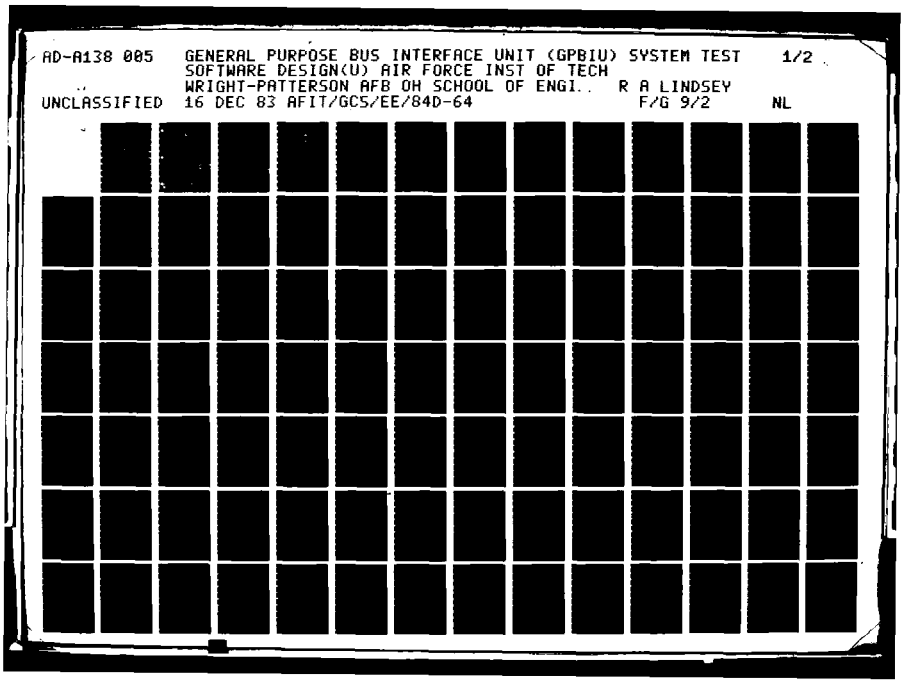
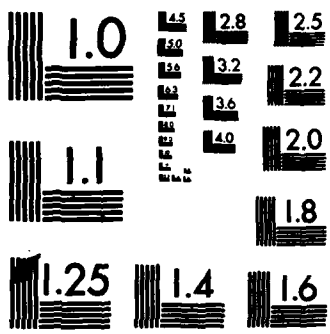

AD-A138 005 GENERAL PURPOSE BUS INTERFACE UNIT (GPBIU) SYSTEM TEST 1/2
SOFTWARE DESIGN(U) AIR FORCE INST OF TECH
WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI.. R A LINDSEY
UNCLASSIFIED 16 DEC 83 AFIT/GCS/EE/84D-64 F/G 9/2 NL



General Purpose Bus Interface Unit (GPBIU) System Test Software Design

THESIS

AFIT/GCS/EE/83D-64 Rodger A. Lindsey
Capt USAF

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GCS/EE/83D-64

General Purpose Bus Interface Unit (GPBIU) System Test Software Design

THESIS

AFIT/GCS/EE/83D-64 Rodger A. Lindsey
Capt USAF

Approved for public release; distribution unlimited.

General Purpose Bus Interface Unit
(GPBIU) System Test Software Design

THESIS

AFIT/GCS/EE/83D-64 Rodger A. Lindsey
Capt USAF

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY (ATC)
AIR FORCE INSTITUTE OF TECHNOLOGY

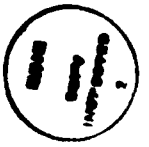
Wright-Patterson Air Force Base, Ohio

AFIT/GCS/EE/84D-64

General Purpose Bus Interface Unit (GPBIU) System Test Software Design

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
in Partial Fulfillment of the
Requirements for the Degree of
Master of Science



by
Rodger A. Lindsey
Capt USAF
Graduate Computer Systems
December 1983

Approved for public release; distribution unlimited.

Preface

This report outlines high level system requirements, specifications and sample design for developing a software test package for functionally testing a General Purpose Bus Interface Unit (GPBIU). These tests are needed to verify GPBIUs are functioning properly because they are used in direct support of standards testing of avionics interconnected subsystems (components).

I wish to express many thanks to Lt Col Hal Carter, my thesis advisor, for providing his hardware expertise and for helping me keep this thesis on track.

I wish to also acknowledge the assistance given by Captain Bruce Varnum, my sponsor, Sgt Jim Bennett, Mr. Tom Zawodny, Mr. Harold Alber, and Mr. Clyde Johnson all of Air Force Systems Command, Aeronautical Systems Division. All gave valuable time and information to inform me of the current development, operation, and testing procedures of the GPBIU.

Thanks also are due Dr. Henry Potoczny, my third thesis committee member. He provided much encouragement and guidance throughout my stay at AFIT.

I dedicate this work to my wife, Marian Starr Lindsey, and our daughter, Virginia Abigail, born to us during my stay at AFIT. Thank you.

Rodger A. Lindsey

Contents

List of Figures

Abstract

This project presents high level system requirements, specifications and sample design for testing General Purpose Bus Interface Units (GPBIUs). The GPBIUs are being developed and used by the Air Force Systems Command, Aeronautical Systems Division in direct support of avionics interconnected subsystems (e.g. radar, navigation, target identification, sensors, displays, etc.).

An overall system chart is developed and includes ~~the~~ four major test systems:

1) Host System Validation;
2) Mode-Test-I (two GPBIUs, one host);
3) Mode-Test-II (two GPBIUs, two hosts); and
4) Off-Line Tests.

This project addresses only Mode-Test-I for two GPBIUs and one host. The design for the test system is highly flexible and maintainable and adaptable to changing technologies and requirements as upgraded GPBIUs are developed.

GENERAL PURPOSE BUS INTERFACE UNIT (GPBIU)

SYSTEM TEST SOFTWARE DESIGN

I. Introduction

The General Purpose Bus Interface Unit (GPBIU) is a software directed microsequencer which serv' as a communications interface between a Digital :quipment Corporation (DEC) UNIBUS and a MIL-STD-1553 mul)lex data bus (see Figure I-1). The bus interface unit .s great applicability for testing avionics systems when independent GPBIUs can aid in the simulation of avionics subsystems (e.g. airspeed, radar, or fuel) by making the data available on a global bus to other terminals, monitors or controllers which use the data (Figure I-2).

Background

Modern sophisticated aircraft require a large amount of in-flight data processing for effective mission performance, and, as avionics and control systems have become progressively more complex, new concepts for integrating these systems have been developed. Earlier avionics and control systems were independent or self-sufficient, but as aircraft became more complex and the onboard systems became integrated, these systems increased aircraft weight and

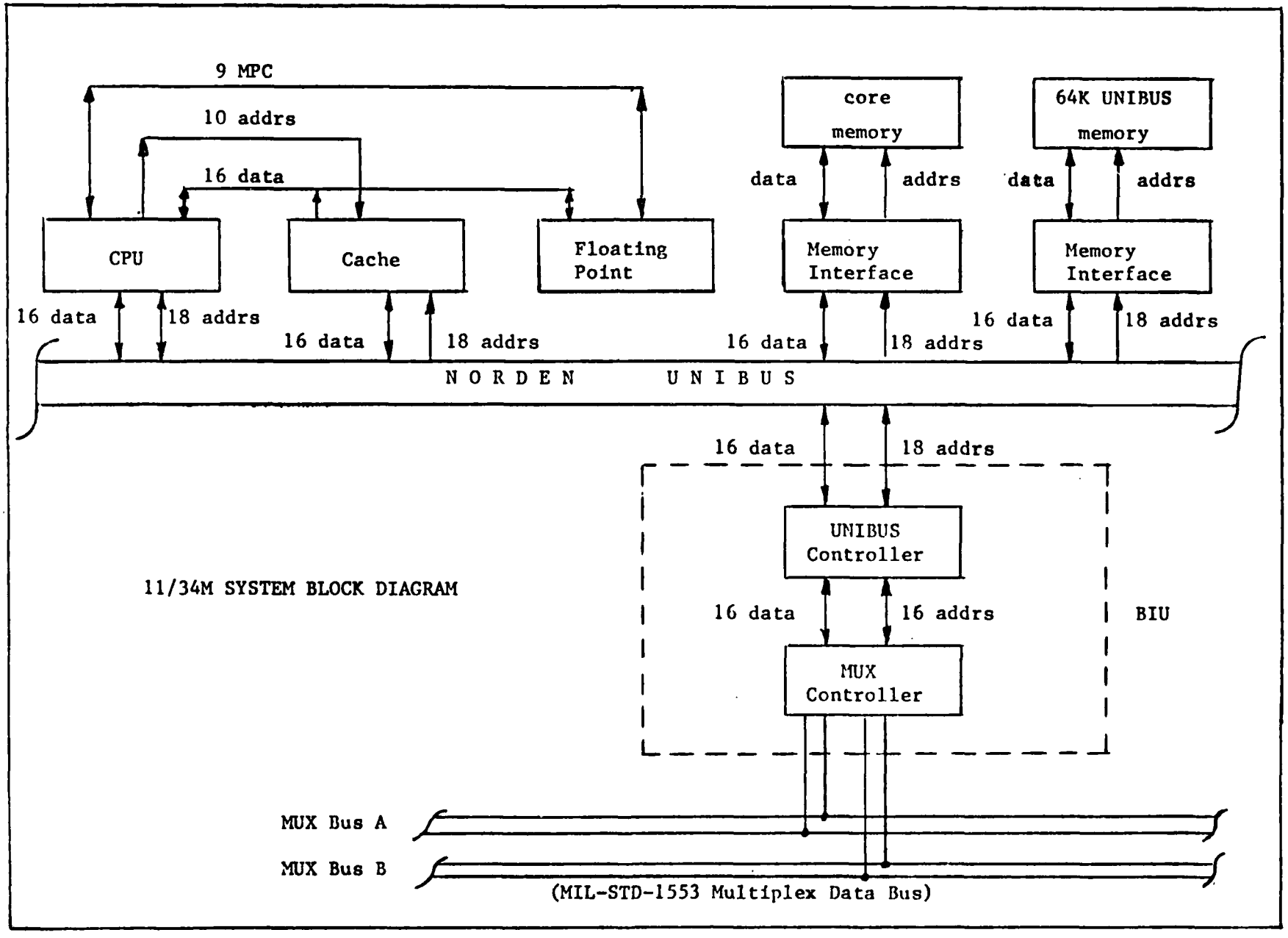


Figure I-1. GPBIU Interface. (Ref 10)

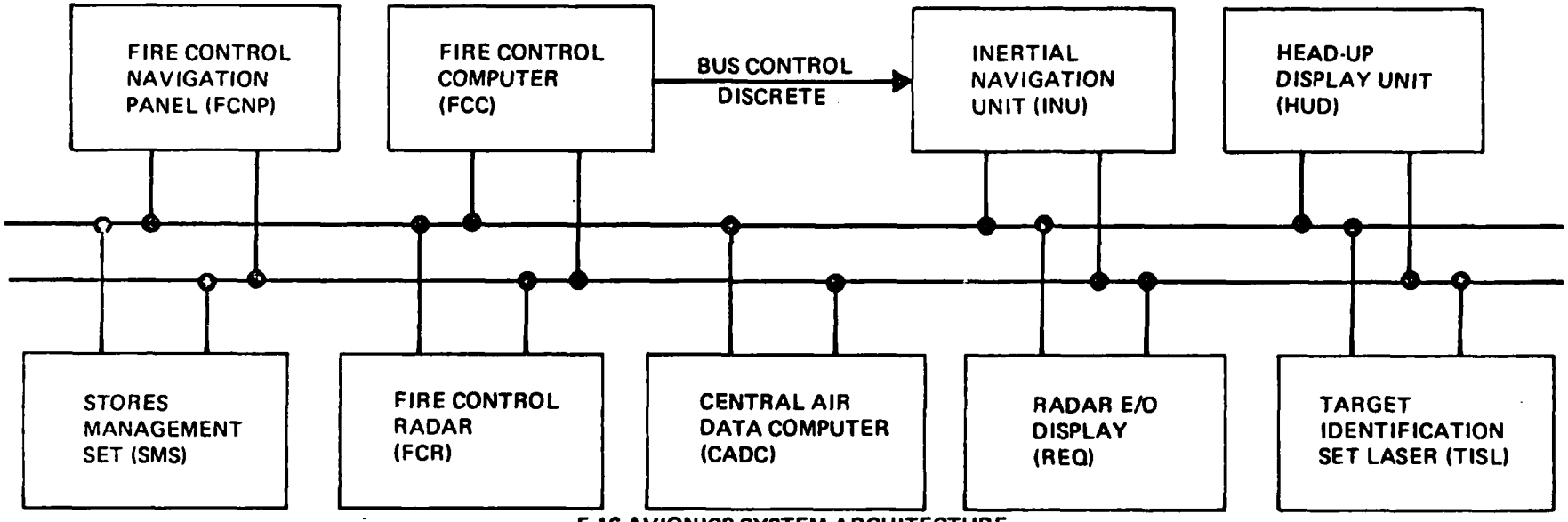


Figure I-2. *F-16 Avionic System Architecture.* (Ref 4:33)

required more space and electrical power (Ref 4:2.2). An early solution was to centralize the systems, relying on one central computer to process and distribute information to various components. The main fault with this system is the heavy dependence on the central computer. If the computer, or the interface to it, fails, the aircraft mission performance and reliability may degrade drastically. Another problem with the centralized approach was that the subsystems were designed with little concern for interconnection with other systems. As a rule, each subsystem was specialized and the interfaces reflected this specialization. The central computer I/O circuitry was designed to perform the function of ordering or queueing the incoming and outgoing data. This alone was a burden since the computer was often small compared with the size and complexity of the I/O (Ref 4:2.2).

Although the centralized concept upgraded the capability of the overall system, it was determined that the central computer's complexity could be lessened by partitioning and distributing the I/O circuitry and by using I/O data multiplexing. Multiplexing simplifies I/O and makes information exchange convenient because the information transfer medium is reduced to a single wire pair (i.e. bus).

Current avionics integration methods, often referred to as multicomputer systems, are made possible by today's smaller processors and lower cost digital components.

Computation is distributed and several computers (microprocessors) replace the more powerful central computer. Each processor, or group of processors, receives raw data from some aircraft sensor, processes it, formats it, and makes it available on a global bus to all the other microprocessors that need it. Even newer integration approaches use additional processors and buses to functionally partition the avionics systems. This functional partitioning further eases the integrating problem by allowing design of the functions to be developed more independently of each other prior to completing the total avionics system integration (assuming of course all functions are accomplished according to the same standards). This decentralized system concept should be highly reliable, since the loss of one microprocessor does not result in complete system failure and the mission can continue in a degraded mode. However, problems have surfaced with this new integration approach. Three of these are: software complexity, data communication and synchronous operation.

Contributing to software complexity is the necessity to perform message transmission control and message integrity checks (referred to as cyclic redundancy checks (CRC)). Some general algorithms exist which will provide insight to handling the required cyclic redundancy check on message transmission (Ref 7). Interfacing to memory is another major area of software complexity for producing test programs when dealing with Air Force System's Command,

Aeronautical Systems Division's (ENASF) current computer configuration. Currently, ENASF has a total of four DEC computer systems with MIL-STD-1553 interfaces. These are a VAX 11/780, a PDP 11/55, and two PDP 11/34s. Between the PDP 11/55 and PDP 11/34s, there is no difference between either the hardware or software interfacing for MIL-STD-1553; however, the VAX 11/780 interface is slightly different in that the GPBIU is not capable of using the host system's memory as it is on the PDP 11s. This qualifies the need for UNIBUS memory on the VAX (64K words is provided (refer to Figure I-1)). In the case of the PDP systems, UNIBUS memory is also the computer system's memory. Therefore, some type of control needs to be maintained so that the GPBIU does not write over areas in memory that are also being used by the host. The standard method for this control is to create a system common block and have each piece of software directly map to this common block. In the case of the VAX 11/780, the UNIBUS is just an adapter so that many of the products designed for the PDP systems may be used on a VAX also. DEC has made a provision for memory to be used on the VAX UNIBUS and has provided a method for disabling the system usage of the mapping registers to this memory. Although this allows the memory to be placed on the UNIBUS and allows the GPBIU access to this memory, it does not allow applications software to easily interface to it. This has to be accomplished by special applications software and common blocks that are provided in existing

documentation for programmer usage (Ref 1).

