Certain subsystems require large amounts of contiguous data to be transferred (e.g., the display system, telemetry).

d) At the present stage of Shuttle development the specification of an optimum fixed data size cannot be made with any certainty.

These observations support the premise that a variable data length is required. The simpler software and hardware of a fixed

data format must be traded against the overhead of:

a) transmitting redundant bits for shorter messages;

b) structuring a number of transactions to transfer messages longer than the fixed length.

There is obviously a need to limit the maximum length of the variable message. It is desirable to have a limit on the time occupancy of a bus communication to ensure that a sufficient number of messages can be sent by the computer each minor cycle, and that the response requirements dictated by the higher frequency sampling rates can be satisfied. A variable data format with a maximum length is recommended for bus data transmission. A 32 byte maximum with fixed lengths of 4, 8, 32 bytes seems to be a reasonable length. It requires buffering of up to 32 bytes at each terminal, and the order code must contain two more bits to specify the fixed lengths.

7.2.4 Functions of the Terminal

This section reviews some of the functions that may be performed by the bus terminal, and makes recommendation for and against the inclusion of various data handling capabilities.

7.2.4.1 Standardization of Terminal

The basic task is to provide for the transfer of data and commands from the bus to the destined avionics equipment. This involves extraction of the data from the bus signal, decoding the function, selecting the appropriate equipment interface, and converting and conditioning the digital data to the form (analog, serial digital, etc.) specified for that interface. Specification of the unit is made difficult by the conflict between standardization and flexibility. At this early stage of Shuttle avionics definition it is impossible exhaustively to specify the number and range of equipment interfaces. It is not possible, therefore, to define the optimized set of common interface requirements to which a standard subsystem interface unit should be designed. Three major options face the data bus designer:

a) maintain standardization by equipping every terminal with the capability of servicing the gamut of possible electrical signal types;

b) maintain standardization at the expense of generality by providing a restricted set of standard interfaces;

c) provide generality at the expense of standardization by custom designing the electronic interface to meet individual equipment requirements.

The first option provides complete freedom to the system designer, but at prohibitive cost in complexity, power, and size. The last allows the same freedom, but the cost is reflected in the design, development, production and eventual maintenance of a diverse collection of different terminal units. The second approach compromises flexibility and complexity in order to maintain standardization, and is the one recommended. Interface requirements that fall outside the selected standard set require the design of special equipment, to be associated with the subsystem. For the range of signals to be specified in the next section, this situation is expected to arise only rarely.

7.2.4.2 Type and Number of Interfaces

The next question becomes: what shall the set of standard interfaces be? Both the diversity and the number of signals is important. The product of these quantities will be defined as the number of channels for which the address in the bus command word must be sized. A nine bit address allows up to 512 channels to be specified. It is considered that this is more than adequate for any single item of equipment, and is in line with the terminal address field requirement discussed in Section 7.2.2.

A fixed selection of signal types must include analog, parallel digital, serial digital and possibly discrete on/off. The discrete signal is extremely convenient in the control of electronic equipment, but is awkward to handle in a byte-oriented data organization. It is recommended that these signals are grouped and read as parallel digital inputs rather than individually. The setting and resetting of latches or flip-flops required to read discretes presents a degree of difficulty to the bus command structure, and it is considered best to let the subsystem cope with it. The 512 channels must include the ability to specify the length of the variable block, and to indicate whether the data command is for input or output. The 9-bit channel address can be assigned the following fields to realize a fixed standard electronic interface capability.

Number of Bits	Function
1	Input or output indication
2	Block size 01 4 bytes 10 8 bytes 11 32 bytes
2	Signal type 01 analog 10 parallel digital 11 serial digital
4	Signal address

Four signal address bits allow up to 16 signals of each type to be addressed. This interface specification represents a maximum capability. Fewer signals can be implemented in a particular terminal, as discussed in Section 7.3, or a channel bit can be saved if a maximum of 8 signals is adequate.

7.2.4.3 Other Functions

The basic function of the terminal is to transfer data and commands to the equipment. Additional capabilities have been proposed for the terminal, such as:

a) checking input signals for out-of-nominal limits;

b) local timing and counting;

c) local data compression;

d) local sequencing of subsystem functions such as built-in test equipment.

Even though these functions are considered to be valuable, for off-loading the central computer's task, and for relieving the bus traffic, it is nevertheless recommended that the basic specifications of the data bus and the terminal be devoted to the communication function.

The detailed design of the bus control structure, the data format and the organization of the terminal need not exclude the eventual incorporation of further capabilities. Techniques for expansion without reworking the whole structure are discussed in Section 7.3.6.

7.3 Organization of the Data Bus Terminal

7.3.1 Introduction

The discussions of Chapters 3, 4, 5, and 6 and the preceding sections of this chapter have been concerned with how the Shuttle avionics subsystems, the computers, and the data bus should be configured and controlled. An overall set of requirements for the nature of the interface between a subsystem and the data bus have been generated. It is the objective of this section to conduct a more detailed examination of the characteristics of this interface and to make specific recommendations for the interface unit.

7.3.2 Functions of the Bus Terminal

The functions of this interface are determined by:

a) The basic bus design and bus control philosophy;

b) The generalized subsystem (or LRU) interface requirement;

c) The degree of decentralization of the avionics functions (e.g., how much failure detection, local processing, etc. is done locally rather than by the central computer).

The first and second categories determine the nature of the interface with the bus and the LRU respectively. The third category cannot be defined with much certainty at this stage of Shuttle development. The addition of capabilities beyond those required for communication over the bus is described in Section 7.3.6.

7.3.2.1 Functions of the Bus Interface

Physically this interface consists of a single data path for each bus line. Electrically, it may be a coaxial or a balanced shielded line terminated in its characteristic impedance. Functionally, it performs the following basic tasks. First, those associated with the receive mode:

a) Coupling of the interface unit to the bus line.

b) Detection, demodulation of the incoming signal, and conditioning to standard logic levels.

c) Identification of the synchronization signal which indicates that a message is about to begin.

d) Derivation of clock signal if inherent in the data, or synchroni: tion of a separately derived or received clock with the incoming data.

e) Decoding of address by "anding" with a wired-in pattern to determine if terminal is the intended recipient of the message.

f) Determination of the active bus line. This sets an internal flag to indicate on which line to respond.

The operations that follow this point in the receive sequence involve the manipulation of the data rather than its reception. They are:

g) Acceptance of the message, byte-by-byte.

h) Decoding of the message into function codes, channel addresses and data.

i) Determination of correct parity in the received data.

From this point the receiver functions of the interface unit are determined more by the subsystem and operational requirements. For the recommended Command/Response addressing technique, the basic bus transmitter functions are:

j) Coupling to the lines.

k) Modulation of the active line.

l) Transmission of beginning-of-message sync pulse, if required.

m) Transmission of wired-in address, if correct receipt of the incoming message has been established.

n) Transmission of the terminal status bit(s) (e.g. EIU/ LRU busy).

o) Transmission of all data bytes, as determined by the function code.

p) Generation of horizontal parity on every byte, and vertical parity on data blocks.

q) Transmission of an end-of-message sync, if required.

The above receive and transmit functions are provided by the basic control elements depicted in Figure 7.1, which purposely does not show all logical interconnections.

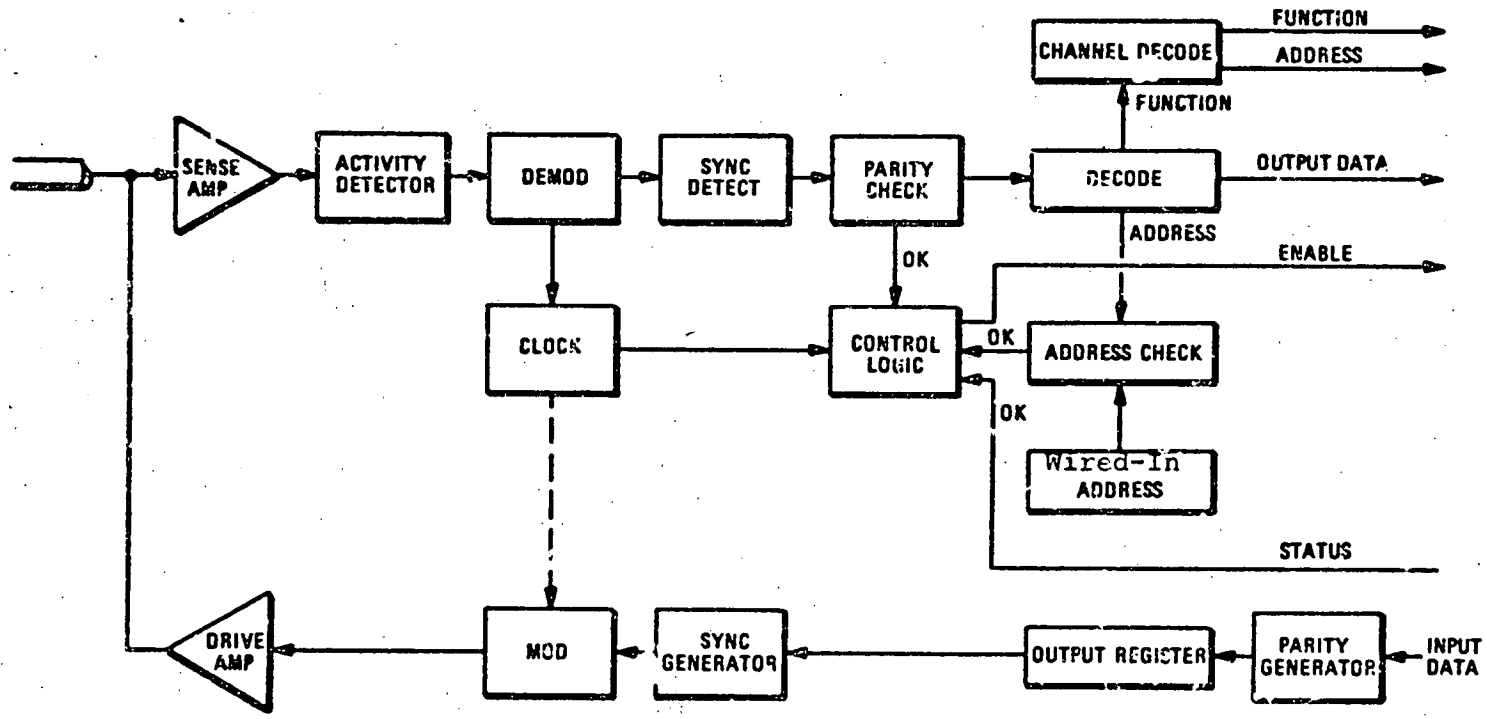


Figure 7.1 Standard bus interface functions

7.3.2.2 Functions of the Subsystem Interface (EIU)

In Chapter 4 and Section 7.2, the requirements of the electronic subsystem interface were analyzed and estimated. Taking the recommendations of Section 7.2, the electronic interface consists of the following signals:

a) 16 analog input

b) 16 analog output

c) 16 parallel digital input

d) 16 parallel digital output

e) 16 serial digital input

f) 16 serial parallel output

g) 4 digital address lines

h) 1 LRU status signal

This interface capability is illustrated in Fig. 7.2, which also points out the elements of an interface controller necessary to allow these interfaces to be operated by a bus interface unit. (Again, the diagram is not intended to define all the logic, only the basic functions.) In the simple command/response control philosophy that has been recommended the interfaces are addressed serially; i.e., not more than one input or output channel of any one signal type can be active at a time. This allows a simplification of the addressing and control logic and a reduction in the number of interface connections. For example, one set of 4 digital address lines will suffice to select any of the 64 digital channels. The analog output could probably be multiplexed on one line by using the address lines to indicate its destination. The individual signal interface types can be made modularly expandable to the maximum of 16 allowed by a fixed distribution of 512 channels, without changing the basic control structure, by plug-in cards or by customized LSI chip assembly. This allows the LRU interface unit to be matched to the requirements of a specific LRU electronic design without incurring an inordinate component redundancy, or power dissipation from unused circuitry within the EIU. The software that communicates with the subsystem must, however, be made aware that the EIU interface has been thus restricted. An "invalid channel address" signal would probably be justified to warn of the erroneous addressing of functions that had not been implemented in a particular EIU.

An important function of the EIU portion of the interface unit is to provide sufficient buffering to hold the maximum

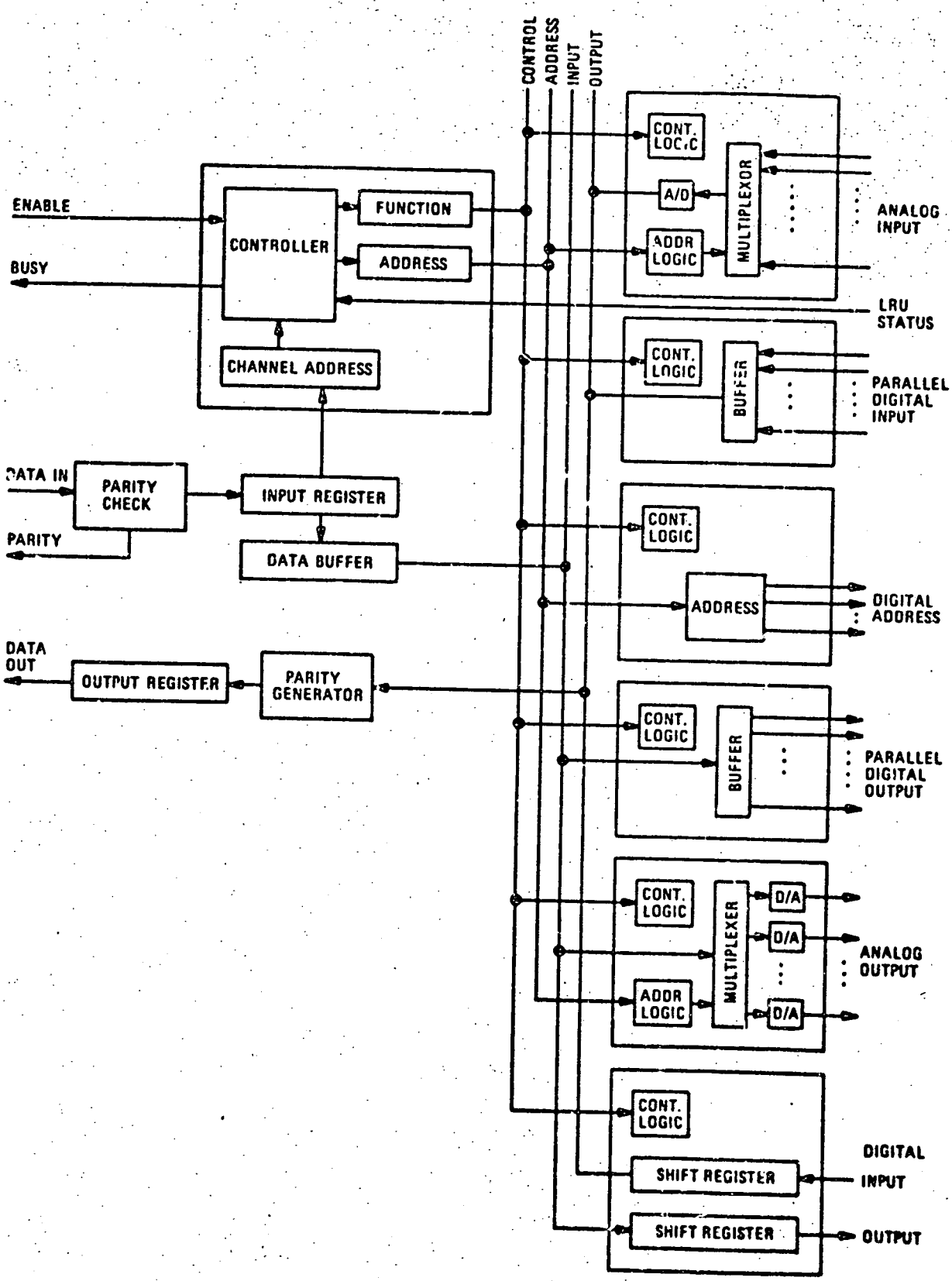


Figure 7.2 Electronic interface functions

block size of 256 bits suggested in Section 7.2 until the vertical parity bit has been received at the terminal. During the subsequent transmission of the block to the LRU the EIU "busy" signal is set. (See Section 4.4 for details.)

7.3.3 Interfacing the LRU to the Bus

The problem of interfacing a quad-redundant bus with a subsystem of arbitrary redundancy was discussed in Chapter 3. This area will now be re-examined in the light of the detailed analyses of the bus system function offered in Chapters 4, 5, and 6, and the previous sections of this chapter. The problems are conveniently illustrated by the case of a triply-redundant subsystem. Ignoring the fairly unattractive case of cross-strapping the 50-100 wires at each LRU interface, there are two basic approaches.

7.3.3.1 Cross-strapping between SIU and EIU (Figure 7.3)

This is the case of the physically separate and remote EIU discussed as Configuration 7 in Section 3.3.2. Each SIU must provide a path to several EIU's, to a maximum determined by system considerations. (Some current designs propose an EIU fan-out capability of eight.)

Since the EIU in this configuration is a separate physical unit, geographically located with the LRU rather than the bus, its connection with the SIU constitutes at least the five signals indicated, namely:

a) Input data

b) Output data

c) Enable

d) Parity check ok

e) Status (busy)

In the example considered here, at least 15 wires per SIU, or a total of 60 wires for the complete SIU/EIU interface are necessary. However, since the utilization of the EIU interfaces, and therefore the EIU's themselves, can be sequential, only one SIU to EIU path is active at any one instant. The 60 wires could be arranged into a 15-wire bus, or they could all be multiplexed into one 5-wire channel to minimize physical

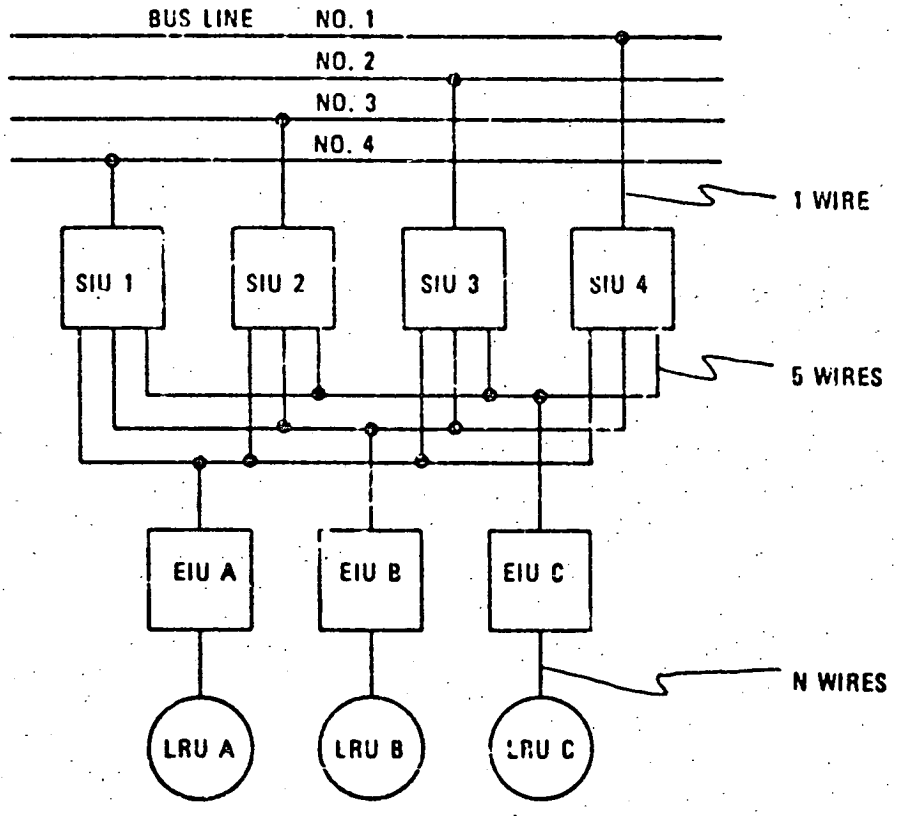


Figure 7.3 SIU to EIU cross connection complexity

connections. However, this would incur a considerable cost in driving and sensing, encoding and decoding, multiplexing and demultiplexing, circuitry. Such an arrangement would constitute a local bus, requiring its own control, addressing, and error detection policies.

7.3.3.2 Cross-strapping Between Bus and SIU

If the EIU were not a separate unit, and if each EIU were associated with only one SIU, the local communication problem would be eliminated. There would be fewer shift registers and less parity logic since only the bus communication channel need be verified in a combined SIU/EIU.

This approach was described as Configuration 6 in Section 3.3.2. The major disadvantages were identified as:

a) The SIU constitutes a single point failure for the bus, since it connects to all lines.

b) Four bus connections are required for each redundant LRU (totalling 12 in the chosen example).

The first of these objections could be answered by providing a separate SIU for each bus line, and combining the outputs of 4 SIU's into a common summing point at the input of an EIU. (See Figure 7.4). The justification for this approach would be:

a) Since the SIU is dedicated to one EIU it requires less addressing capability, and can therefore be a much simpler device.

b) The SIU to EIU connection could consist of less than the 5 wires previously defined.

It is unlikely, however, that the SIU/EIU interfaces can be less than 2 or 3 wires (input, output, and control). When summed into a single connection, this still results in a total of 5 to 10 wires for the three SIU/EIU interfaces in this example. The approach requires a total of 12 SIU's rather than 4, and the total number of connections to the bus is still 12. In summary, the separation of SIU and EIU does not seem to offer a solution to the cross-connection problem.