The MIL-STD-1553 communications bus consists of two cables called busses. One is the primary bus and the other is an alternate in the event problems are detected on the primary bus during a message transmission. Switching is software selected by the GPBIU in Bus Controller (BC) mode. This switching allows both busses to be utilized as primary bus and alternate bus, thus increasing user confidence that at least one bus will be functioning at all times.

With so many messages (often at differing transmission rates) being passed, traffic must be synchronized so that the data is not distorted or lost. Some sort of interface is necessary to accept messages at one rate and allow storage or transmission at another rate. The GPBIU provides this interface. In actual avionics subsystems, each subsystem, or terminal, as a possible receiver of a message must check each message being transmitted to determine if the message is one for it to receive. This is implemented partially by hardware in actual systems but is simulated in the current applications software routines for the GPBIU. Also a concern in actual avionics systems is the problem of handling message header errors and/or encoding errors; however, this will not be a problem for testing the GPBIU since messages are created and loaded into memory before an applications program is executed to exercise the GPBIU. In other words, testing will be done using "canned" message blocks.

Problem

Air Force Systems Command, Aeronautical Systems Division (ENASF) is the Office of Primary Responsibility (OPR) for MIL-STD-1553, Aircraft Internal Time Division Command/Response Multiplex Databus. The branch is currently developing General Purpose Bus Interface Units (GPBIUs) in direct support of standards testing of avionics interconnected subsystems (components). Currently, two GPBIUs are installed. Eventually, ENASF hopes to have eight GPBIUs installed/operational. Each GPBIU will support a separate avionic system. At present, there is no comprehensive software package available to test all functions of the GPBIU. The most immediate reason why such a test package is needed is because the contract with Digital Equipment Corporation (DEC) does not require them to support any non-DEC equipment. Since the GPBIU was developed and installed in-house, it is necessary to show whether or not it is the cause of any system failures. this will allow DEC to perform its diagnostic testing more confidently and in accordance with the contract. Showing that the GPBIU is not at fault will also save the Air Force money by eliminating service calls to DEC which result in DEC pointing the finger at the GPBIU. Instead, justified vendor maintenance calls will result. The tests are needed, then, to give in-house operators the capability to isolate equipment failures along the interconnected avionics

subsystems and within the GPBIUs.

Since development of the GPBIU, ENASF personnel have written many applications routines for accessing and/or using the GPBIU; however, these routines are not comprehensive nor well documented, are scattered about and are cumbersome to use for someone not familiar with their function. (See Appendix A for a hierarchy chart of one existing system.) A comprehensive test package will enhance operator performance and analysis of proper functioning of the GPBIU. The test package will also facilitate debugging of subsequent added subsystems, thus allowing them to become operational sooner. Currently it takes a knowledgeable technician two to four days to check out a GPBIU. Hopefully the use of an integrated on-line software test system will reduce this time to minutes.

Objective

The objective of this thesis is to provide ENASF with a high level preliminary design package for functionally testing a GPBIU. Because error detection approaches through software are not well defined, the task of specifying and defining system requirements must precede this design. Chapter two addresses these system requirements and chapter three addresses the subsystems requirements from the standpoint of <u>what</u> the system is to accomplish rather than how the system will accomplish those requirements. The detailed design, coding, testing, implementation and

integration of selected subsystems are outside the scope of this thesis effort; instead, they will be proposed as a progression of follow-on thesis efforts.

Scope

This thesis effort is primarily concerned with specifying and defining preliminary system requirements for designing a documented test package for testing the GPBIU in the four modes described below. Additionally, a sample design (flowchart and Program Design Language (PDL)) of a portion (executive and driver modules) of the overall system will be presented in chapter four.

1) Bus Controller (BC) mode: The primary function of a bus controller is communication control of data bus traffic (Ref 4:3.14). There currently are four functions which allow the GPBIU to perform as a MIL-STD-1553 Bus Controller device. These are:

a) BC --> RT, Bus Controller to Remote Terminal data transfers;

b) RT --> BC, Remote Terminal to Bus Controller data transfers;

c) RT --> RT, Remote Terminal to Remote Terminal data transfers;

d) MODE, Mode Command transfers.

The above functions are discussed in chapter three under "Configuration Setup". There is a fifth BC mode

function, TEST, which allows an internal test to be performed within the GPBIU (Ref 1). The Bus Controller mode is the only mode for which the capability exists for performing such a test. This hardware controlled "internal wrap" test operates essentially by preventing data from being transmitted via the MIL-STD-1553 bus. Instead, the data is routed back to the input ports of the GPBIU and it is verified that the message received is the same as the message sent (both messages are stored in memory). Although this wrap test provides a high degree of confidence when testing a GPBIU (a technician has stated that approximately 80 - 90 percent of the GPBIU's circuitry could be tested using this test), it has a major drawback. If the host computer is the VAX 11/780, the VAX system must be shut down and the board removed to a dedicated PDP 11/55 where this test is run. The VAX must also be shut down again to remount the board after testing. This is a major inconvenience for other users of the VAX system. Also, powering a system up and down too often may cause hardware or electrical problems and is not a recommended standard procedure.

2) Single Address Remote Terminal (SART) mode: Single Address Remote Terminal mode allows a GPBIU to simulate and respond as a single non-bus controller/non-bus monitor device (i.e. as a remote terminal). Different message block locations in memory may be used to represent the remote terminal. This allows the capability to simulate an

avionics subsystem with a single piece of hardware (Ref 1).

3) Multiple Address Remote Terminal (MART) mode: Multiple Address Remote Terminal mode allows a GPBIU to simulate and respond as multiple remote terminals. Different message block locations in memory represent the various remote terminals. This allows the capability to simulate multiple avionics subsystems with a single piece of hardware (Ref 1).

4) Bus Monitor (BM) mode: Bus Monitor mode allows a GPBIU to passively monitor traffic on a MIL-STD-1553 data bus. In this mode, the GPBIU only "listens" to data. An extension to this mode allows monitoring of the data bus for activity. No communication by any station means a primary bus controller failure has occurred and the backup bus controller needs to be activated (Ref 4:3.37) (see Figure I-3). Another possible bus control failure occurs when an active bus controller retains control of the bus indefinitely (Ref 4:3.37). The Bus Monitor is capable of detecting this and taking the appropriate action to activate the backup bus controller.

Approach

In the early stages of this project, it became clear that this design project represented a "wicked" problem. A wicked problem is defined as a particularly elusive one, in that the solution of one of its aspects may reveal an even more serious difficulty (Ref 9:25). This translates to

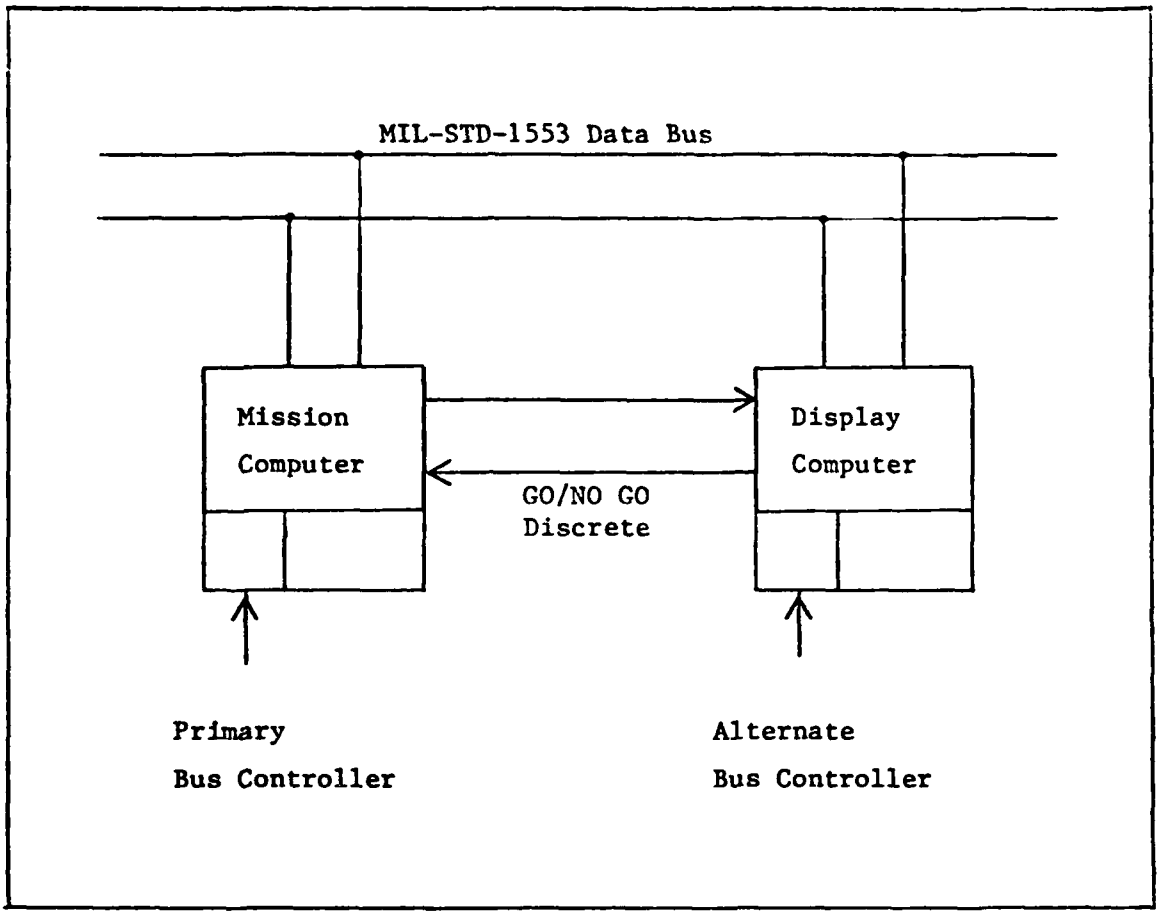


Figure I-3. Bus Controller GO/NO GO Discrete. (Ref 4:3.37)

something like, "the more one learns about a problem, the more there is yet to learn." Lawrence J. Peters (Ref 9) includes many properties of wicked problems and concludes by saying in essence, "wicked problems are troubling in the sense they do not paint a rosy picture of design but they are comforting in the sense they provide some verification that there is no easy way to reduce these problems even if we are diligent." I have tried to be diligent and break the problem of designing a GPBIU test system down into what are generally the necessary tests and have further offered some guidelines for their congfiguration. I can make no guarantee that the proposed tests will test 100 per cent of the GPBIU circuitry or paths. Because the fault dictionary would become too large, exhaustively testing a GPBIU is not feasible. There are too many bit combinations to allow exhaustive testing in a reasonable period of time. Instead, controlled random testing is recommended. In this method, confidence in the correct functioning of a GPBIU can be obtained by maintaining certain areas fixed while increasing the frequency of change in other areas. A "best guess" of the user may also aid in determining those areas held constant and those areas manipulated or changed. (Common sense is a viable input to testing.) Allowing tests to run for longer periods of time will also raise user confidence that a GPBIU is functioning properly.

The following accepted principles of structured design were applied: analyze the problem, produce preliminary

requirements and specifications, partition the problem into modules and develop the modules as "black boxes" (i.e. independently) (Ref 8:19-21). The system will be highly flexible and maintainable and adaptable to changing technologies and requirements as upgraded GPBIUs are developed. Modifications may need to be accomplished as advances in GPBIU design are accomplished and as work progresses to support the entire test system.

Overview

Chapter two covers the system requirements for the overall GPBIU test system and identifies necessary tests and what these tests check. Chapter three covers the subsystems requirements and specifications (in the form of structure charts and schematics of hardware connections) for testing a GPBIU on a single host computer. Chapter four offers a sample design for the GPBIU Test Executive and selected lower modules. Chapter five presents findings, recommendations and concluding remarks.

II. System Requirements

Introduction

This chapter provides a high-level description of the proposed requirements for the GPBIU software test package. These requirements were arrived at through interaction with the sponsoring user organization. At the request of the using organization, graphical and textual materials are used to state the requirements of the GPBIU test system. The graphical techniques are used for clarity to communicate to the using organization what this thesis proposes. Hierarchy charts and hardware schematics were the methods most desired by the user to show the managerial and functional modules.

It is assumed the tests are to be run following a perceived malfunctioning of a GPBIU.

Functional Requirements

An operator should have the capability and flexibility to select one, several or all tests to be run; therefore, a menu-driven user-friendly executive program will be designed to control which configurations and modes are tested based on user selection. Good programming practice allows the software to do as much of the selecting as possible; therefore, the system will be designed to have minimal operator intervention to perform the tests on a GPBIU. However, an operator will have to assure the proper hardware configuration and choose among various options for the

different tests.

To reduce the amount of time currently required by off-line GPBIU tests and to reduce the amount of VAX user inconvenience because of powering down the VAX to remove a GPBIU, it is desired that on-line testing of the GPBIU be sufficient to determine if the GPBIU is functioning properly. Should the on-line tests by the operator reveal no failures within the GPBIU, it will be assumed the GPBIU is functioning properly. However, should the tests reveal a malfunctioning in a given mode, the GPBIU will need to be further tested and may have to be removed and given to a technician for further 'off-line' testing.

The four following test subsystems as a minimum should be included in the overall system test design (see Figure II-1):

1) Host-System-Validation;
2) Mode-Test-I (one host, two GPBIUs);
3) Mode-Test-II (two hosts, two GPBIUs); and
4) Off-Line Tests.

The chart of Figure II-2 shows the recommended tests and the functional areas they affect. Only Mode-Test-I is discussed in detail in this thesis. The Host-System-Validation and Mode-Test-II modules will be left as a progression of extensive follow-on thesis efforts required to accomplish the eventual and complete GPBIU software test system. The experience gained in each of the

follow-on investigations will provide enhancements to the GPBIU software test system.

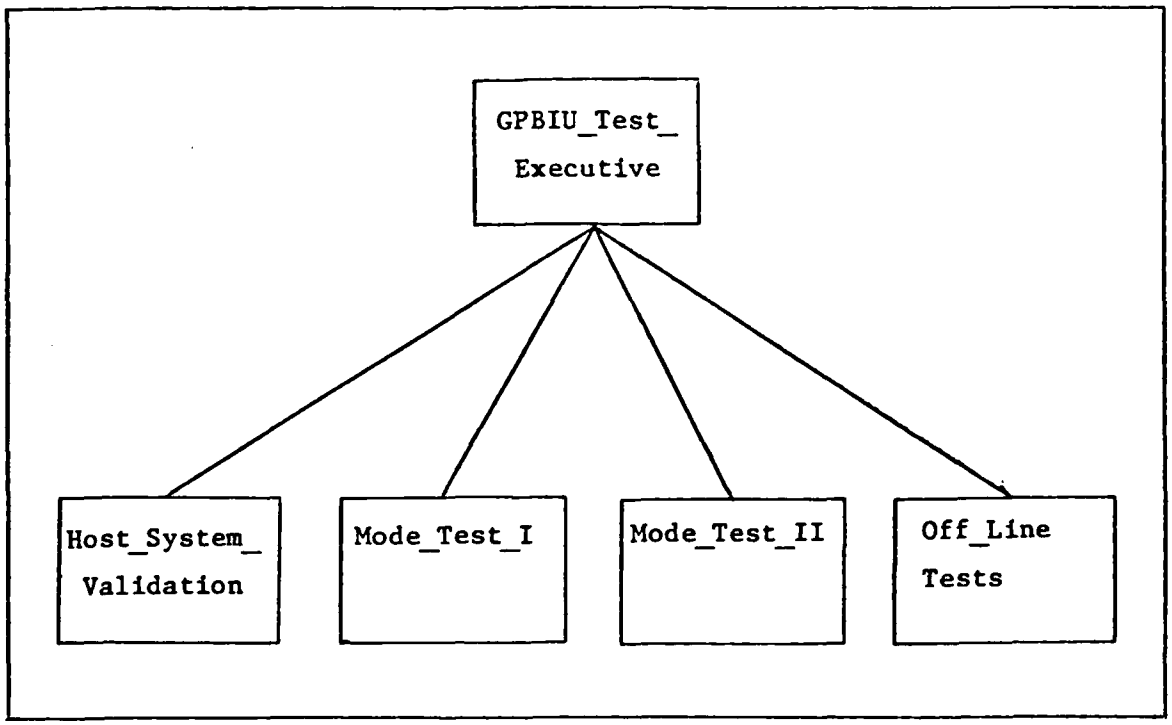


Figure II-1. System Diagram.

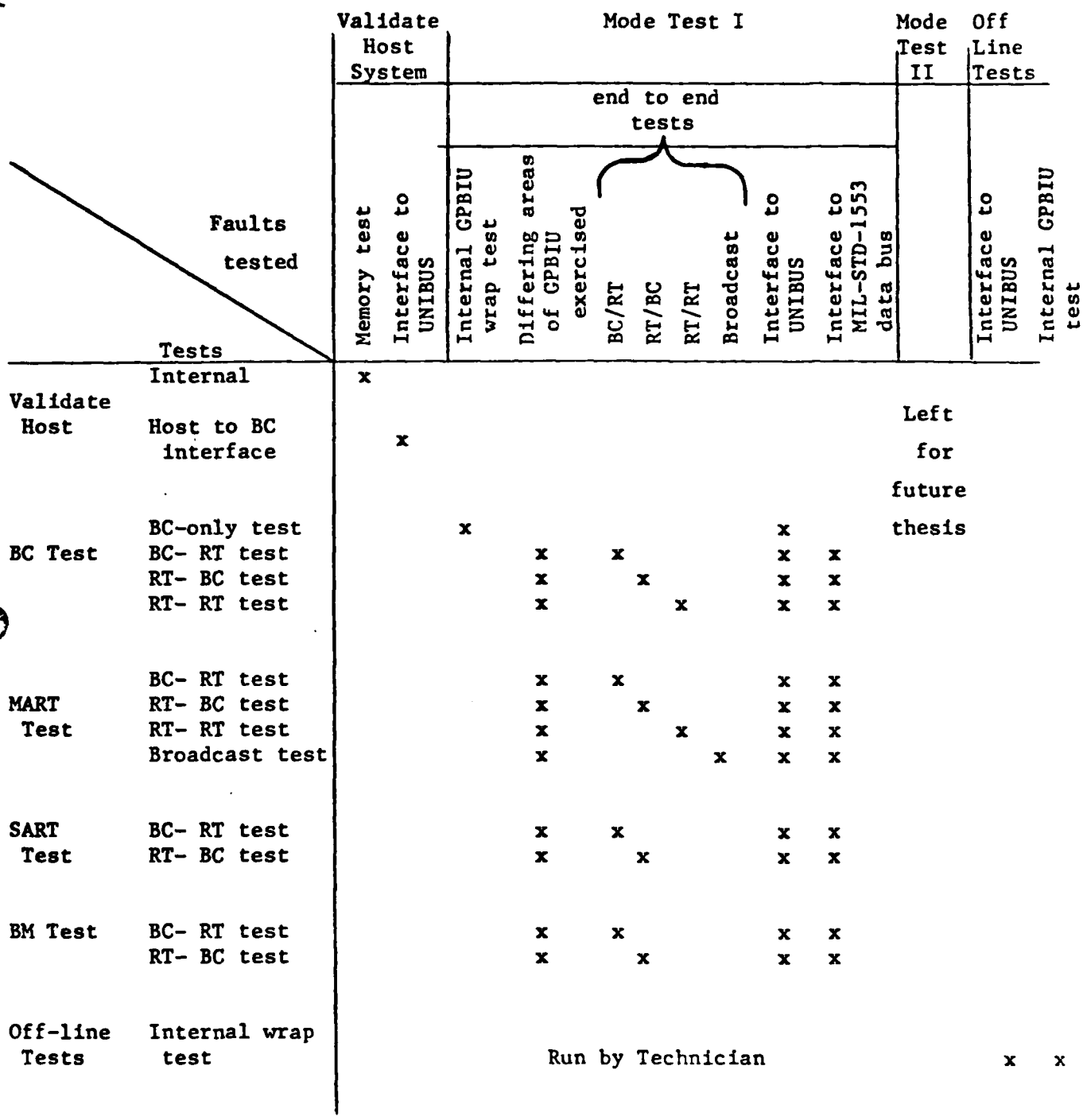


Figure II-2. Test Impact Chart.

The following sections discuss the rationale as to why the above tests are necessary and sufficient for testing a GPBIU.

1. Host Computer Validation

Obviously a GPBIU can only be confidently and properly tested if one is confident that the host computer on which the GPBIU is installed is known to be functioning properly. Therefore, it is logical to suggest that the Host-System-Validation subsystem always be executed prior to any other tests (see Figure II-3).

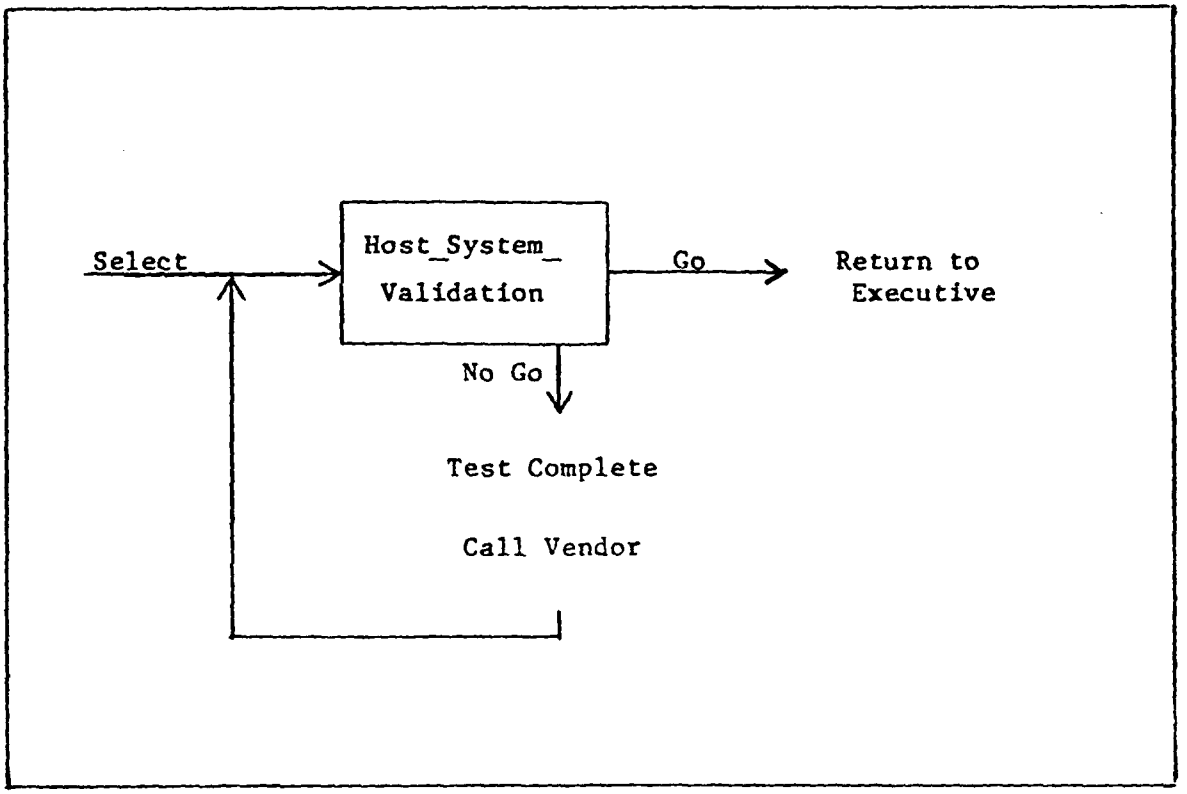


Figure II-3. Host_System_Validation Diagram.

Validating correct functioning of the host computer(s) is outside the scope of this thesis but is a necessary test to accomplish prior to testing a GPBIU. This test will be left for ENASF personnel. They are familiar with their machines and have appropriate vendor supplied diagnostic routines for performing memory checks, timing checks, correct functioning of the UNIBUS, etc. If they cannot determine with the provided routines where the problem is, the vendor should be called in to diagnose and correct the malfunction.

Inputs

An operator may select to run the host validation routines through a GPBIU-Test-Executive program (see Figure II-1). The Host-System-Validation subsystem should issue the calls to the appropriate vendor supplied test routines.

Outputs

Outputs of the Host-System-Validation subsystem shall be as determined by the vendor or ENASF personnel. If a problem is reported, the vendor should be called in to perform further hardware checks.

Constraints and Assumptions

The host computer shall be configured to support the

GPBIU while these tests are being run. Should no errors be flagged, the operator may proceed to the next test subsystem. However, if the diagnostic testing reveals a problem which cannot be corrected by the operator, the vendor should be called in to check the computer. Since the GPBIU is a non-DEC piece of equipment, the vendor representative may request the GPBIU be removed before vendor testing begins. This request should be complied with since the purpose of the Host-System-Validation subsystem is to validate the host computer is functioning properly. The GPBIU can be properly tested only if the host is operating correctly.

2. Mode-Test-I

The Mode-Test-I module (see *Figure II-4*) will allow testing of all GPBIU modes and configurations. The Roman Numeral 'I' in the subsystem title refers to a single host computer. The Mode-Test-I module will be a menu driven executive to allow an operator the flexibility to select one of four GPBIU modes of operation for testing.

The operator will be prompted to put the hardware into the desired physical configuration prior to actually starting the test in the selected GPBIU mode. A status of "good" will be returned if no errors are detected or a status of "bad" will be returned if any fault is discovered. The operator will have the option of obtaining a printed copy (memory dump) of the memory locations found in fault or

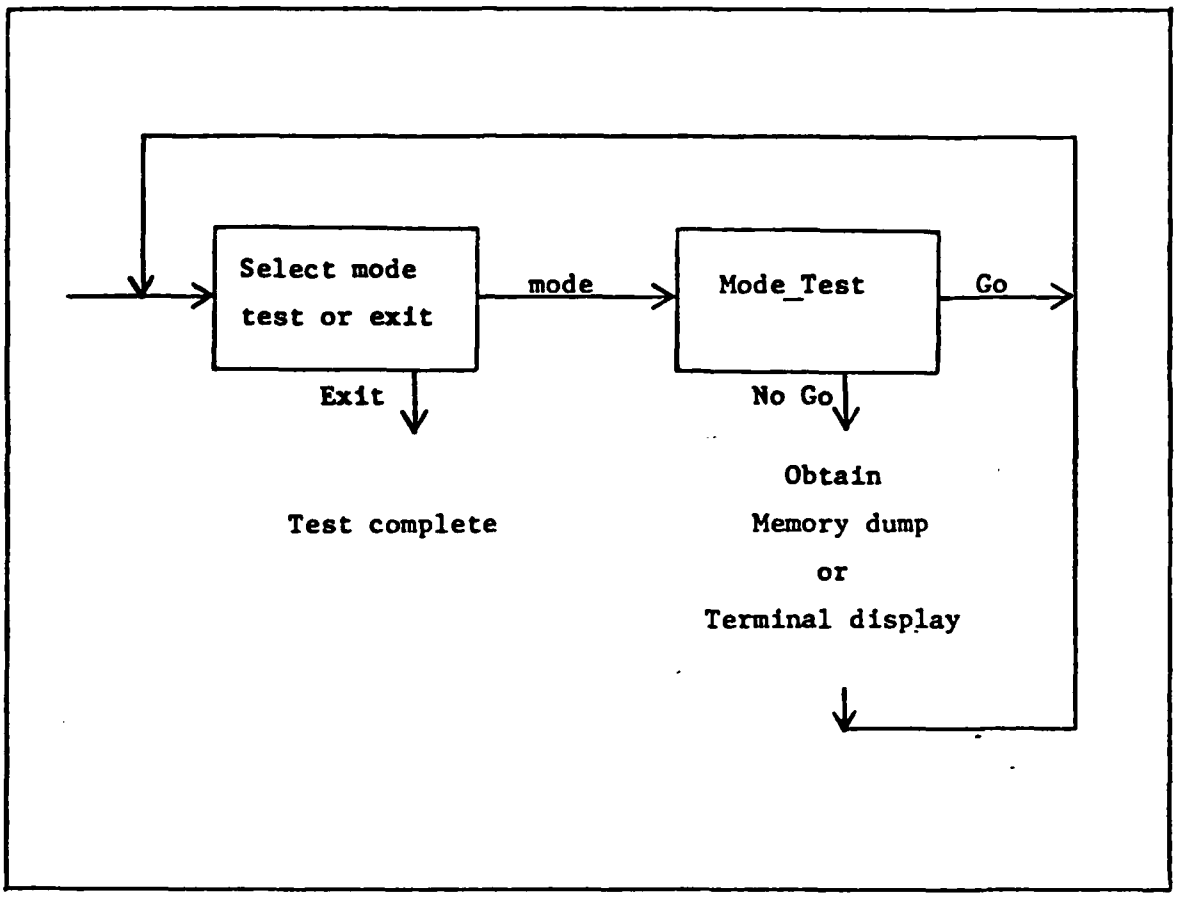


Figure II-4. Mode_Test_I Diagram.

of obtaining only a display to the terminal screen. This test subsystem is addressed more thoroughly in chapter three.

Inputs

Mode-Test-I is to be operator selected from the GPBIU-Test-Executive program (see Figure II-1). The Mode-Test-I module shall allow the operator to select a desired GPBIU mode of operation test (BC test, SART test, MART test or BM test) or exit back to the GPBIU Test Executive program. Further, selected tests of valid data transfers shall be accomplished in the selected mode. (NOTE: Normally, testing should first be accomplished in the mode which was being run when a problem was perceived.)

Outputs

This subsystem shall inform the operator whether or not a fault was detected during its execution. If no fault is detected in a selected mode, the operator shall have the opportunity to select another mode for testing. Should a fault be detected, a terminal display or a memory dump of the message blocks shall be provided for inspection (see Figure II-4).

Canned message blocks containing the data for sending/transmitting are loaded into the host computer's memory or into the UNIBUS memory prior to exercising a

GPBIU. Therefore, the primary means of verifying correct functioning of a GPBIU will be in verifying that the data in these message blocks correctly passes through the GPBIU over the busses and is likewise stored in an area of memory reserved for storing transmitted messages (blocks). Verification will be accomplished by comparing the contents of the sending message block with the contents of the receiving message block (i.e. an end-to-end test).

Test Operations

If the data word contents as stored are the same as the data words transmitted, the GPBIU will be assumed to be operating correctly for the chosen mode. If the data word contents differ, a software test routine will provide a terminal display or a memory dump of any differing data words. An operator or technician can then examine the contents and make a determination if there is need to run other tests or have a technician remove the GPBIU for off-line testing. Further tests may be run in other modes to determine if errors continue to be observed. If an operator is unable to state why or where the problem is occurring, it will be necessary to have a technician remove and test the GPBIU.