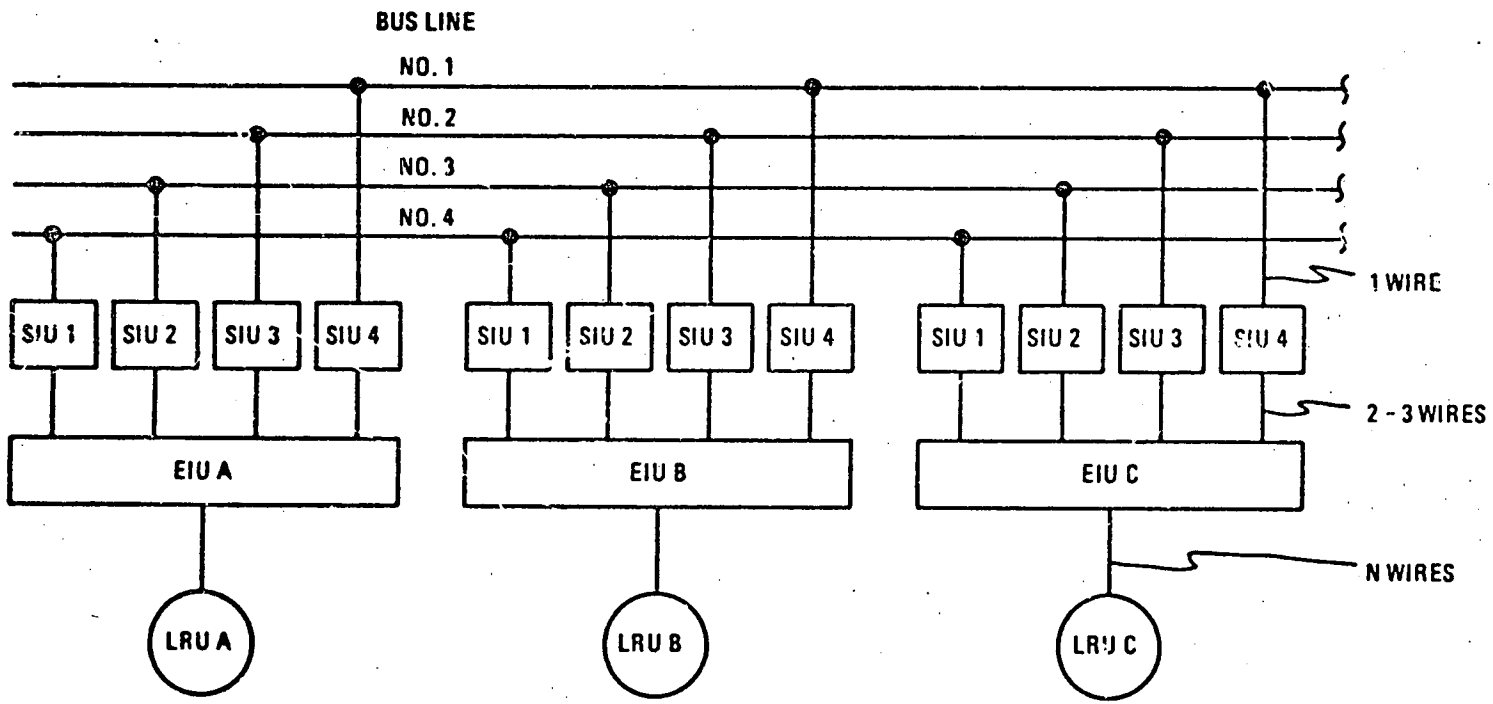


Figure 7.4 Cross connection at bus via separate SIU

7.3.4 Recommended Bus/SIU/EIU Configuration

The SIU could be combined with the EIU if bus termination and fan-out were considered as a problem outside of the SIU. A line coupling unit could provide this function, and its use is illustrated in Figure 7.5. A line coupler (LC) is considered a part of each bus line. It is basically a receiver-transmitter, i.e., a line repeater, which in a real implementation of a data bus in the Shuttle may be required in any case for the reasons of attenuation, pulse delays, reflections, etc. discussed in Chapter 6. Each LC provides a local fan-out of the bus to a number of SIU's. From the point of view of the bus a LC is identical to a single SIU connection, both electrically and in terms of vulnerability to failure. To an SIU each fan-out has the characteristics of a bus connection.

The resulting combined SIU/EIU unit made possible by this concept results in the simplest approach to matching redundant subsystems to the bus. It scores because it minimizes the following:

a) The number of physical interconnections.

b) The complexity of the SIU and EIU.

c) The total number of SIUs and EIUs.

The disadvantages of the combined SIU/EIU configuration that were identified in Section 3.3.2 may be re-evaluated for the line coupler approach as follows:

a) Only one EIU may be interfaced via an SIU. Since the combined SIU/EIU is not much more complex than the EIU itself, additional EIU capabilities can be provided by interfacing another SIU/EIU via the line couplers to the bus.

b) The combined SIU/EIU is required to interface every LRU function. The question here is, can an LRU be interfaced directly to the SIU. If not, an EIU of some minimal complexity is required. (The modularly expandable EIU design approach of Section 7.3.2 can assure minimal complexity.)

c) Only one bus line at a time can be used for communication on the data bus. Although this objection can be levelled against other bus interfacing configurations that use a simple addressing technique, it seems an obvious limitation for this SIU design. It is discussed in the next sections.

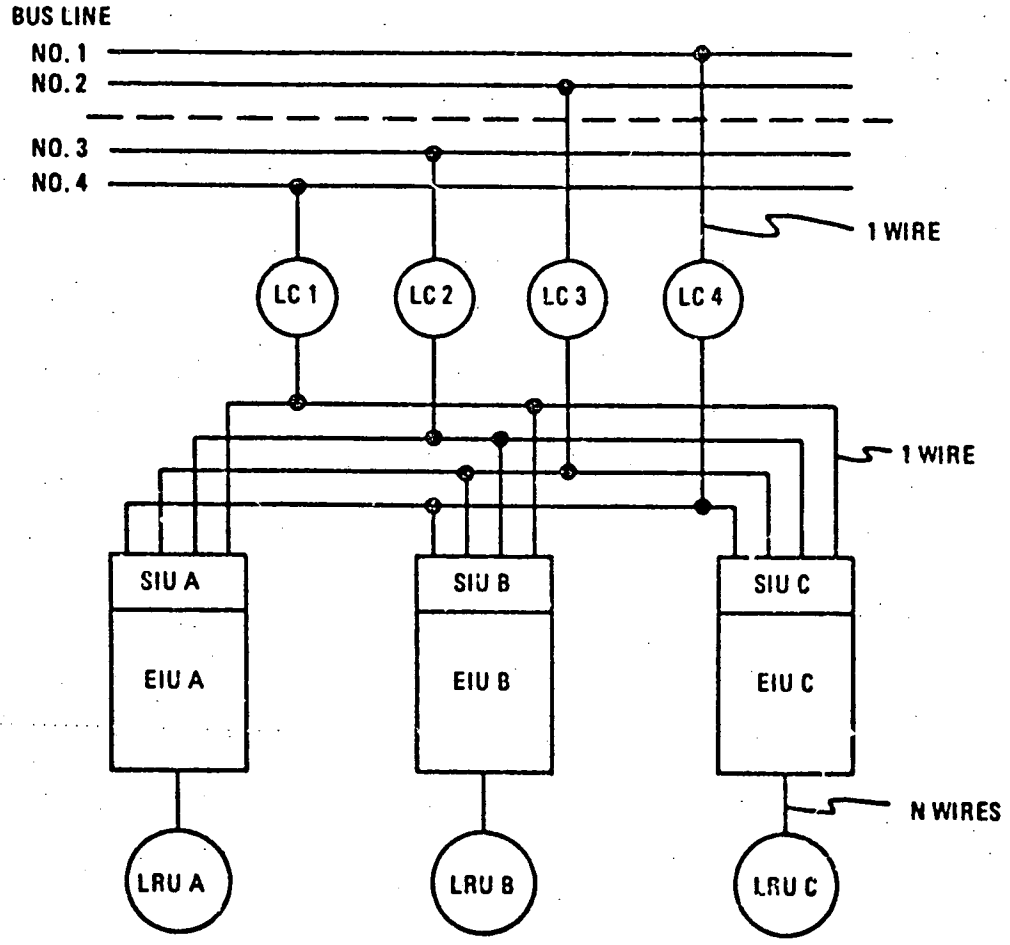


Figure 7.5 Cross connection at bus via line couplers

d) Each SIU, being interfaced to all four bus lines, constitutes a single point failure for the whole bus system for signal failures such as:

1) Continuous low level

2) Continuous high level

3) Uncontrolled transmission of a meaningless signal

4) Intermittent or noisy signal levels.

These would be faithfully communicated on all the bus lines by the purposely passive line coupling units. This subject is also discussed in the next section.

7.3.5 Recommended Bus/SIU Interface Design

Although possessing the outstanding advantages of minimal interconnection complexity, the chosen SIU/EIU configuration appears to suffer from a vulnerability to common noise and failure modes:

a) If all bus lines are simply 'OR'-ed into each SIU, the presence of noise or erroneous signals on one bus line may prevent the signal-carrying line from being properly read by the terminal.

b) A noisy SIU could transmit garbage indiscriminately on all lines, destroying the operation of the bus.

c) A short circuit or continuous high level common to all the terminals of one SIU could prevent the line couplers from responding to any input or output signals, effectively shutting off the whole station, and perhaps destroying bus operation.

These deficiencies can be remedied by careful specification of the line coupling unit and the SIU input channels.

7.3.5.1 SIU Input Channel Specification

It is apparent that a simple input OR-ing of the bus lines is not an adequate approach. Each bus termination at the SIU must be able independently to perform the following functions:

a) Coupling and line termination

b) Modulation and demodulation

c) Detection of sync signal

d) Address verification

e) Bus line usage indication (and latch)

For each SIU these functions must be repeated four times, once per bus line. Further security can be achieved by separating the circuitry, electrically and physically, by individual power buses and individual packaging. The other functions of the SIU/EIU defined in previous sections are not duplicated. Each SIU bus interface element feeds into, and is fed by, a common point in the unit, beyond which only simple redundancy is employed. Figure 7.6 illustrates these details. The reasons for the input duplication are the following:

a) The separate line terminations prevent common connector and input circuit problems from affecting all bus lines.

b) Providing separate sync and address verification on each bus line prevents a noisy line from affecting the signal line. Incoming noise must pass the sync test and address comparison before it reaches the point where it merges with valid signal. The likelihood of this occurring with random noise is remote.

c) Placing the bus usage indicator after the address comparator prevents the latch from being set by noise. It also enables a future extension of the data bus capabilities to simultaneous use of more than one line by more than one SIU.

d) Transmission from the EIU onto the bus must pass the usage latch test before being passed to the line drivers. Since only one latch may be set at any time, this prevents a "berserk" EIU from "drowning" all lines with garbage.

e) To further minimize the possibility of noise from being transmitted onto any line by the SIU, the inactive bus transmitting circuits in each SIU may be powered off by the usage latch condition.