Constraints and Assumptions

The software test designer must know the specific characteristics of the bus interface hardware design to understand the roles of the processor and the bus interface hardware (Ref 4:5.1.3). The hardware configuration for this test shall consist of two GPBIUs installed on a single host. It will be assumed that a minimum configuration of two GPBIUs will be available at all times and the design will reflect this assumption. Two GPBIUs will allow all modes to be tested. Having two GPBIUs, one as the primary bus controller and the other as the backup bus controller, will also allow for bus controller switchover failures (see Figure II-5). The following is extracted from MIL-STD-1553 Multiplex Applications Handbook:

> "Bus controller operation in the event of failure is important to an integrated data bus system. Several methods for switchover to backup bus controller are being applied in military systems today. The key to successful backup bus controller design and implementation is the ability to meet these criteria:
>
> a. Primary bus controller recognizes internal failures and ceases operation.
>
> b. Backup bus controller recognizes the failure of primary controller and initiates action to take over control.
>
> The simplest approach is the bus controller GO/NO GO discreet (see Figure I-3), which is shared between the primary bus controller and the backup bus controller. Periodically ..., the bus controller must signal the backup bus controller indicating that it is still in control of the bus and is in good health (i.e., both hardware and software)." (Ref 4:3.36-37)

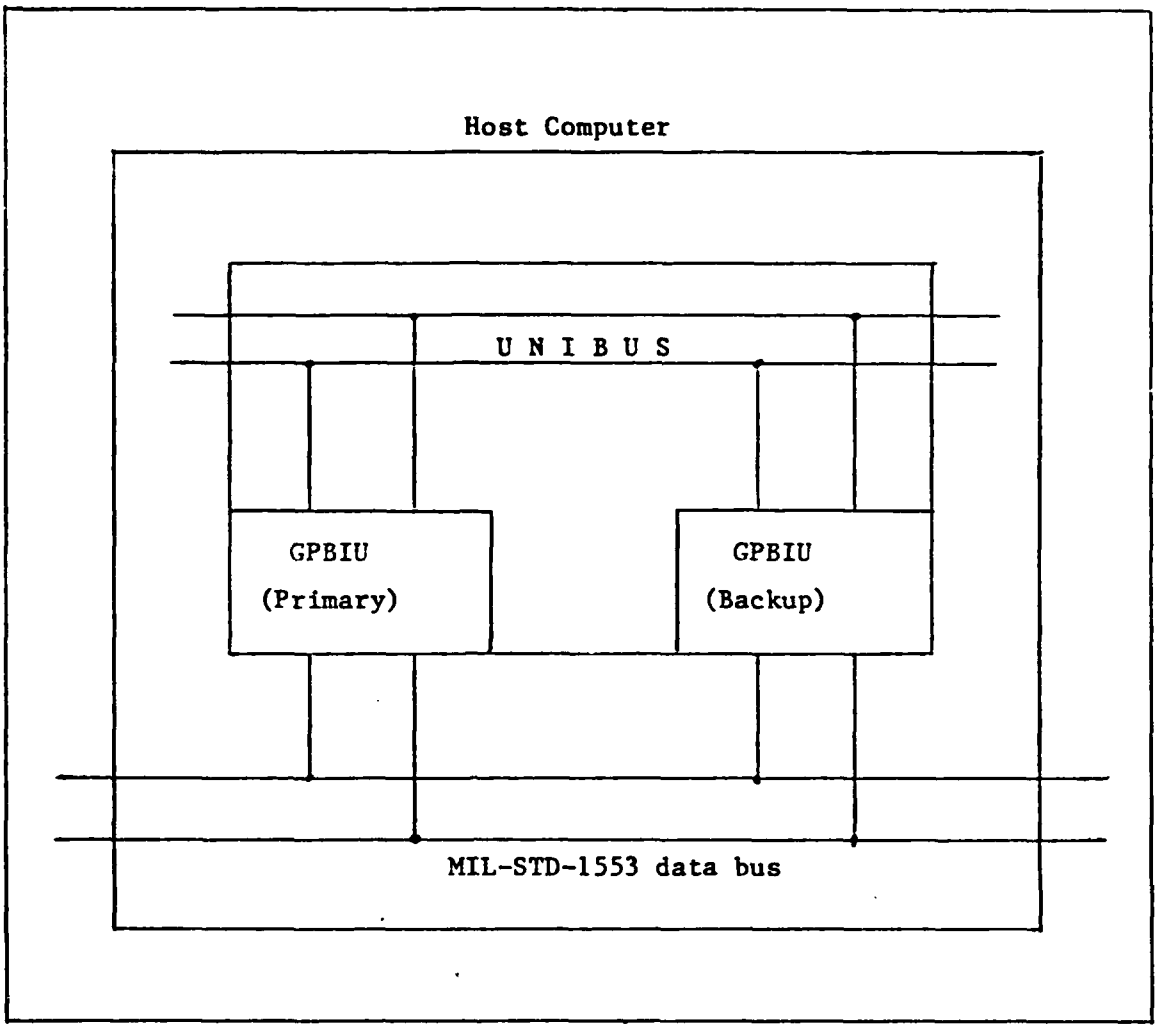


Figure II-5. Single Host - Two GPBIU Configuration.

Dealing with bus controller switchover failures is outside the scope of this thesis. It is mentioned to support the need for testing with two GPBIUs installed on a single host computer.

The design also reflects that if no faults occur, an operator is to assume the GPBIU is functioning properly. This is hoped to be the normal mode of operation.

The design of the test system shall be modularized and accomplished from high level design to low level design to allow change, expansion, flexibility and maintainability in future test designs as the GPBIUs are modified/upgraded. Appendix B shows the high-level structure charts for the proposed GPBIU test system. Chapter four and Appendix C show the flowcharts and Program Design Language (PDL) of selected modules.

3. Mode-Test-II

This subsystem is a necessary member of an overall system design for effectively and thoroughly testing a GPBIU. The Roman numeral "II" in the subsystem title refers to two host computers, each of which will contain a GPBIU (see Figure II-6). In actual avionics systems implementation, there may be several GPBIUs installed on various processor controlled devices. These GPBIUs must communicate over the MIL-STD-1553 multiplex data bus. One GPBIU may even have to take over for another as primary bus controller if the original primary bus controller

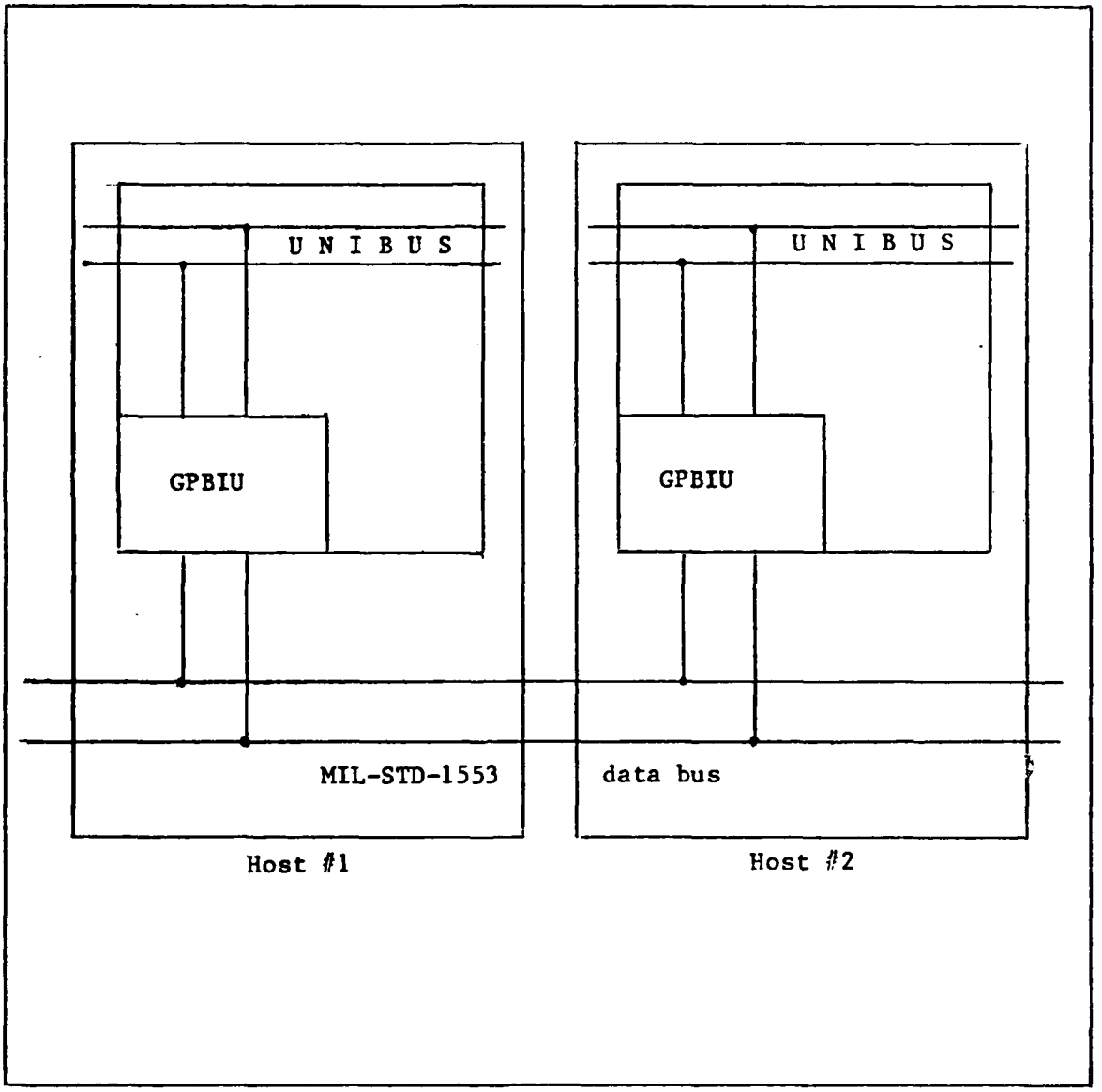


Figure II-6. Two Hosts - Two GPBIU Configuration.

malfunctions. Mode-Test-II will allow more comprehensive and thorough testing of a GPBIU.

Although the physical configuration of the hardware may differ somewhat, the composition and design of Mode-Test-II shall be very similar to that of Mode-Test-I (see Figures II-7 and III-1). Inputs and outputs will be essentially the same but further constraints and assumptions may need to be made in the subsystems specifications. This subsystem will be left for a recommended future thesis project by ENASF.

Constraints and Assumptions

The same constraints shall apply to Mode-Test-II as applied to Mode-Test-I except that the configuration for this test shall consist of a minimum of two host computers each with a GPBIU. The two hosts shall pass messages between themselves through the GPBIUs in their selected modes. These messages will then be compared to insure each device is "seeing" the same data.

4. Off-Line Tests

Off-Line tests should be accomplished when any of the previous tests indicate a problem or malfunction (see Figure II-8). An operator or technician shall be called in to remove the GPBIU board from the host and carry it to an appropriate test area. Current performed off-line tests appear to be thorough but require two to four days

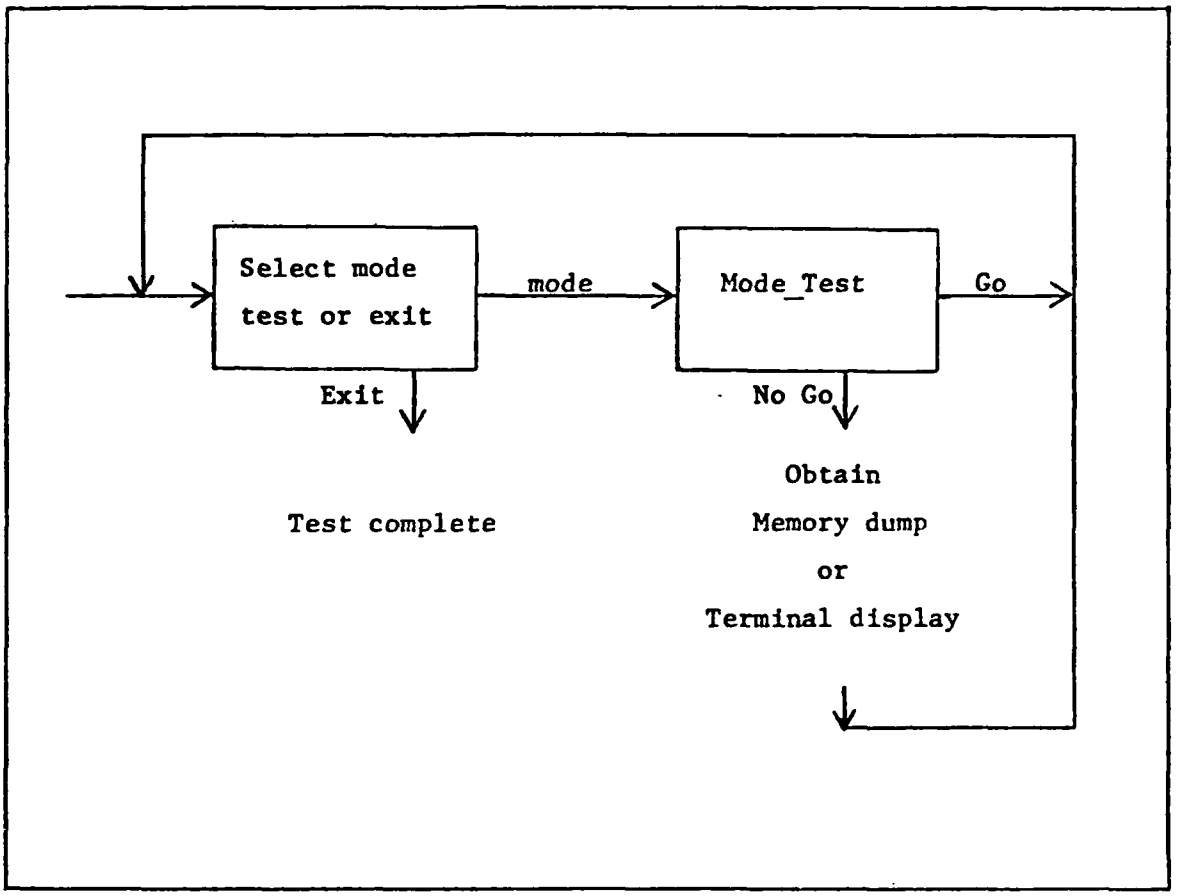


Figure II-7. Mode_Test_II Diagram.

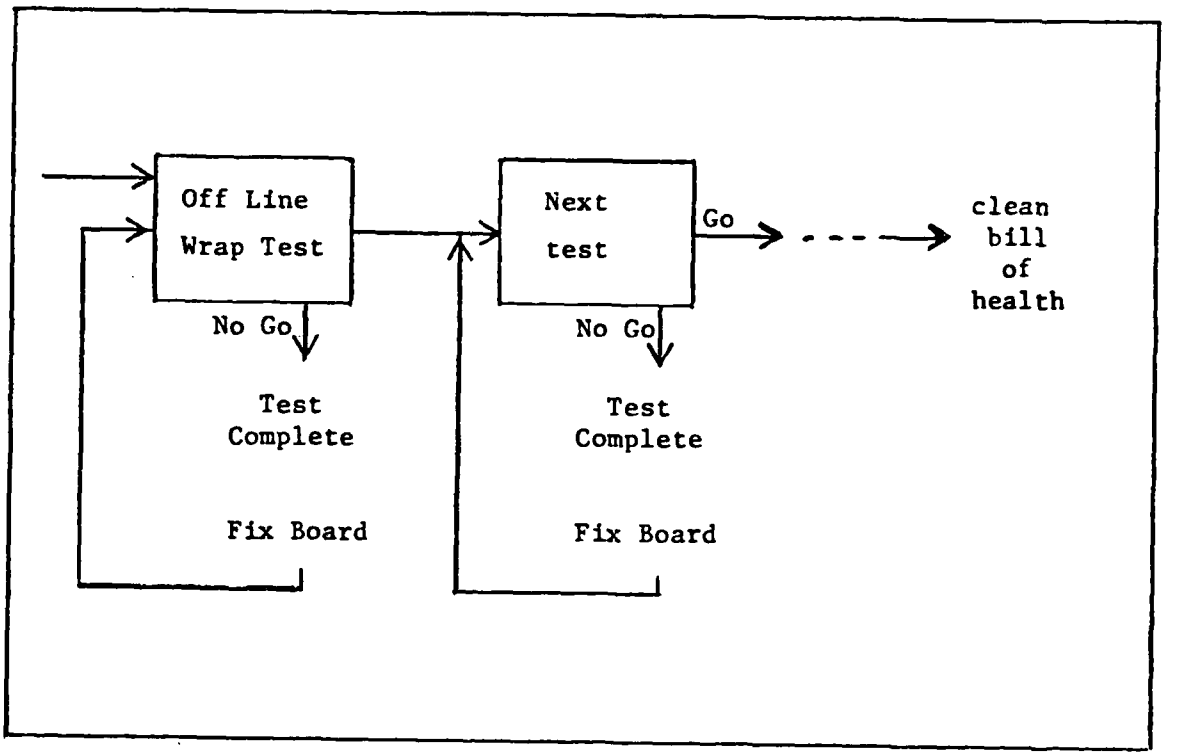


Figure II-8. Off_Line Tests Diagram.

to be accomplished by a knowledgeable technician. This is a long time and it is hoped that the off-line tests will have to be accomplished only as a last resort when the on-line tests determine a malfunction not correctable by an operator.

General Comments

Due to time constraints, the preliminary requirements and specifications aspect (chapter three) of this project addresses only Mode-Test-I. The detailed design of the Mode-Test-I subsystem and the other subsystems are left for suggested progressive follow-on thesis projects as upgraded GPBIUs are developed.

The system will be highly flexible and maintainable and adaptable to changing technologies and requirements as upgraded GPBIUs are developed.

III. Preliminary Subsystems Requirements and Specifications

Introduction

This chapter expands the Mode-Test-I subsystem by discussing what logical tests are to be performed in the four modes of operation (BC, SART, MART, BM) being addressed (see Figures II-1 and III-1). It may appear that many of the modules in the subsystems diagrams show up repetitiously. This is necessary to comprehensively show the tests necessary for each mode of operation. In actual implementation the same coded module may be called from several places to perform its function; however, the starting values (addresses of command blocks and receiving areas) are going to be different in each test and for each configuration. Therefore, each test is actually a different test in that each will test a different mode of operation of a GPBIU as well as different physical paths on the data busses and different sections of microcode located on a GPBIU.