7.3.5.2 Line Coupler Specification

The line coupler's primary functions are:

a) Bus line termination;

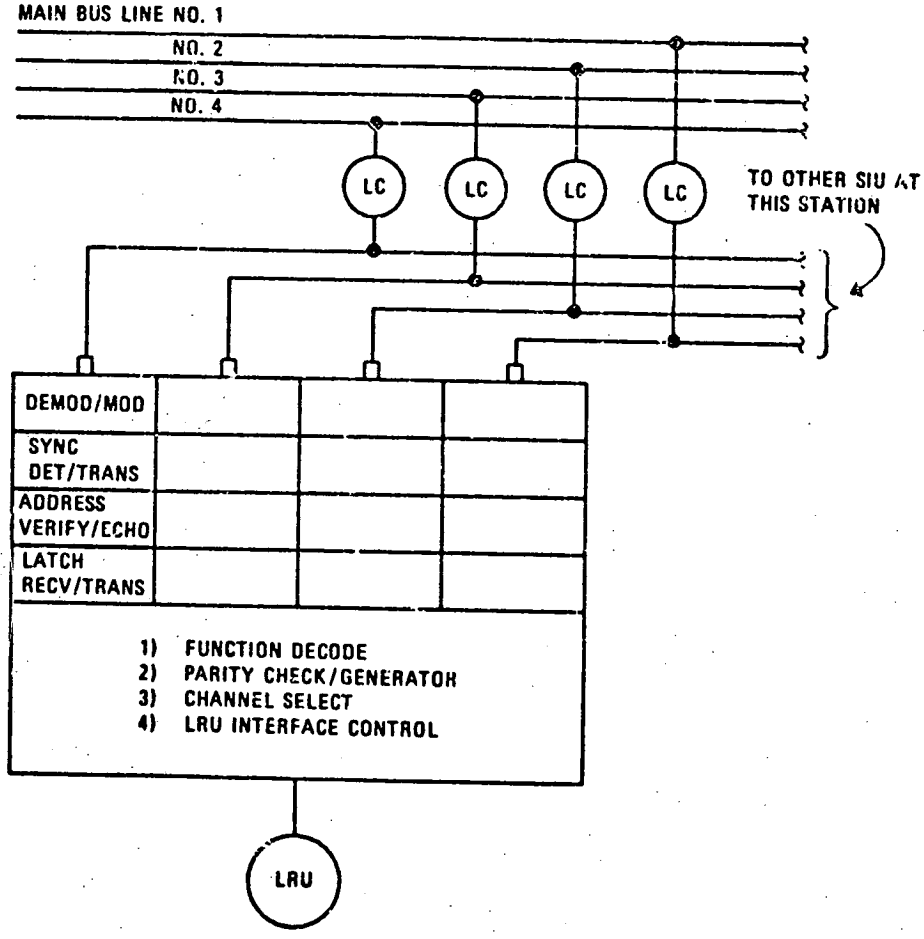


Figure 7.6 Recommended SIU/EIU terminal organization

b) Signal sensing, amplification, conditioning, and retransmission.

c) Fan-out and fan-in.

Its electrical design can aid the security of the system by providing some discrimination to noise as follows:

a) The susceptibility of the bus system to continuous high or low level signal levels, produced by short-circuits to ground or to DC supply lines, can be lessened by AC coupling techniques (see Chapter 6). This factor favors the use of modulation techniques that have no DC requirement.

b) Signal discrimination is possible in each line coupler receiver. Input circuits can be sophisticated to reject signals whose characteristics differ from the chosen modulation scheme (e.g. discrimination against pulses whose durations fall outside the nominal spread of the expected data pulse).

7.3.5.3 Summary

The combined SIU/EIU organization of Figure 7.6 is the recommended design for the Shuttle data bus standard interface unit. In conjunction with the proposed line coupler elements it demonstrates a minimal interconnection complexity, yet retains a simple addressing and control policy. The suggested organization offers considerable security to insidious hardware and signal failures. Its complexity lies between that of the single combined SIU/EIU and that of the EIU with four physically separate SIUs. The approach scores over the configuration described in 7.3.3.1, because it eliminates the necessity for a multiplexed SIU to EIU interface.

7.3.6 Expansion of SIU/EIU Capabilities

The capability of the standard terminal can be increased by the incorporation of a small control memory and associated sequencing logic. Such a terminal would possess considerable "intelligence", i.e., be capable of making and acting on local decisions, which is not allowed in a command/response control structure. However, the provision of the "terminal busy" indication as part of the echoed address (see Section 7.2) opens

the possibility of this more sophisticated type of terminal operation without violating any control ground rules. For example, even if a terminal is initiated into a lengthy sequence of operations, during which time many bus transactions are possible, the busy signal will prevent unintended transactions from interferring with the sequence. The terminal would operate in a similar manner to block transfers (see Section 7.2). Two concessions to the command/response access control must be made:

a) terminal sequences may only be initiated by the computer;

b) no terminal may initiate bus activity unless commanded to by the computer.

A suggested mechanism for this expansion is to treat part of the 9-bit channel address field as the address of locations in a read-only control memory which stores the special sequences. If the sequences are themselves stored as a microprogram, then the basic terminal can remain standardized, provided that a standard (but diminished) set of electronic interfaces is also a part of the design. Such an expanded terminal would be able to provide the further functions outlined in Section 7.2.

PRECEDING PAGE BLANK NOT FILMED

Appendix A

Discussion of the Effects of Cross-strapping

This section discusses the effects of cross-strapping bus elements connected in series. The purpose is to evaluate when cross-strapping is desirable or undesirable and to determine the associated gains or losses in reliability resulting from cross-strapping.

Consider the two simple configurations illustrated in Figure A-1. In either configuration A or B the system requires that one S and one L unit are operating for the system to operate. B is cross-strapped, i.e., each S unit is interconnected to each L. Let P_S be the probability of failure of the S unit, and P_L that of the L unit. Then let

$$P_S = 1 - e^{-\lambda_S t}$$

$$P_L = 1 - e^{-\lambda_L t}$$

where λ_S and λ_L are the failure rates for units S and L respectively. If the bus to which the S units are connected is assumed infinitely reliable, its effect can be ignored in the following calculations.

The probability of failure of configuration A is given by:

$$P(A) = P(S_1 \text{ or } L_1)\ P(S_2 \text{ or } L_2)$$

Since one S unit and one L unit are required for operation, this results in

$$P(A) = P_S^2 + P_L^2 + 2\,P_S\,P_L - 2\,P_S^2\,P_L - 2\,P_S P_L^2 + P_S^2\,P_L^2$$

The probability of failure of configuration B, again assuming that one S and one L are required to operate, is given by:

$$P(B) = P(S_1 \text{ and } S_2) + P(L_1 \text{and } L_2) - P(S_1 \text{ and } S_2)\ P(L_1 \text{ and } L_2)$$

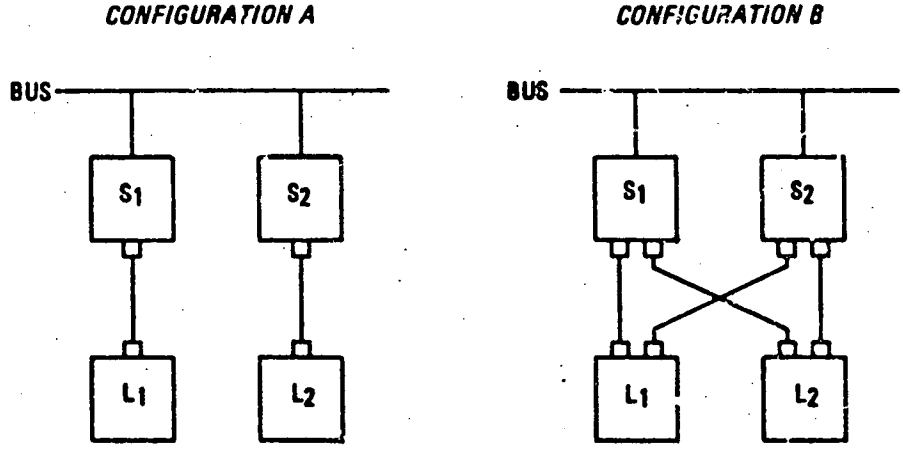


Figure A.1 Simple and cross connected configurations

or $$P(B) = P_S^2 + P_L^2 - P_S^2 P_L^2$$

However, the additional complexity of the S and L units in configuration B to provide cross-strapping will result in increased failure rates. Therefore let:

$$P_S(B) = P_S(A) + \delta P_S$$

$$P_L(B) = P_L(A) + \delta P_L$$

The gain or loss incurred by cross-strapping can be gauged by defining a figure of merit:

$$FOM = \frac{P(A)}{P(B)}$$

In order to simplify the number of variables the following definitions and assumptions are made:

a) Let $P_L = K P_S$, i.e., let unit L be K times as likely to fail in a given interval as unit S.

b) Let $\lambda_S(B) = \lambda_S(A) + \delta\lambda_S$

$$\lambda_L(B) = \lambda_L(A) + \delta\lambda_L$$

and furthermore let

$$\delta\lambda_S = \delta\lambda_L$$

from which it can be shown that

$$\delta P_S = \delta P_L$$

c) Let $\delta P_S = F P_S$ where F is the fractional increase in unreliability due to interconnection.

d) Finally, since:

$$P_S(B) = P_S(1 + F)$$

$$P_L(B) = P_S(K + F),$$

FOM can be expressed as a function of P_S, F, and K.

or $$FOM = \frac{P(A)\,[P_S, K]}{P(B)\,[P_S, F, K]}$$

FOM is plotted against K for various values of F in Figures A-2 through A-4, for failure probabilities $P_S = 10^{-1}$, 10^{-4}, and 10^{-8}.

The principal result is that the maximum gain in total reliability of configuration B over configuration A occurs when the probabilities of failure of the two units are equal, i.e., K = 1, and when the reliability of the added interconnection is good, i.e., F = 0.