This preliminary specification and design was arrived at through iteratively breaking down subsystem components to just a single function. As in the previous chapter, hierarchy charts and schematic diagrams are used to enhance and add clarity to the textual material. Refinement of this preliminary design stage will come during the detailed design stage in a follow-on thesis effort.

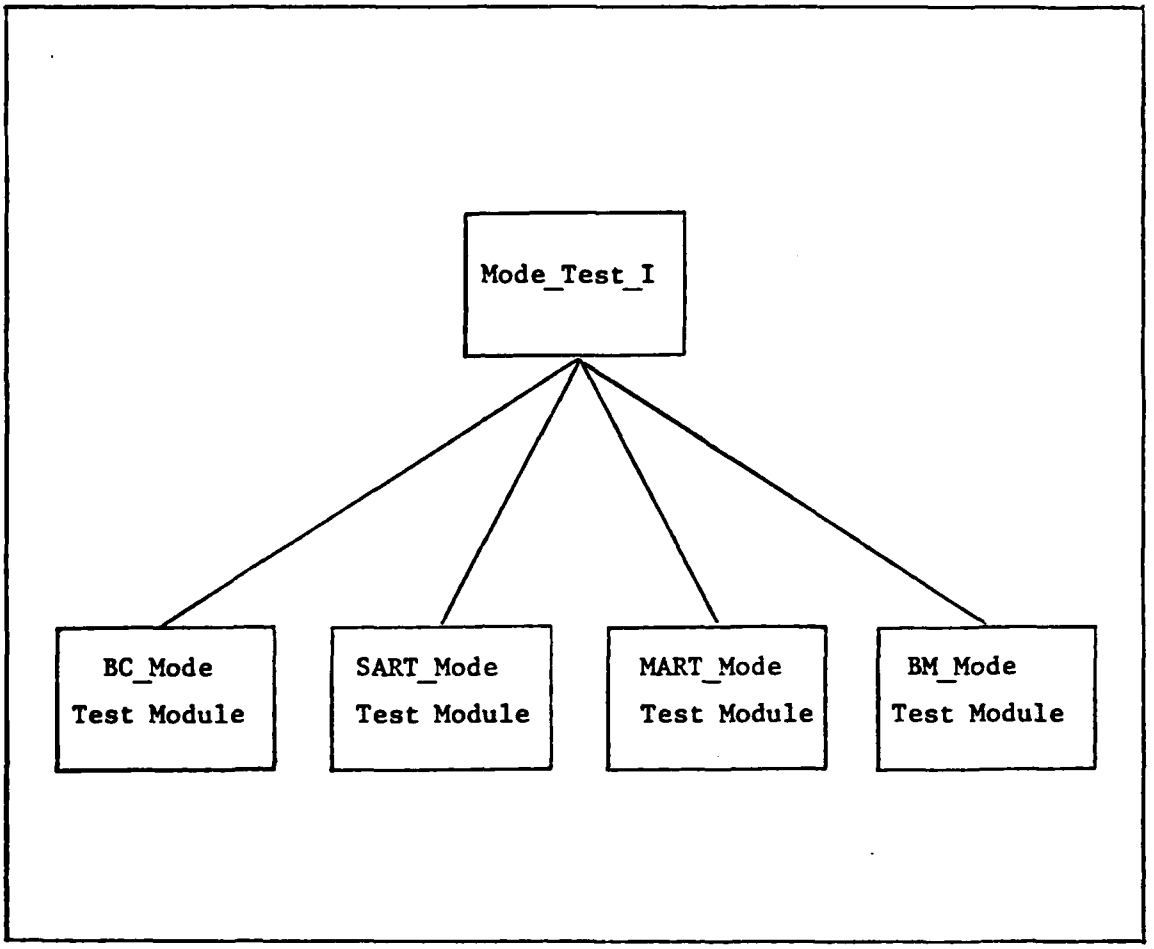


Figure III-1. Mode_Test_I Subsystem Diagram.

Mode-Test-I

Mode-Test-I consists of testing a GPBIU in a selected mode (BC, SART, MART, or BM). Each mode is represented by a different module (see Figure III-1). The requirements for each of these modules follow.

Bus Controller (BC) Mode Test Module

As stated in chapter 1, the primary function of a bus controller is communication control of data bus traffic. We can add to this the statement that a bus controller may encounter various configurations along the data bus and therefore must know the current configuration to maintain communication control. In the design of testing a GPBIU in the BC mode, it will be an operator's responsibilty for insuring the appropriate hardware configuration is accomplished. The design of the BC-Mode Test Module will allow the operator to do this by selecting a test configuration from a menu which is to be displayed when first entering the BC-Mode Test Module (see Figure III-2 and chapter 4). Existing software routines will be utilized as appropriate to allow an operator to change message block contents during a bus controller mode activity to accomplish bus switching and message routing (direction) across the bus.

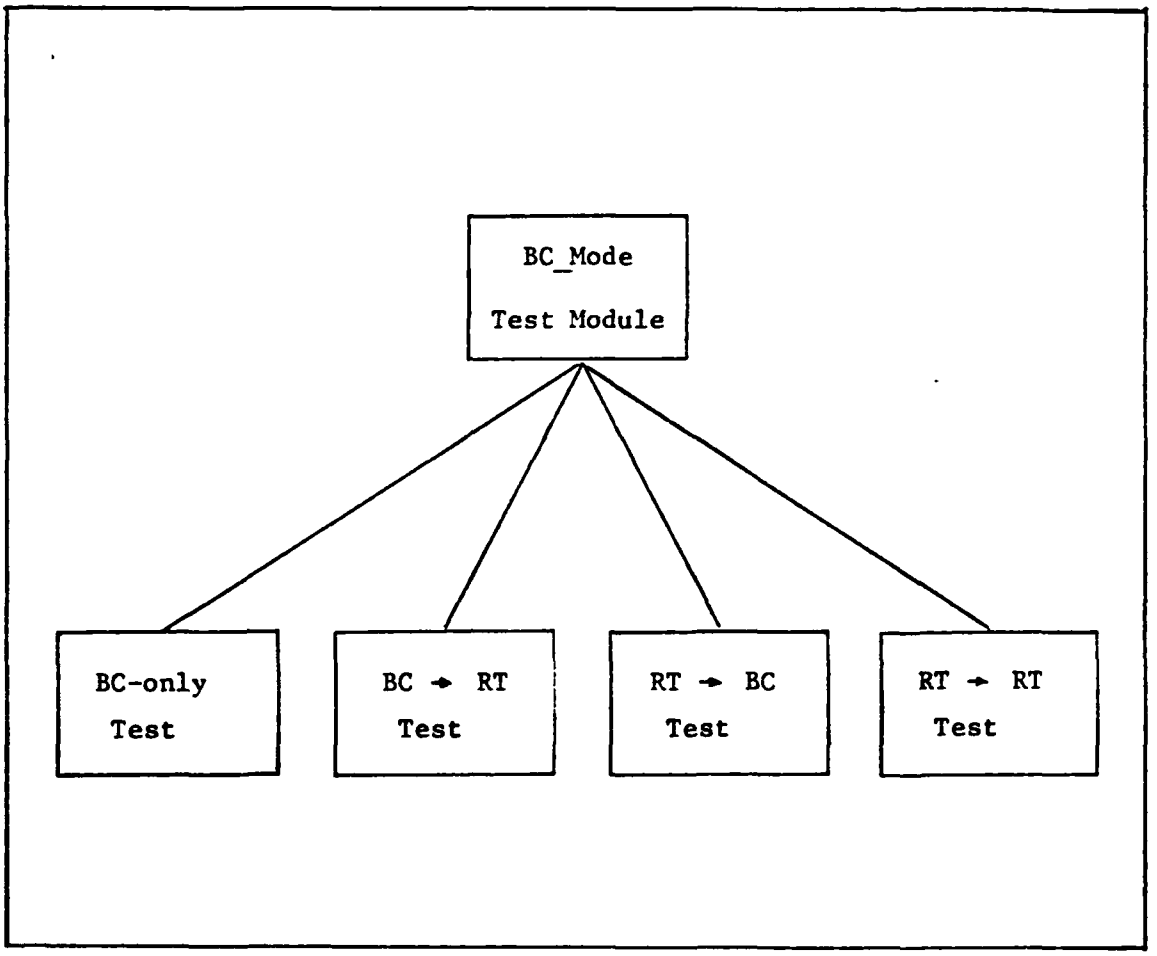


Figure III-2. BC_Mode Test Module.

Configuration Setup

The host computer for the Mode-Test-I shall be configured as represented by Figure III-3. The two GPBIUs will be connected to the host system's UNIBUS and also to each other. Connection of the two GPBIUs will be via a MIL-STD-1553 multiplex data bus. The GPBIUs will utilize either the host's memory or a separate area of UNIBUS memory (see Figure I-1).

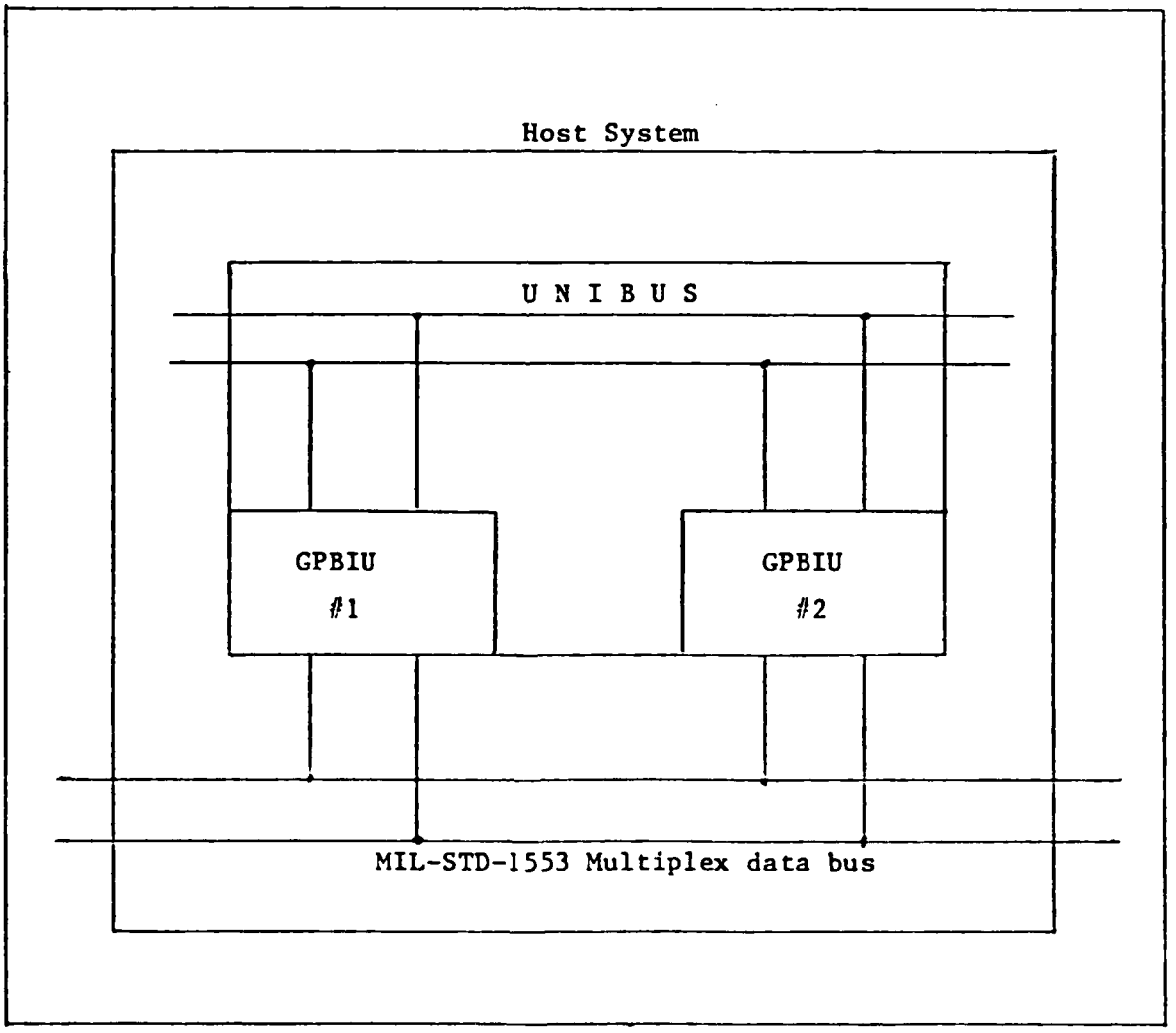


Figure III-3. Schematic Configuration for Two GPBIUs on a Single Host.

Figures III-4 thru III-7 show the four data transfer tests which may be selected from the BC-Mode Test Module. Figures III-8 thru III-10 show the physical connections for the modes. Once a physical configuration is selected, the operator will establish the logical configuration utilizing as much as possible the existing support routines for initializing memory to the desired starting addresses, loading data into these areas and issuing an appropriate command word to the GPBIU to start the GPBIU executing message transfers or to accomplish the on-line internal wrap test (see Figure III-4). (NOTE: Because an internal wrap test can test a large amount (a technician states 80 - 90 percent) of the GPBIU circuitry, it is recommended it be the first test run following host system validation.) The internal wrap test will only be run in bus controller (BC) mode under the BC-only test (discussed below). The test shall be run on both installed GPBIUs. A memory dump or optional display will be provided whenever a malfunction is detected.

A brief explanation of each data transfer configuration will show why these configurations are important and why they were chosen to be included in the bus controller mode design.

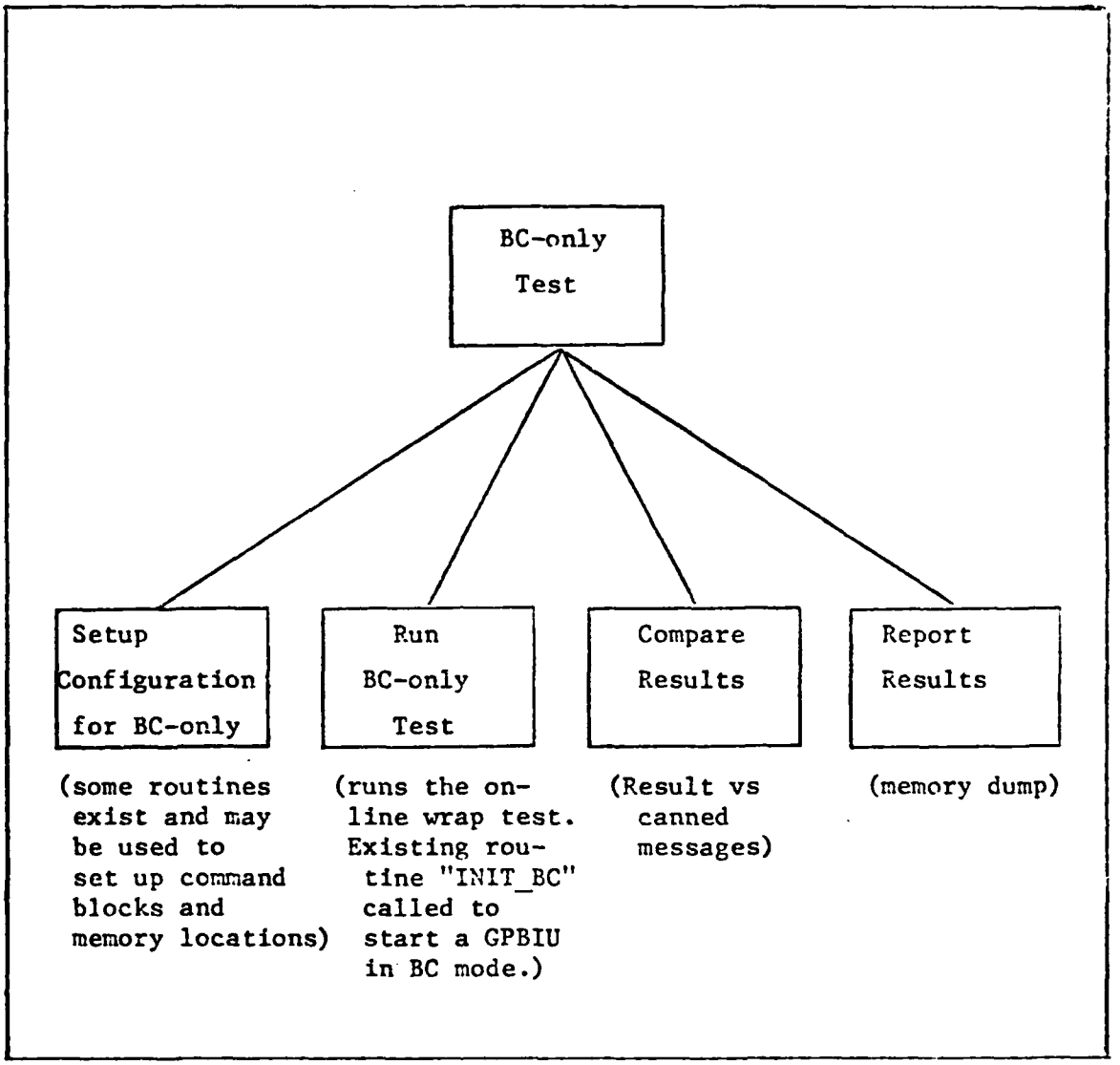


Figure III-4. BC-only Test.

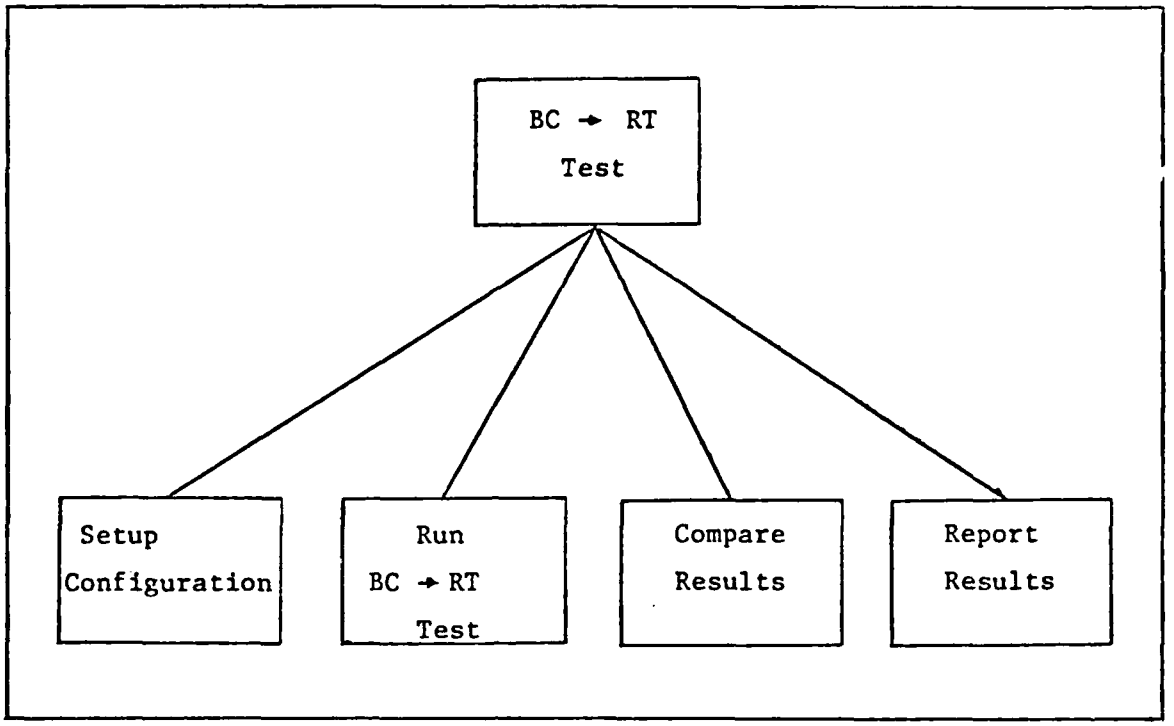


Figure III-5. BC → RT Test.

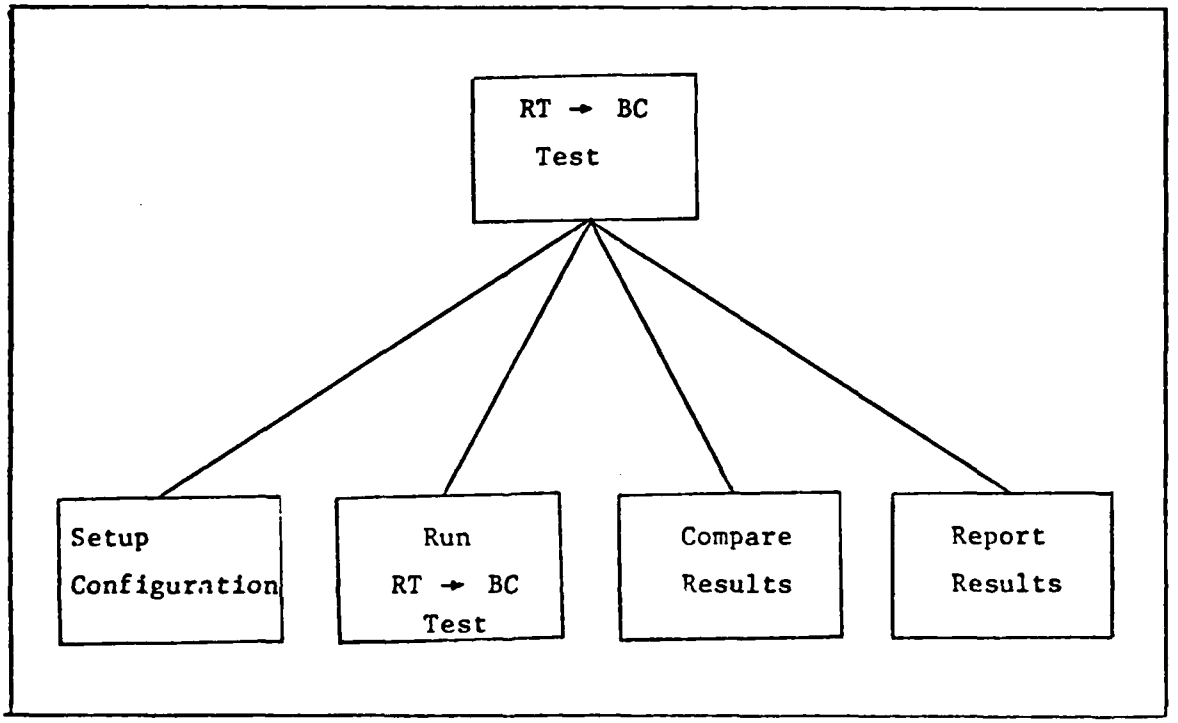


Figure III-6. RT → BC Test.

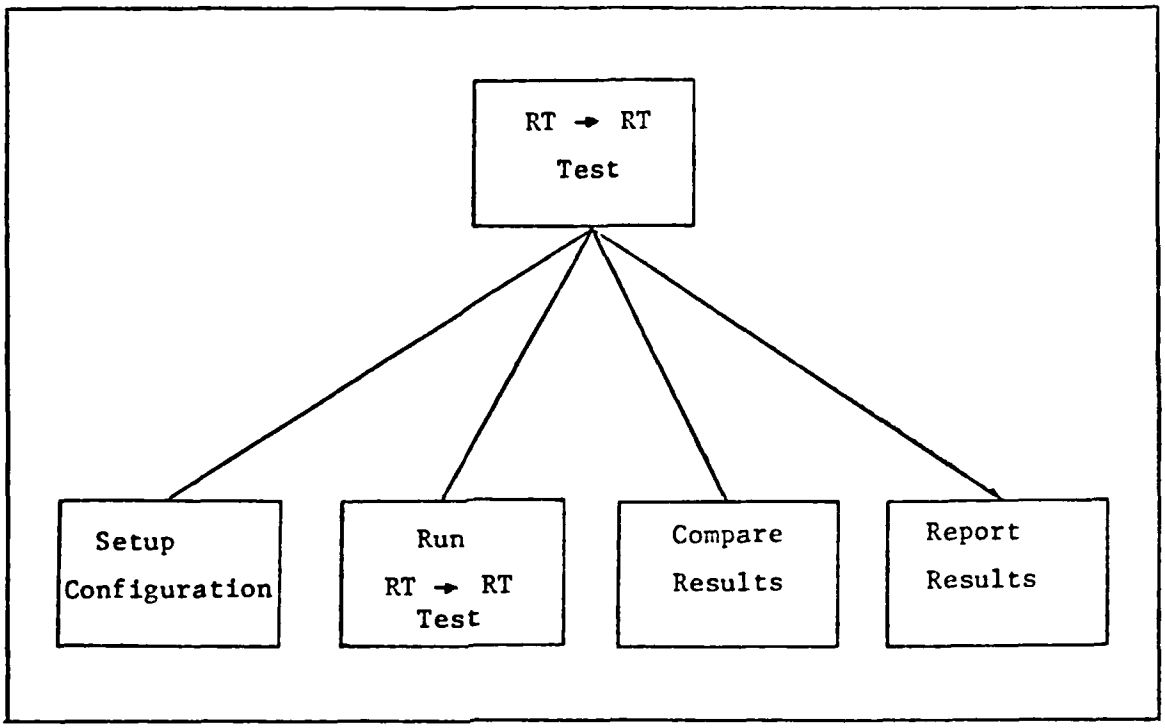


Figure III-7. RT → RT Test.

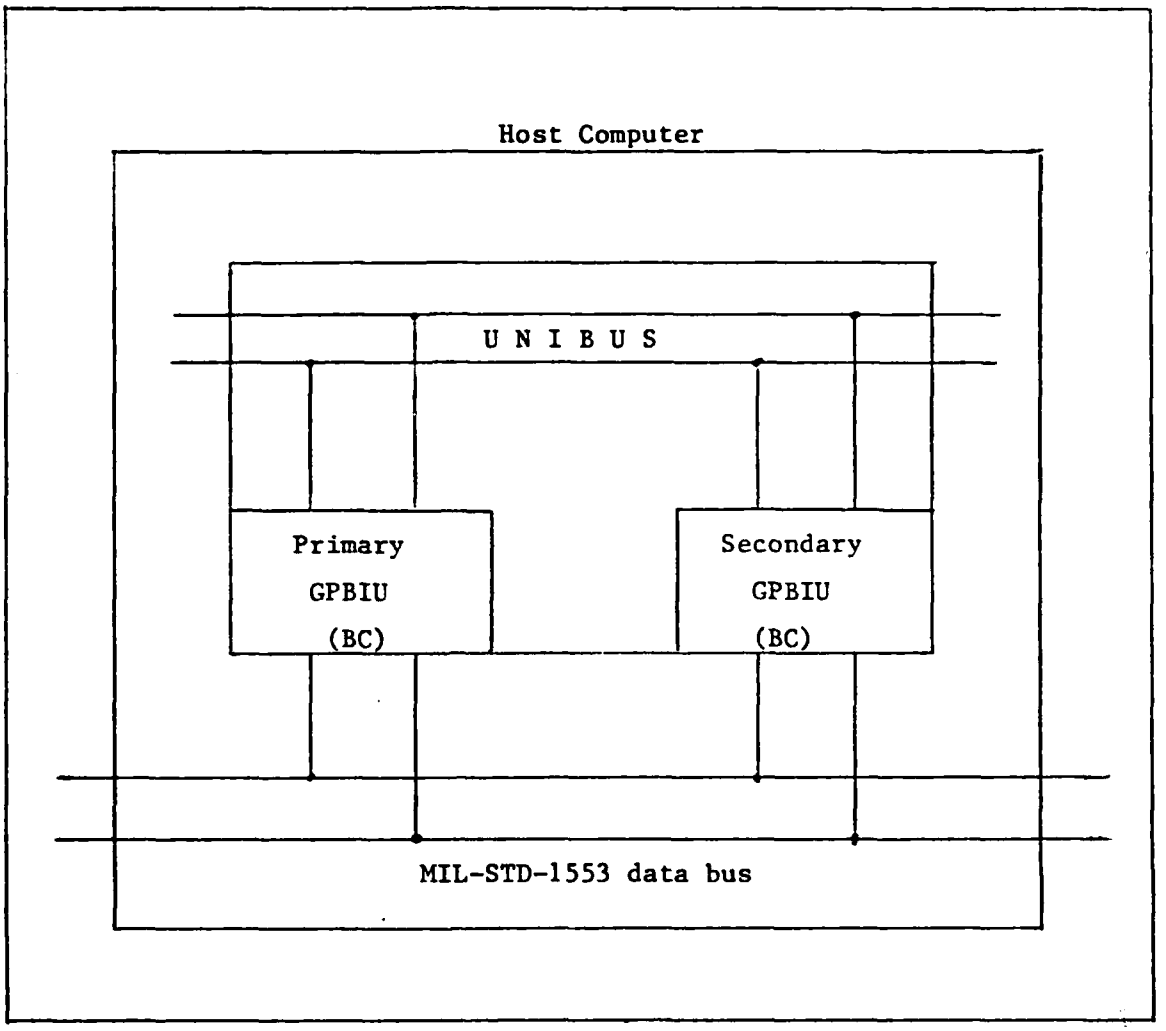


Figure III-8. Bus Controller (BC) Configuration.

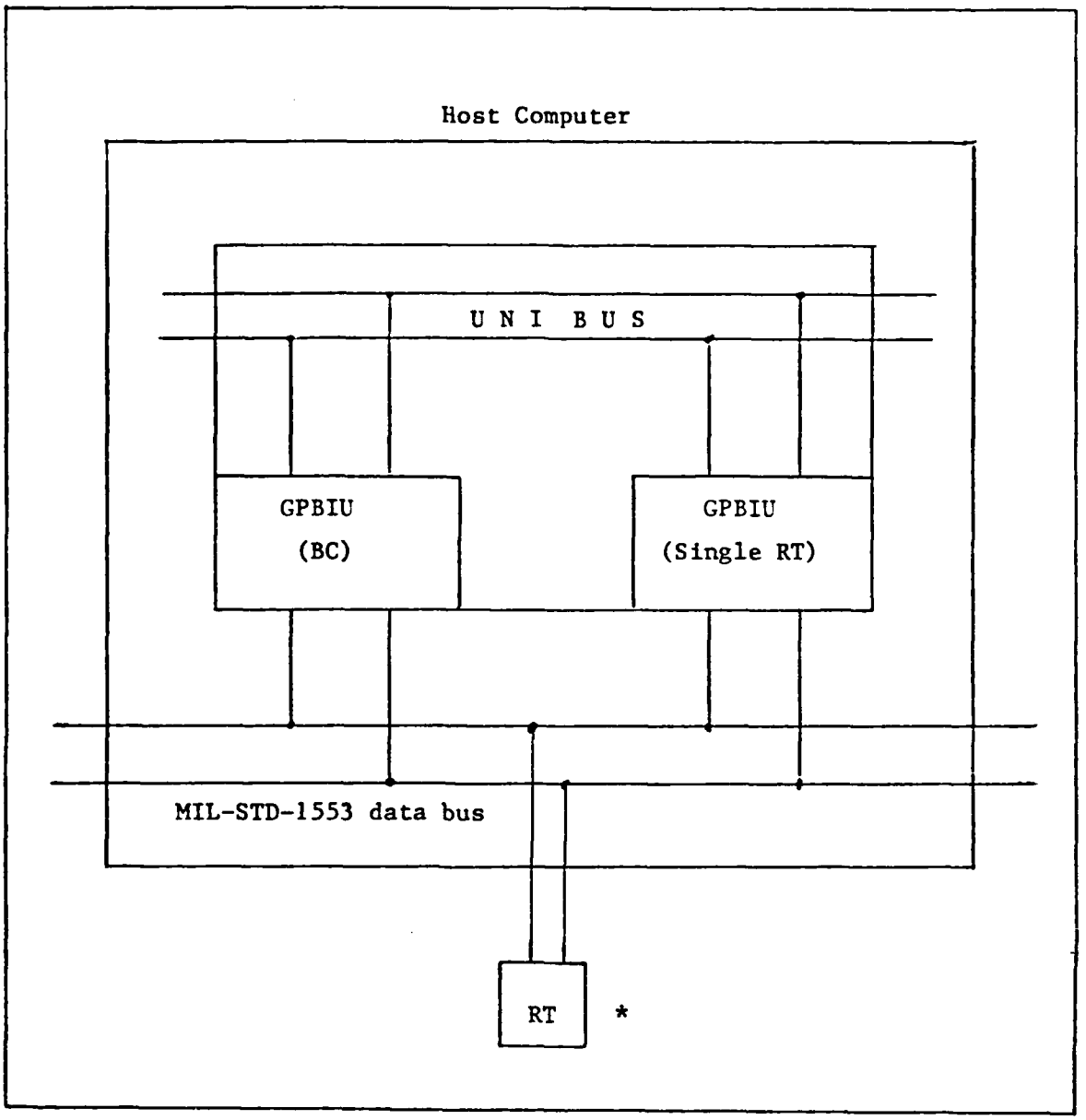


Figure III-9. Bus Controller (BC) to Remote Terminal (RT) or RT to BC Configuration.