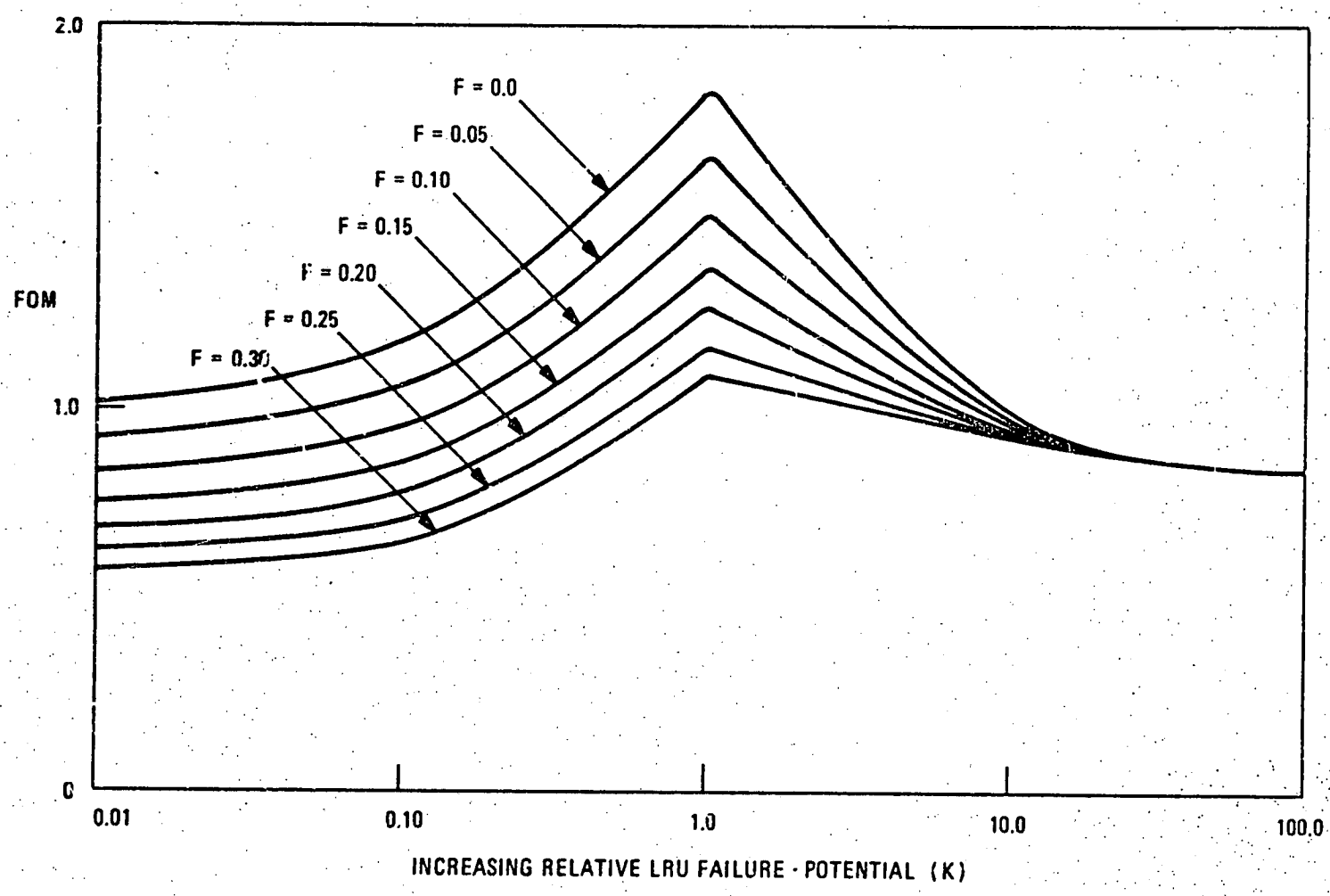


Figure A.2 Figure of merit (P(A)/P(B)) for $P_S = 10^{-1}$

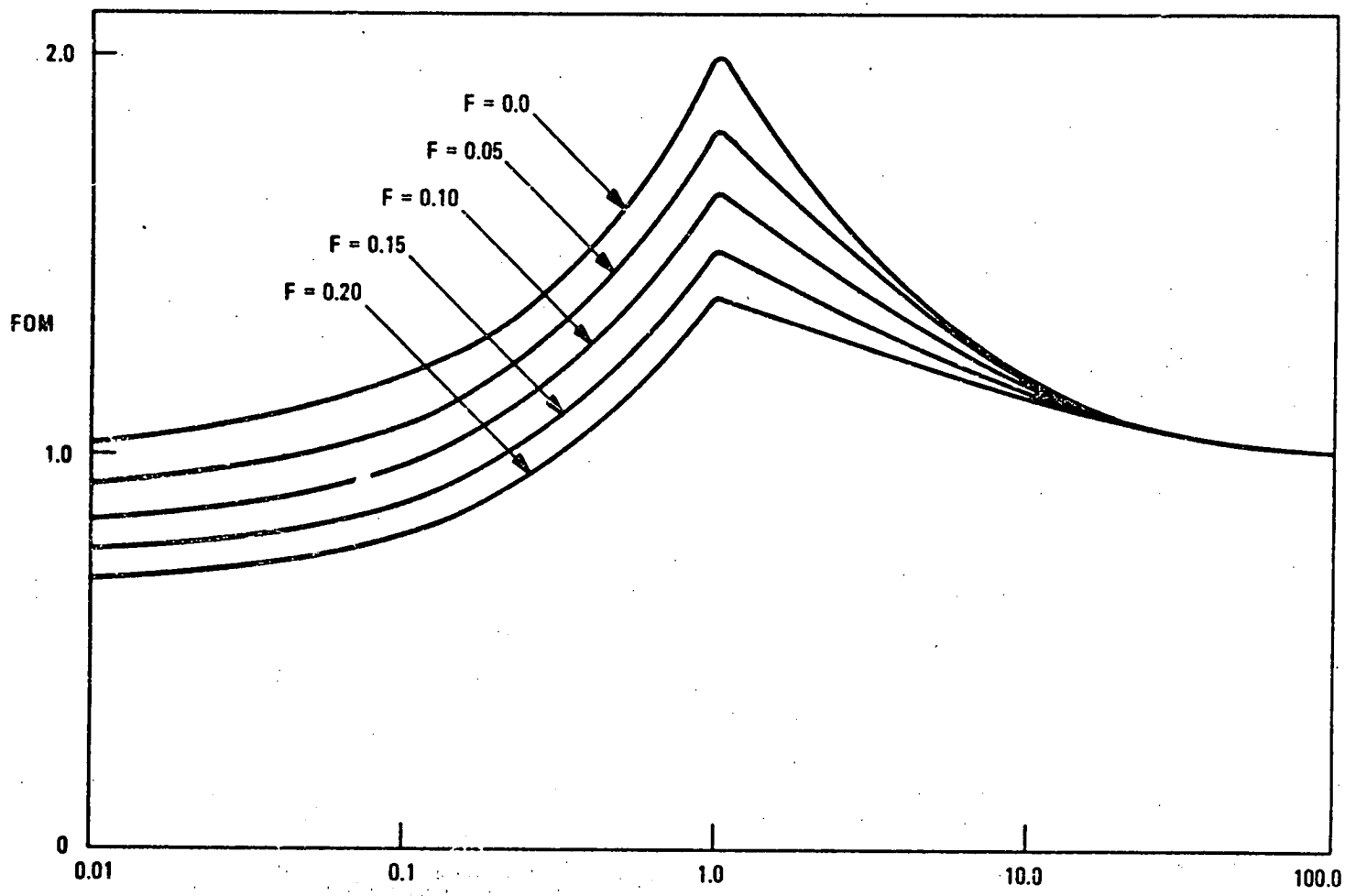


Figure A.3 Figure of merit (P(A)/P(B)) for $P_S = 10^{-4}$

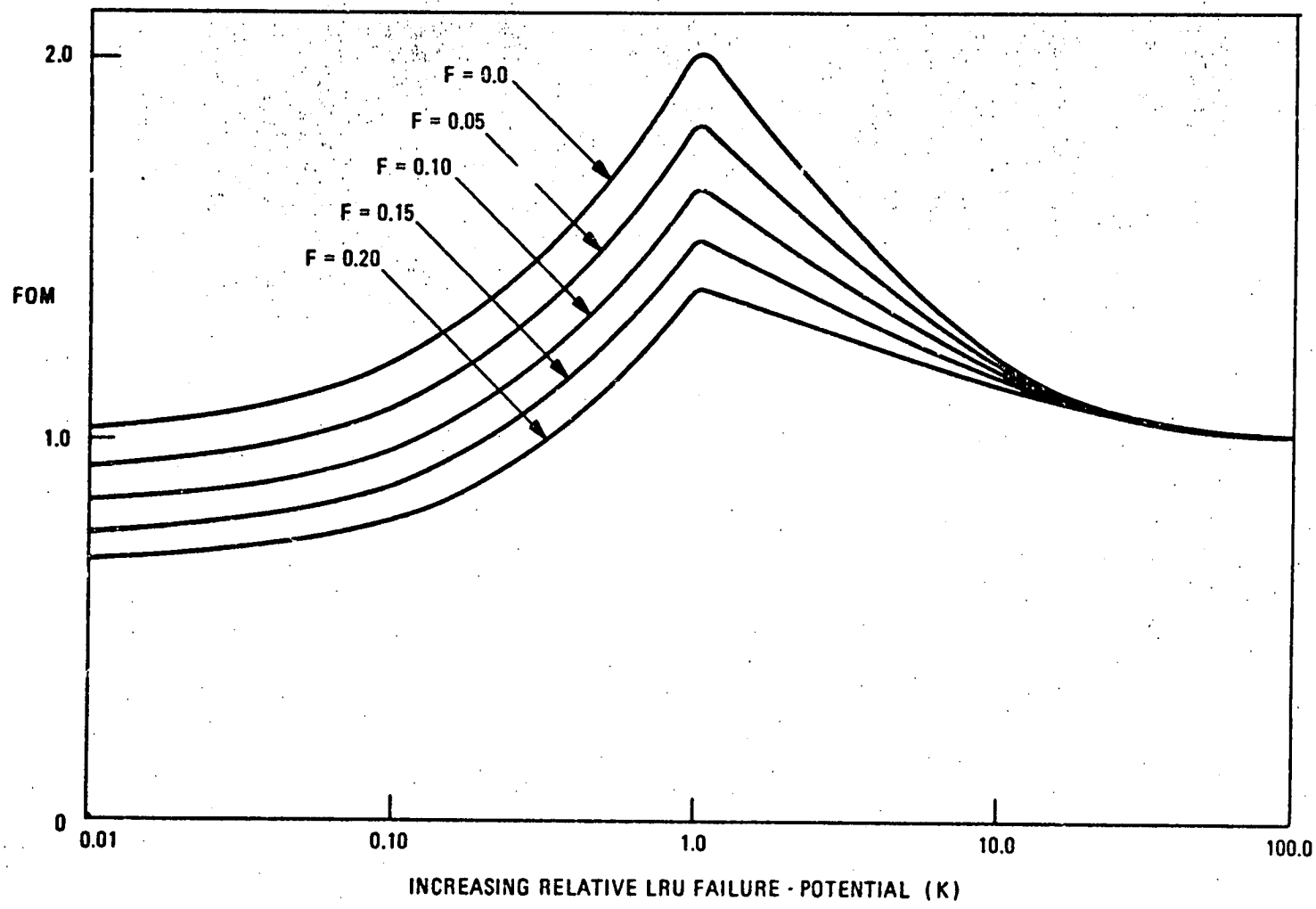


Figure A.4 Figure of merit (P(A)/(PB)) for $P_S = 10^{-8}$

PRECEDING PAGE BLANK NOT FILMED

Appendix B

Analysis of a Typical Avionics Subsystem

B.1 Introduction

No specific items of Shuttle avionics equipment have been defined to date. The design of a data bus interface unit is, however, dependent on the specific requirements of the various interfaces. In order to provide a basis for a general evaluation of design criteria, some assumptions of interface characteristics were made in the body of this report. The following provides a description of the Distance Measuring Equipment (DME) produced by the Cubic Corporation as CR-100. Although it is not intended to suggest that this particular unit should be specified for the Shuttle, it is considered to be representative of the types of problems that will be encountered in all data bus interfaces. No attempt has been made to optimize the interface to the CR-100. Instead, the interface of one version of the prototype is taken as a given constraint, as if the system were purchased "off the shelf".