* Alternate configuration using available remote terminal (RT)

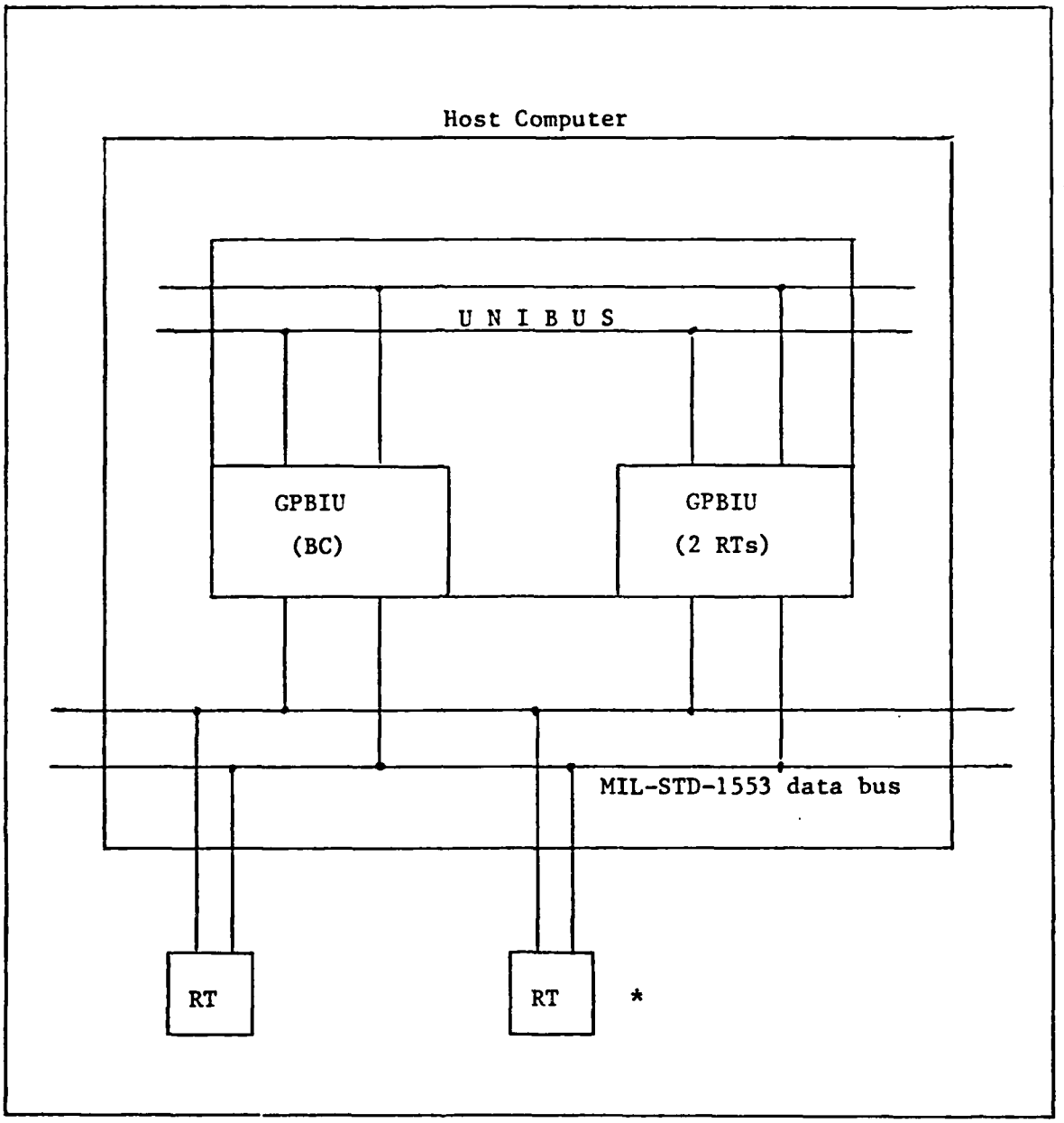


Figure III-10. Remote Terminal (RT) to Remote Terminal (RT) Configuration.

* Alternate configuration using available remote terminals

Bus Controller Only (BC-only) Test

This function allows the wrap test to be run (see Figures III-4 and III-8). Currently, an on-line wrap test doesn't exist. An off-line wrap test does exist on a PDP 11/55 along with other tests and a technician states that approximately 80 - 90 per cent of the GPBIU circuitry can be tested using these tests. If possible, the current off-line wrap test may be modified to be used on-line on the VAX 11/780. If not feasible, a new test may have to be accomplished.

On-Line Internal Wrap Test

Following host system validation, an on-line internal wrap test would be an excellent starting test for an operator to run following a suspected malfunctioning of a GPBIU. The reason is because, as stated above, it would test a large percentage of the GPBIU circuitry. Running the wrap test on-line is desired so that the host computer need not be brought down to remove the GPBIU off-line to accomplish the test. If the on-line wrap test identifies failures of the GPBIU, other tests as discussed in chapter three may be accomplished. If further tests produce other than positive results, then the GPBIU may be taken off-line for other testing. It is hoped this will be the last resort since bringing down the host computer is a significant inconvenience to other users.

Accomplishing an on-line wrap test would also help to verify logical and electrical GPBIU specifications thereby increasing user confidence that the GPBIU is functioning properly. If the data sent through the GPBIU is returned and stored and matches the canned message blocks sent, an operator will assume the GPBIU is functioning logically and electrically correct for the on-line wrap test.

Bus Controller to Remote Terminal (BC->RT) Test
(see Figures III-5 and III-9)

This type of data transfer may be used to get data from a central computer to a sensor or avionic subsystem. MIL-STD-1553 Multiplex Applications Handbook states:

> "Typically these types of data transfers are again related to the role of the processor, which has the bus control function. A mission computer may have the requirement to be the data source to sensors, providing such data as position update to an INS, or the requirement to transmit display parameters to a graphics generator. The mission computer oftens serves as the processor to effect weapons system control, such as fire control, in which case it is controlling both the multiplex system and computing parameters for target designation, weapon initialization, etc. In this case, controller-to-remote terminal transfers are data transfers from the fire control computer to the remote terminals, which contain the interfaces to target designators, stores, etc."

A second GPBIU can and may be used as the remote terminal since it has been stated a minimum of two GPBIUs are to be installed during testing.

Remote Terminal to Bus Controller (RT->BC) Test
(see Figures III-6 and III-9)

MIL-STD-1553 Multiplex Applications Handbook states:

> "This type of data transfer is used to get data to the bus controller. The bus controller is usually a mission computer, fire control computer, navigation computer, etc. As such it requires data from several sensors such as air data, internal navigation system or inertial measurement unit, radio navigation, etc., to perform its assigned sensor computational functions. Therefore, its (the mission computer) assignment as bus controller is natural. It will initiate the requests to the remote terminals for the data that the procesing software needs. The data needs of the mission computer's assigned processing tasks establishes the requirements for data transfers to it from other sources (terminals) on the bus."

A second GPBIU can and may be used as the remote terminal since it has been stated a minimum of two GPBIUs will be installed during testing.

Remote Terminal to Remote Terminal (RT->RT) Test
(see Figures III-7 and III-10)

MIL-STD-1553 Multiplex Applications Handbook states:

> "The bus controller does not need to receive and retransmit all data even though it is in control of the bus. An important class of data transfers is the direct transfer of data from one remote terminal to another, which can be used if the processor that contains the bus controller is not involved in the processing of the data and if reformatting is not required. In avionic systems that employ more distributed processing (e.g., CADC, INS on the bus), the additional processing capability at those terminals can be used to select and format data for remote terminal-to-remote terminal data transfers."

A second GPBIU can be used to simulate the two remote

terminals (see Figure III-1Ø) and may be used to do so if two remote terminals are not available at the time of testing.

The above quotations should make it obvious why tests are needed in the stated modes and configurations.

Validate Message Traffic

Messages should be validated between the source and destination. When a GPBIU in bus controller mode sends data to a remote terminal, it is very important that what the remote terminal "sees" is what the bus controller sent. The same is true for data being sent from a remote terminal to a bus controller or between remote terminals. This module will consist of comparing memory locations in the message blocks to verify what was sent out over the bus is the same as what was received back over the bus. If any of the messages are any different, a memory dump and/or terminal display of the two message blocks (sending and receiving) will be performed.

Validation of data transfer will be accomplished for each type of data transfer identified above. Canned message blocks will be used as the sending data. Reserved memory locations will serve as the destination terminal(s) so validating messages will involve comparing those memory locations. If there is no discrepancy between corresponding words, it will be assumed the messages are being properly directed by the GPBIU.

Memory Dump (Report Results)

Following determination that a message received (stored in its "receiving" message block) differs from what was initially stored in the "sending" message block, the test module will provide a printed dump and/or terminal display of the two message blocks. These can be examined by an operator or a technician to determine in which word the difference occurred. It is recommended that the test software be written with the flexibility to allow the operator a choice of seeing only the words which differ or seeing the entire message blocks. The operator will also have the choice of displaying the results onto the terminal or having a printed copy.

Single Address Remote Terminal (SART) Mode Test Module

This module is also to be called by the GPBIU Mode-Test-I executive (see Figures III-1 and III-11). In this mode, one GPBIU is established as the bus controller and the other as a remote terminal (see Figure III-9). It is recommended each GPBIU be run in each role. At first glance, SART mode may appear to be only a subset of MART mode (to be discussed shortly), and to some extent it is. However, different microcode on a GPBIU is exercised in the two modes. Therefore, SART is to be treated as a separate test mode.

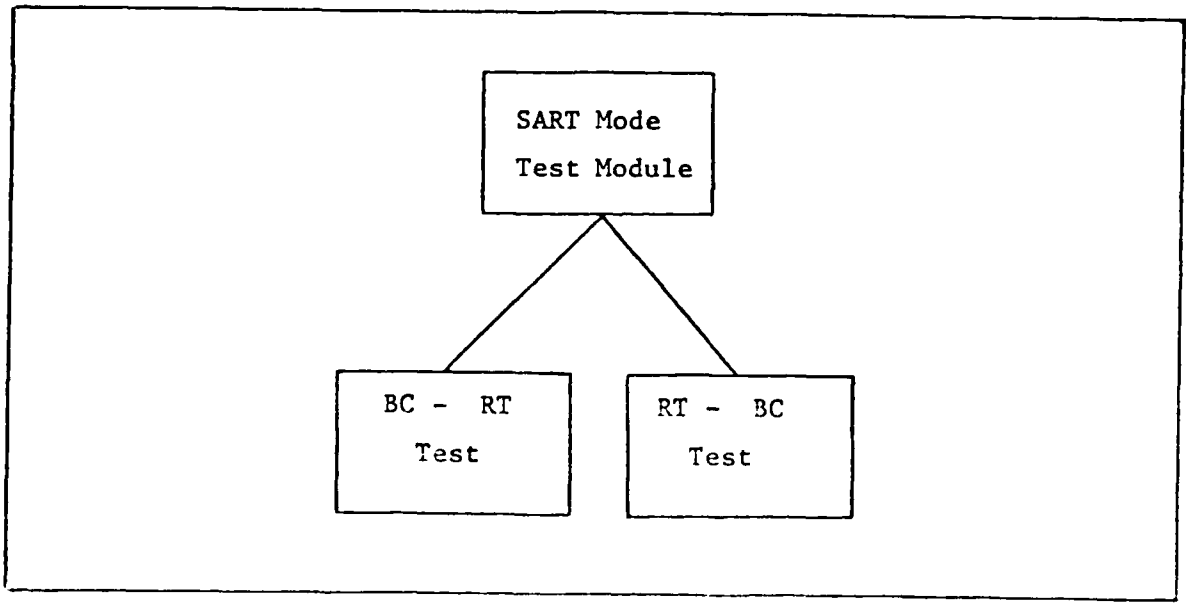


Figure III-11. Single Address Remote Terminal (SART) Mode Test.

Configuration Setup

As in the BC-only test module, the test configuration is to be selected from the Mode-Test-I menu provided to the operator. The data transfers to be accomplished in SART mode are initiated thru a command word which an operator will provide to the GPBIU. The same procedures will be followed for establishing the logical configuration as in the BC-Mode Test Module.

The types of data transfers legal in this mode are BC->RT and RT->BC (see Figures III-9, III-12 and III-13). Both types of transfers and configurations were discussed during the BC-only Mode Test Module.

Validate Message Traffic

The same guidelines apply here as with the BC-Mode Test Module discussed previously.

Memory Dump (Report Results)

The same guidelines apply here as with the BC-Mode Test Module discussed previously.

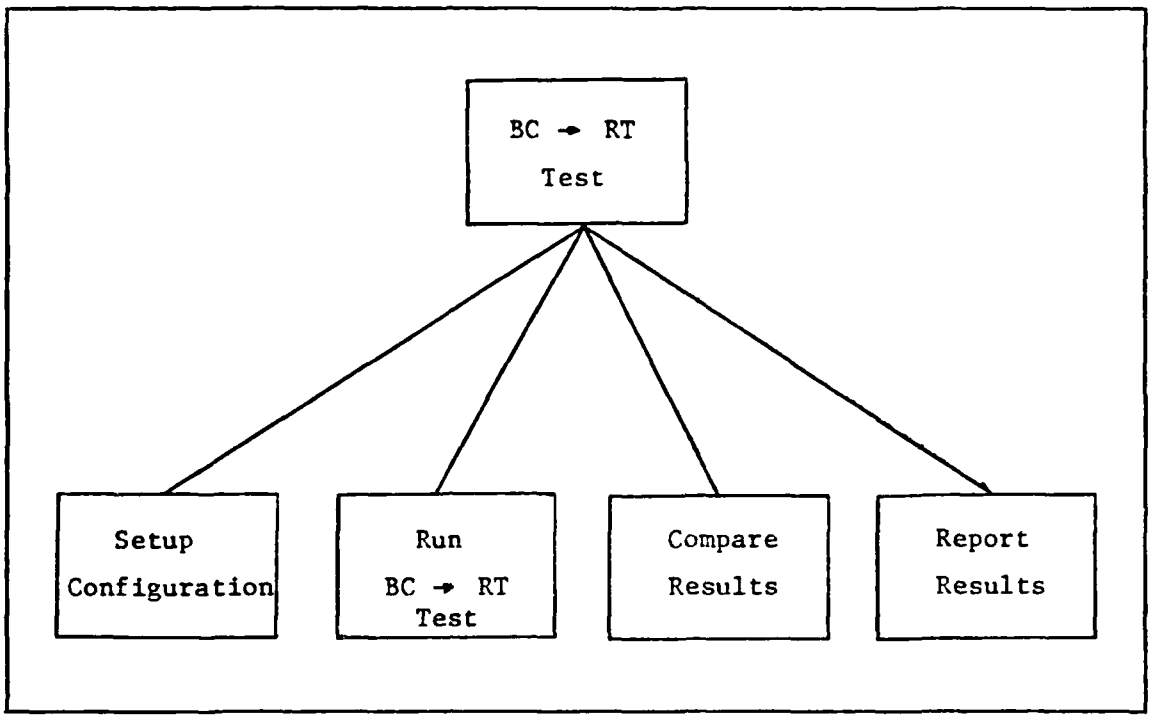


Figure III-12. BC → RT Test.