The system operates as follows. A phase modulated signal is transmitted by an interrogator in the vehicle along with an identification code to designate which of a number of transponders located at known points is to reply. All transponders within range of the interrogator are listening, and if the designated transponder is within range, it retransmits the modulation from the received signal back to the interrogator on a different carrier frequency. The interrogator receives this signal, recovers the modulation, and compares it with that transmitted. The time delay between the transmitted and the received modulation, multiplied by the speed of light, is twice the distance to the transponder.

B.2 Operation

All moding and timing is controlled external to the CR-100. The required sequence of control commands is as follows. Assuming that the interrogator is already powered up, the DME must first receive the ID of the desired transponder, and then be given a mode command to begin the interrogation. A mode command consists of a 5 bit parallel input which must be held for the duration of the mode.

The mode must be maintained for at least the round trip transmission time (1 mile = 11 microsec) plus 50 millisec for the circuits in the transponder and interrogator to lock onto the signals. After this time has elapsed, the DME can be commanded to terminate the measurement. The time at which the command is received will be important, since it determines the time for which the measurement is valid.

The output data becomes available about 0.5 millisec after the command to terminate the measurement is received by the interrogator. The data consists of five 11-bit words, four of which represent the range, and the fifth is a data quality word whose bits indicate whether or not the various parts of the DME functioned properly. These five words are read out of the DME one at a time by sending a unique mode command for each word. The readout requires about 100 microsec per word.

Another version of the CR-100 has the capability of also measuring range rate, or more precisely, the change in range over a known time interval. This measurement is made by counting the cycles of doppler shift of the carrier during a known time period. It requires a slightly different timing sequence. In this case, the equipment supplying the mode commands must wait at least 100 millisec after the command to start transmission, and then issue a command to begin the range-rate measurement. It must then wait another period, typically of the order of 900 millisec, before commanding the termination of the range-rate measurement. The accuracies of the measurements are of the order of 30 cm for absolute range and 1 cm for changes in range. The output data is in the form of seven 11-bit words which are read out of the DME in the manner described earlier. The timing requirements are illustrated in Figures B.1 and B.2.

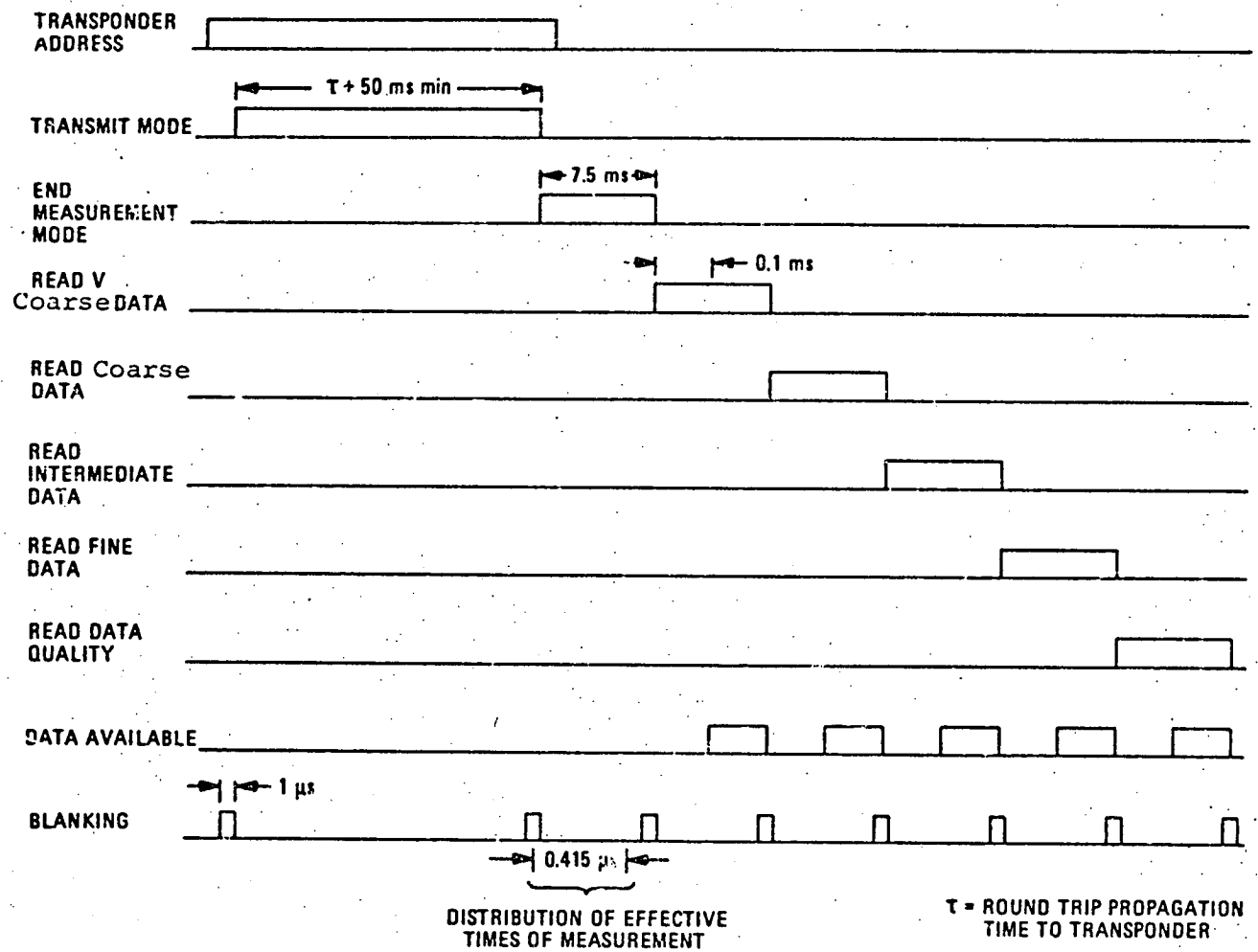
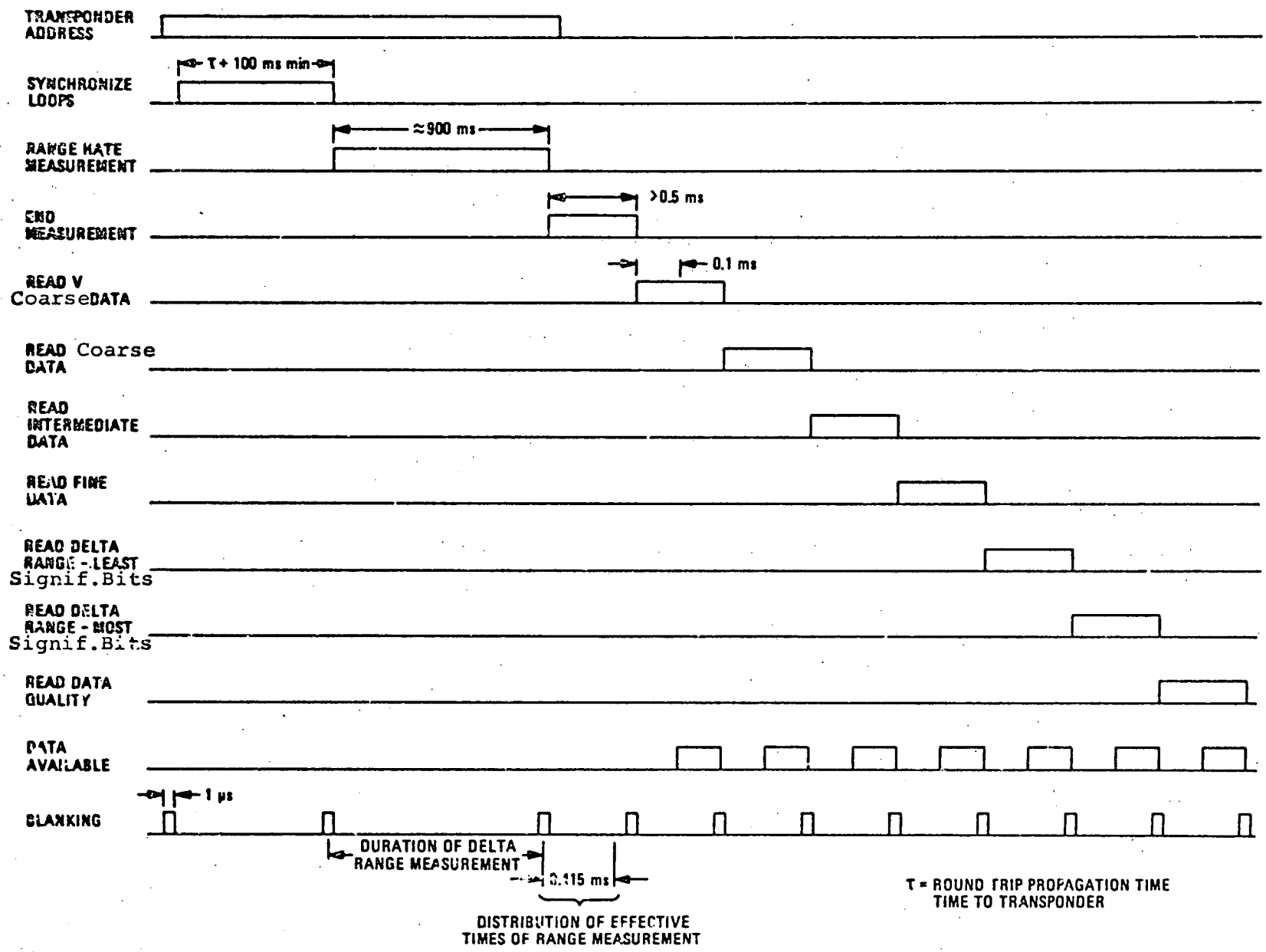


Figure B.1 Control sequence for range measurement


TRANSPONDER ADDRESS
τ + 100 ms min
SYNCHRONIZE LOOPS
≈900 ms
RANGE RATE MEASUREMENT
>0.5 ms
END MEASUREMENT
0.1 ms
READ V CoarseDATA
READ Coarse DATA
READ INTERMEDIATE DATA
READ FINE DATA
READ DELTA RANGE - LEAST Signif.Bits
READ DELTA RANGE - MOST Signif.Bits
READ DATA QUALITY
DATA AVAILABLE
1 µs
BLANKING
DURATION OF DELTA RANGE MEASUREMENT
0.115 ms
DISTRIBUTION OF EFFECTIVE TIMES OF RANGE MEASUREMENT
τ = ROUND TRIP PROPAGATION TIME TIME TO TRANSPONDER

B.3 Control Requirements

A number of functions must be provided externally in order for the DME to operate properly. For the inputs, both the mode commands and the transponder address must be applied to the DME in parallel, and held for the duration of the mode and the transmission cycle respectively. The mode command is a five bit word and the transponder address is eight bits.

The timing accuracies required for the application of the mode commands must be considered. For the range measurement, the effective time of the measurement is a rectangularly distributed random variable, having a distribution of 0.5 milliseconds, immediately following the start of the mode command to terminate the measurement. If the system application requires it, a simple change to the interrogator can reduce the width of this distribution to a few microseconds. In any case, for a vehicle travelling at orbital velocity of 22,000 ft. per second, an error in timing of 1 millisecond can introduce an error of up to 22 feet (depending on the angle between the velocity vector and the direction of the transponder).

For the delta range system, the timing between the start and end of the delta range measurement is critical. An error of 0.9 millisec in a normal 900 millisec measurement will introduce an 0.1% error into any velocity determination that is derived from the delta range data.

In order to obtain the output data from the DME, a different mode command must be used for each of the five or seven data words. The data words become available as parallel 11-bit words 100 microsec after the appropriate mode command is applied.

The source of mode control commands will also be required to generate a blanking input to the interrogator. This is a 1 microsec pulse occurring each time the mode input changes prevent the interrogator from trying to interpret the mode input while it is changing. Most of the information reliability determination and failure analysis is in the data quality word which accompanies the data. This word contains bits which indicate that a return signal was received from the transponder, that the various phase locked loops achieved lock and remained in lock during the measurement, and so on. In addition to this word, it might be desirable to arrange for about four discrete outputs from the DME to indicate that various parts of the equipment are active, and a few analog voltages to allow measurement of such quantities as transmitter input and output power and received signal strength. These measurements, if taken during the right portion of the measurement cycle, could provide useful data for checkout, failure monitoring and telemetry.

	Description	Type	Level	Number of Pins	Comments
Inputs	mode control	5 bit parallel	DTL compatible	5 + ground	severe tim constraint for some modes
	blanking	1 bit	DTL compatible	1 + ground	1 μsec pul
	transmission address	8 bit parallel	DTL compatible	8 + ground	must be he during tra mitting modes
	power on	discrete	close 28V relay coil to ground	1 + ground	
Outputs	range data data	(4) 11 bit parallel	DTL compatible	11 + ground	data word appears on output lin 100 μsec after cor-responding mode comma is receive
	delta range data	(2) 11 bit parallel			
	data quality	(1) 11 bit parallel			
	discrete test points	estimated 4 points	not available	4 + ground	for failur analysis, check out and telemet
	analog test points	estimated 4 signals	not available	4 + ground	

Table B.1 List of DME Interface Signals

Appendix C

Shuttle Software Structure and Organization

C.1 Introduction

The successful implementation of a time-shared data bus for the Shuttle will depend to a large extent not only on the capabilities of the bus system, but its ability to be coordinated with, and effectively used by the computer. An evaluation of the organization and design of the software was not an integral part of this study. However, this appendix has been included to overview and comment on the software concepts under consideration for application in the Shuttle. It presents a general description of the synchronous and asynchronous software control structures, a discussion of I/O operations with each, and identifies their respective advantages and disadvantages.

C.2 Overview of Shuttle Software

The total onboard system has been estimated at approximately 50,000 32-bit words of operating memory, requiring an estimated speed of 200,000 equivalent add operations per second. For purposes of this discussion the onboard software may be broadly classified into two areas: the executive and a set of functional program modules. The executive and supervisory software comprise the following functions:

a) program control (scheduling and dispatching, sequencing control),

b) interrupt supervisor,

c) system subroutines and services,

d) hardware configuration management,

e) common executive data tables,

f) error detection and recovery routines,

g) memory resource management,

h) system monitoring.

The functional software is under control of the executive and supports the phases of the nominal mission: preflight, boost, insertion, orbital operations, coast- and powered-flight, rendezvous, docking, undocking, entry, and landing. The functional areas comprise the following:

a) stabilization and flight control,

b) guidance,

c) powered and unpowered navigation,

d) targeting,

e) displays and controls,

f) onboard checkout and fault isolation,

g) subsystem management.

Ideally, the onboard software and its executive should be designed in a way which is not only tailored to meet the operational requirements of the Shuttle, but is structured to enhance its reliability and ability to adjust to changing needs.

The computational environment of the Shuttle will include three types of jobs:

a) Cyclic tasks. Those tasks which are performed on a regular periodic basis, such as guidance, navigation, telemetry, etc.

b) Demand tasks. These tasks are typically functions which must be performed at a certain time or at the occurrence of a certain event; examples include stabilization and control, turning off jets, etc.

c) Response/request tasks. These are tasks which are performed in response to a pre-selected mode such as the rendezvous

mission mode. Generally these tasks are major sequences of functions initiated throughout the mission by the crew.

An important factor impacting the choice of software organization and control is the level to which the crew will be capable of requesting jobs or major mission modes. If a crew member is allowed interactive communications with the computer, then the job stream will become less deterministic and more random in nature and will require more of an asynchronous structure.

C.3 Synchronous Control Structure

The Shuttle Phase B baseline approaches to data have been based on a synchronous control structure. In a synchronous control structure a predetermined sequence of processing tasks is referenced to some basic time cycle. The main advantage is that scheduling and allocation of the CPU are solved ahead of, rather than in real time.

Mission programs are organized into several major cycles associated with a functional sequence. Each major cycle is composed of a series of operations of minor cycle programs such that the major cycle is completed every N minor cycles. I/O interleaving and memory usage are pre-planned and precedence relationships are built into the sequence.

C.3.1 Description of Synchronous Operation

The following describes a synchronous operation. It is based on a timer-interrupt, fixed schedule, time slice mode of operation. A 20 millisecond interval is used as basic reference frame for the system, providing a minor cycle sampling rate of 50 cycles per second. Under this concept jobs are organized into short routines, and when the executive detects a timer-interrupt (i.e., every 20 milliseconds) it examines the "task schedule tables" to determine which set of routines is to be operated during the next program interval. Each 20 millisecond interval contains all 50/second tasks, and portions of other lower frequency tasks. The minor cycle is operated every 20 milliseconds and a percentage of that time is distributed among the tasks that are assigned to each minor cycle. A background job is run in the slack time before the next minor cycle. Under a command response concept, scheduling I/O in a synchronous structure is similar to the scheduling of tasks. The I/O requirements for each mission phase or major cycle are predetermined and sysnchronized with the structure of tasks operated in the major cycle. The I/O request list is assumed

to be fixed. Since the I/O requirements will have different frequencies, they are incorporated in each minor cycle in correspondence to load balancing of the processing tasks.

For an example, assume all I/O requirements for a particular mission phase are organized into 3 categories of frequencies: 50 times/sec, 5/sec, and 1/sec. Assume that X, Y, and Z are the number of commands in each category. Assume further that a minor cycle occurs every 20 ms and that a BCU is commanded with a list of I/O requests each minor cycle. The average number of I/O operations required to be scheduled each minor cycle are: all of the 50/sec requests, 1/10 of the 5/sec requests, and 1/50 of the 1/sec signals. In a synchronous structure tables of predetermined I/O requests are organized according to sampling frequencies. The appropriate number of I/O entries to command each minor cycle are selected from these tables. The synchronized concept attempts to avoid non-deterministic behavior of I/O, I/O queues, and I/O backlog.

Several types of I/O activity cannot be determined in advance; for example, the command of jets on and off. The I/O scheduler may accomplish this by providing a place for the command in the appropriate list and then causing the BCU to skip the command or incorporate it, depending on the results of the stabilization and control tasks.

C.3.2 Advantages and Disadvantages of a Synchronous Control Structure

This type of executive provides some significant advantages:

a) It has deterministic behavior and simplicity.

b) The requirement for re-enterable programs is either eliminated or minimized, since the environment is not a multiprogrammed one (provided of course, first, that major cycle tasks are not interrupted via the timer interrupt; and second, that they are totally independent of the minor cycle).

c) Conflicts in processor allocation, memory allocation, and data tables are avoided by scheduling and allocating in advance.

d) It eliminates the need for the dispatcher to search a priority queue, which minimizes the executive overhead.

e) Finally since scheduling and allocation are preplanned, theoretically there are no computations or I/O overloads or any degraded response.

However, there are some disadvantages.

a) This type of fixed-sequence executive organization does not provide a structure which allows for external interaction by the operator, or which copes with a random job stream. Jobs must be predetermined and assigned to slots in a sequence and must operate within the basic reference framework. It is not clear at this point whether all Shuttle requirements can be so predetermined.

b) Lengthy calculations must either be broken up into short segments, interconnected in such a way as to meet the requirements of the sequence, or be shifted somehow into the background. There are several activities, as targeting, which involve calculation times on the order of minutes. It is not clear whether it is feasible or cost effective to break them into short segments and interconnect them to form a complete computation. If, however, they are operated in the background and are interrupted by the minor cycle every 20 milliseconds, then they must be multiprogrammed. This implies re-entrant routines and, if they are not totally independent from minor cycle computations, a priority structure.

c) The structure does not seem to possess an inherent flexibility to incorporate changes in the design of the sequences. The requirements to rebalance the load in the fixed sequence after a modification may result in a major redesign.

d) The sequence must accommodate the worst case computational requirement. For example, if the crew is provided the option to display a parameter during a particular mission phase, then the calculation of that parameter will have to be incorporated into the sequence whether or not the crew ever requests it.

C.4 Asynchronous Software Structure

In an asynchronous control structure scheduling and allocation of the processor are accomplished in real time according to the needs of the operating environment. Under this concept processing tasks are assigned a priority which establishes their relative importance to each other. A task with a given priority runs until a wait is encountered, or the existence of a higher priority task is established.

C.4.1 Executive States and State Transition

The distinction between synchronous and asynchrcnous control structure can be illustrated by the "states" in which a task will exist while operating under each structure. In a synchronous structure, tasks are in one of two states: actively running or not running. At any instant of time only one task is in the running state and all others are not running. The transition to the running state occurs when a task's scheduled time slot arrives.

In an asynchronous structure, a task, while present in the system, will exist in one of 4 states: actively running, waiting, ready to run, or "in limbo". The executive insures the proper transition of states depending upon either internal or external stimuli. Refer to Figure C.1. The running state definition is obvious. Note that the "running" state can only be entered from the "ready" to run state. This unifies the dispatcher functions. The waiting state is either a voluntary or involuntary state, depending upon its cause. A voluntary wait would be a wait for completion of I/O, or perhaps some external time stimulus. An involuntary wait would be awaiting resources (i.e., memory) to become available. The state of limbo occurs when the task voluntarily releases the processor without expecting any external stimulus to ready it. The ready state can be entered from all other states and indicates that a job has all the facilities available to it to run. The function of the dispatcher is to pick the most appropriate task from the ready queue and start it running. State changes from wait to ready would occur when the awaited stimulus has occurred. The change from limbo to ready state occurs when a schedule request is issued by some task. The switch from running to ready occurs when a task is preempted by a higher priority task or interrupt.

In summary an asynchronous structure is one in which one or more tasks may be in the ready state awaiting allocation of the processor. In a simplex computer system this is termed multiprogramming, i.e., the concurrent operation of more than one task.

C.4.2 Overview of Asynchronous Operation

An overview of the operation of a general asynchronous executive is illustrated in Figure C.2. The scheduler and dispatcher, once in control, should be able to pick a task and run with it. The scheduler assigns or reassigns task priorities, verifies that all the task resources are available, and maintains

the overall view of real time events. All task starting is done through the dispatcher.

The scheduling function in a broad sense consists of making appropriate entries in task blocks and priority queues so that the dispatcher need only select jobs from the top of the ready list. If there is a number of tasks to be scheduled, the scheduler treats some as more important than others and executes them first. If the dispatch function occurs at some time other than at the end of a program, then a multiprogrammed environment is a direct result.

The interrupt handler "posts" the event complete, makes the task ready if possible, and then passes control to the scheduler to act on the information it has provided.

The resource allocator is invoked as an executive function by the scheduler to test readiness to run, and if not ready, will inform the scheduler of the requirements for readiness. It may also be invoked to test availability of contention items.

I/O in an asynchronous structure is generally scheduled on a demand basis. An active task requiring I/O schedules its request via an I/O queue. The task is placed into the wait state until completion of the I/O request. The I/O control routines operate on the I/O queue and interface the I/O peripheral (i.e., the bus system) to perform the request. I/O is performed asynchronously with other processing tasks in the system. After acknowledging receipt, initiation or completion of the I/O request, the scheduler is informed via a simulated or actual interrupt. The task awaiting the I/O request is then placed into the ready state and awaits processor assignment.

C.4.3 Advantages and Disadvantages of an Asynchronous Structure

Some of the advantages of an asynchronous structure are:

a) it is able to adapt to a random job stream; i.e., it does not require load rebalancing and it can tolerate periodic overload and backlog, because it is, in fact, designed to cope with this problem;

b) it has a greater flexibility for incorporating changes than the fixed sequence approach;

c) coupled with an interrupt mechanism it is more adaptive to a real time environment;

d) its structure does not require long program sequences such as targeting, etc. to be arbitrarily organized into fixed blocks to fit into some fixed cycle or sequence.

The disadvantages are:

a) the multiprogrammed environment resulting from this type of scheduling is more complex and difficult to test and verify;

b) in a real-time system in which a task must be scheduled at specified times, the priority assignment must be chosen and assigned accordingly.

The type of program control ultimately selected will probably be some variation of one of these approaches.

C.5 Computer Interrupts and Their Effect on Organization

One of the basic motivations for structuring Shuttle software in a synchronous control organization is the expected difficulty in handling interrupts. Because of the randomness of operation introduced by interrupts it might be desirable to eliminate external interrupts entirely. However, it is not possible to eliminate all types of interrupts, particularly internal interrupts, which require responses to error conditions. The real problem in verification is not aggravated by the actual* hardware interrupt, but from its effect on the multiprogrammed environment.

The interruption of a running program in response to an external signal was introduced into the computer technology to serve two purposes:

a) to provide rapid response-time to asynchronous events,

b) to eliminate the overhead of polling for the occurrence of an awaited event.

In single-processor systems, particularly dedicated systems, where most or all of the computation is devoted to a single application, the introduction of interrupt-mode computation raises a hazard. At arbitrary times an interruption can introduce what appears to be a parallel task which is, at least conceivably, capable of disrupting the progress of the interrupted task by "invisibly" altering its variables.

---

* By the "actual" interrupt it is meant the hardware transfer to a specific location to perform some minimum function and then resume.

It is reasonable to presume that it is easier to verify a program which will operate from beginning to end (or to some convenient point) without a swap of the processor to some other job. In a sense, the objective is to minimize the scale of multiprogramming in the system.

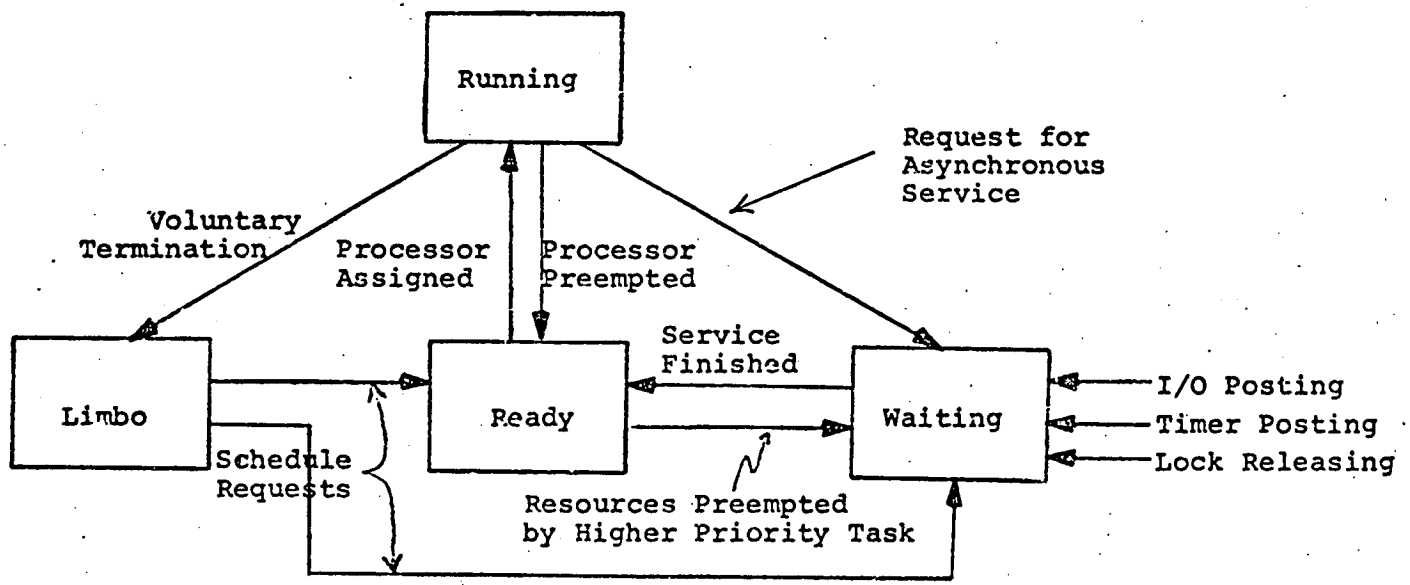
C.6 Summary of I/O Operations Versus Software Structures

A bus I/O transaction once initiated by the computer is independent of the computer software organization. The command/response addressed bus may be directed by a computer with either an asynchronous or synchronous software structure. The main difference will be in the scheduling and dispatching of I/O requests, and in the coordination of I/O with processing.

In the synchronous structure, I/O requests must be preplanned and interleaved with the task processing. I/O requests are dispatched in a list every minor cycle and carried out concurrently with task processing. A synchronous software structure requires a command response bus access method. A polling or contention access method would be difficult to run with a synchronous structure.

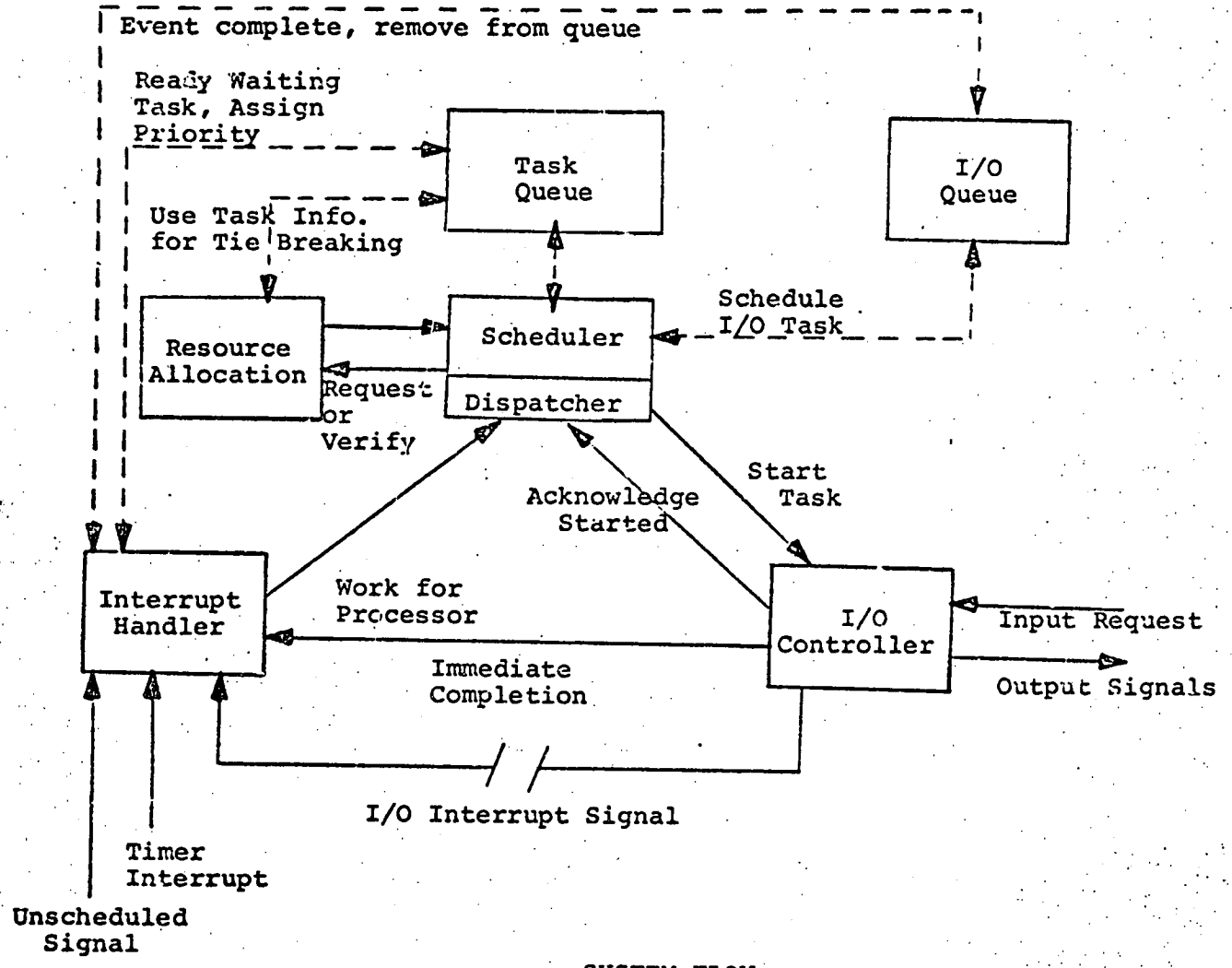
In an asynchronous structure, I/O is scheduled on a demand basis by the processing tasks, and is dispatched to the bus system by the I/O control. After completing the I/O transaction the bus system signals the event (via an interrupt) and the processing task is informed accordingly. The bus system may be commanded with a list from the I/O queue, or with a single request. An asynchronously structured software can command a bus with any form of addressing. It may prove advantageous even with the command/response addressing method.

In either structure, the bus system is designed to accept a command, or list of commands, and executes them as described in Chapter 4. However, the system I/O throughput and response for a given bus design will not be independent of the structure of the software system. Of concern to the bus are the events that occur from initiation of a command from the central computer until its completion by the subsystem. The interactions between the computer and the bus control unit is of specific importance; i.e., what happens to the CPU once an I/O command is initiated, and what happens when the transaction is complete? The answers to these questions will have a great influence in determining the I/O control software performance.



TASK STATES

Figure C-1



SYSTEM FLOW

Figure C-2

END

DATE

FILMED

NOV 22 1971