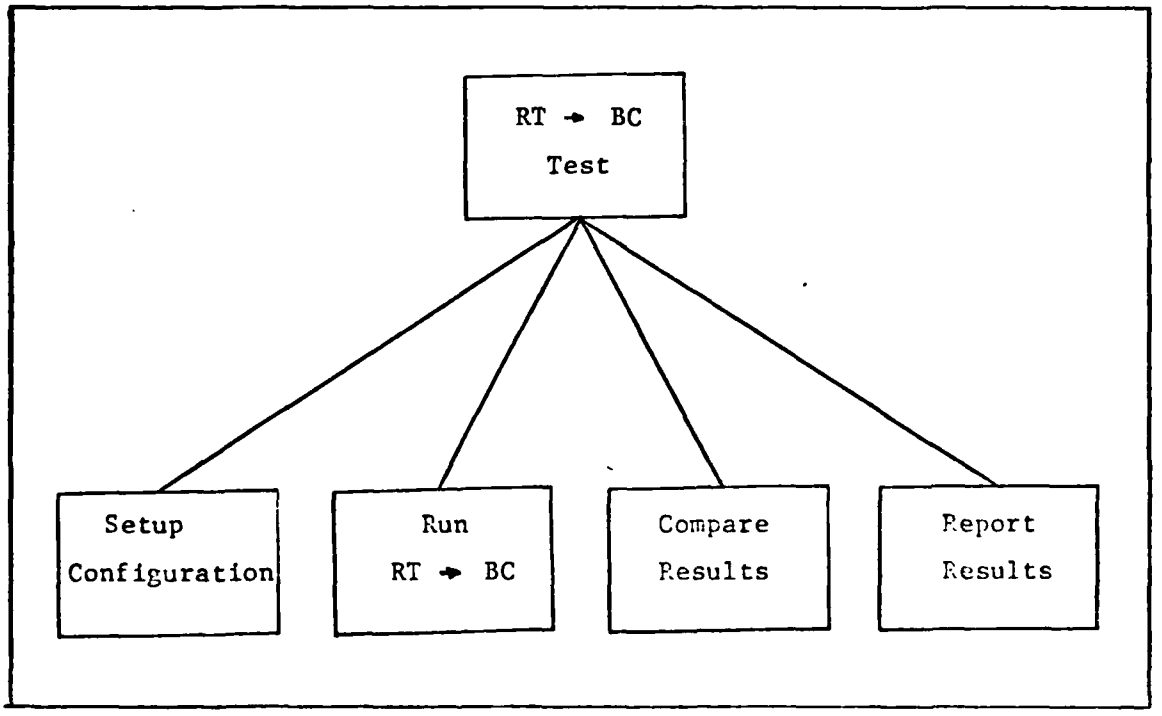


Figure III-13. RT → BC Test.

Multiple Address Remote Terminal (MART) Mode Test Module

MART mode differs from SART mode only in configuration. MART mode allows multiple remote terminals whereas SART mode allows only one remote terminal. The tests BC->RT and RT->BC will be similar but not identical for both modes. The command words and various physical configurations and microcode exercised will be different. Additionally, MART will allow for testing of message transmission/reception between RTs and testing of message reception by all RTs on the bus, i.e. broadcast (see Figures III-14 thru III-19 and Figures III-9 and III-10). NOTE: The broadcast data transfer is an option which is not currently in use in military airplane avionics (Ref 4:3-19).

Configuration Setup

The physical configuration to be accomplished prior to running tests in MART mode include those identified in Figures III-9 and III-10. Additionally, MART mode allows messages to be sent to all RTs on the MIL-STD-1553 data bus, i.e. broadcast (see Figures III-18 and III-19). The operator will have the flexibility to have as many RTs on the data bus as is desired.

The logical configurations such as initializing memory addresses, loading those areas with message blocks and issuing appropriate command words are to be accomplished

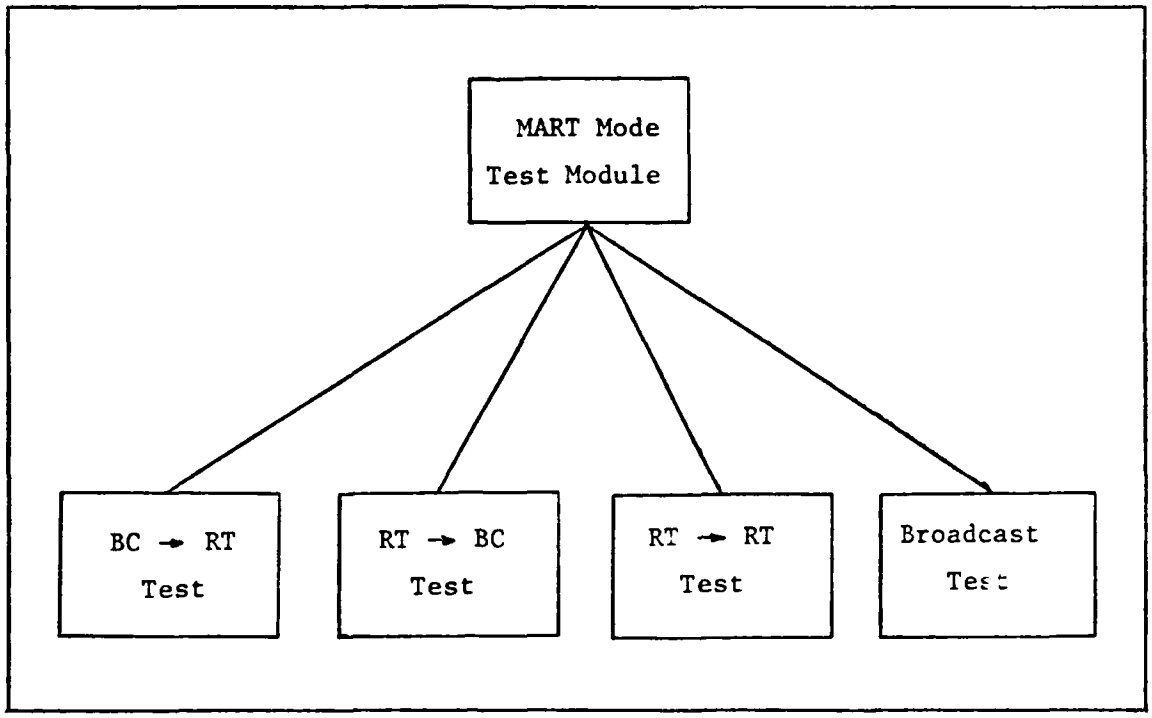


Figure III-14. Multiple Address Remote Terminal (MART) Mode Test.

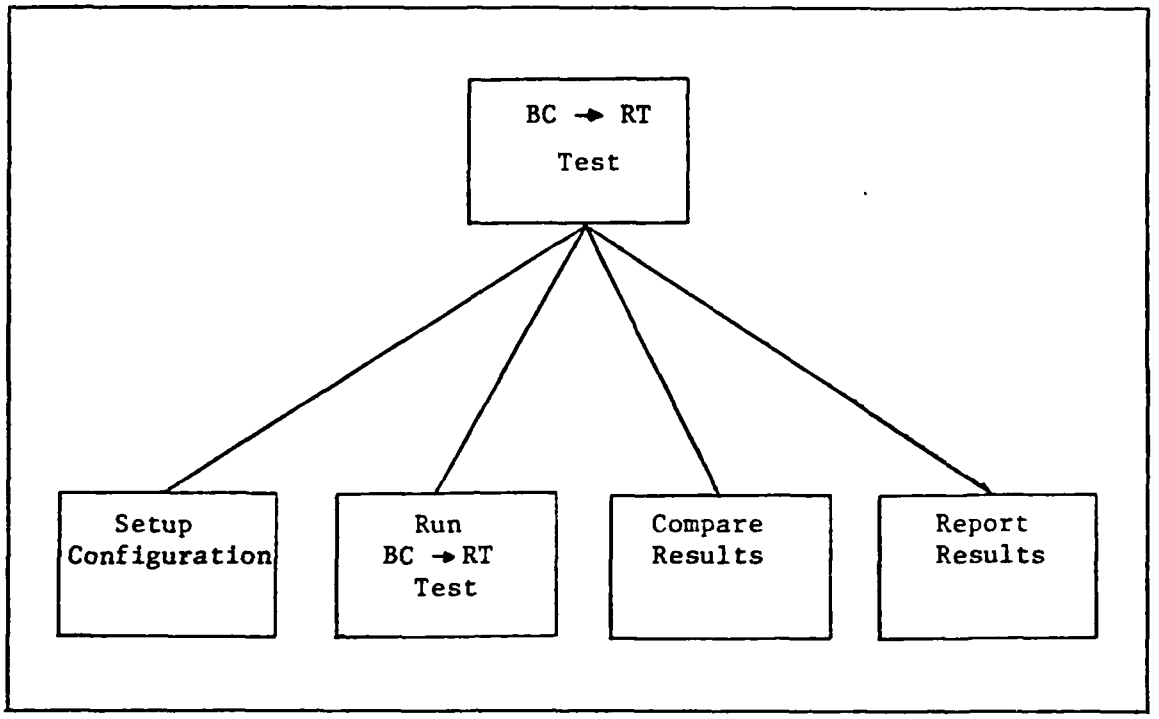


Figure III-15. BC → RT Test.

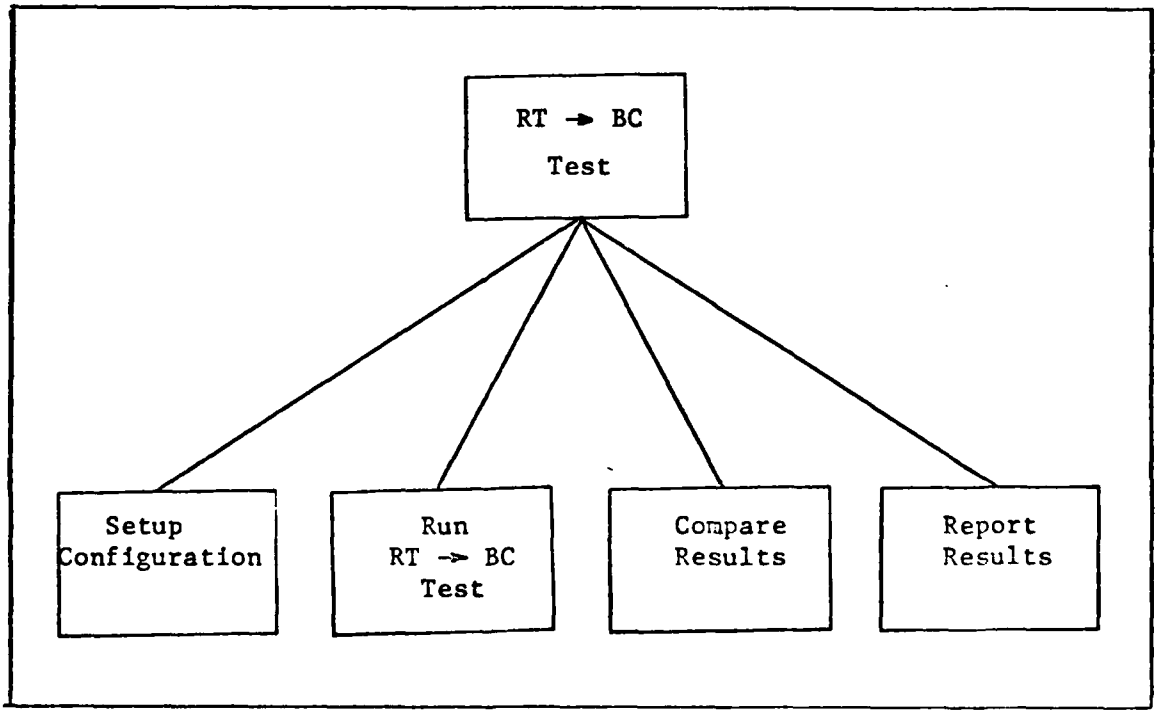


Figure III-16. RT → BC Test.

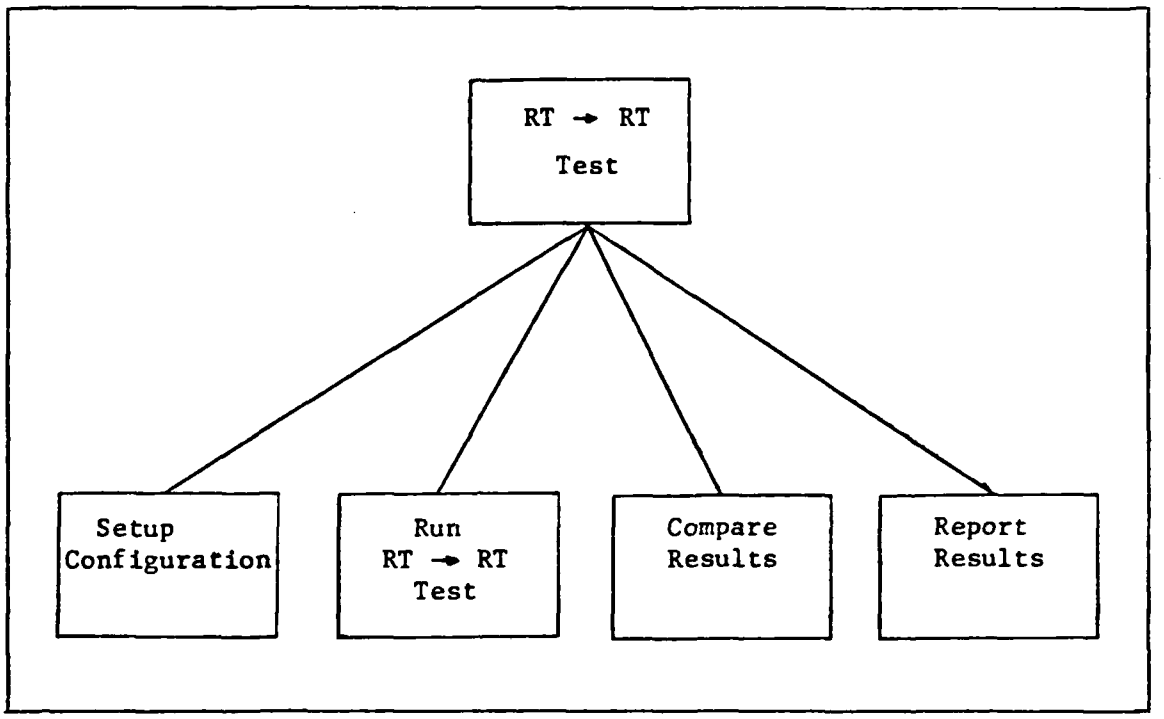


Figure III-17. RT → RT Test.

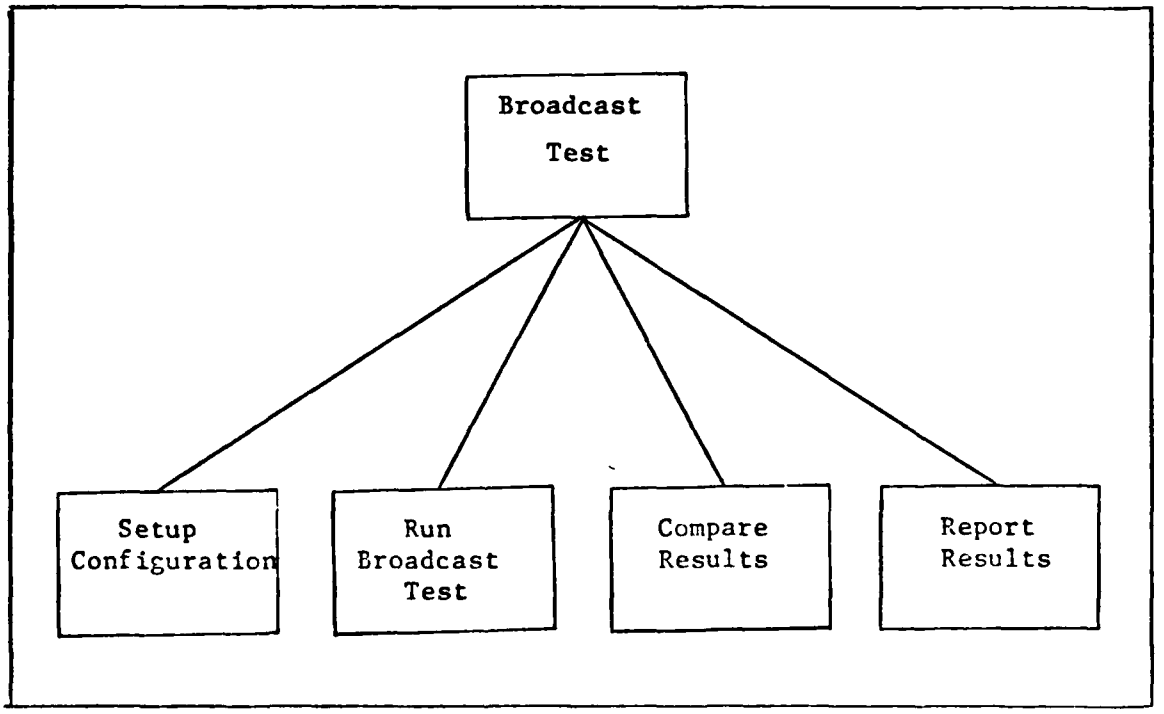


Figure III-18. Broadcast Test.

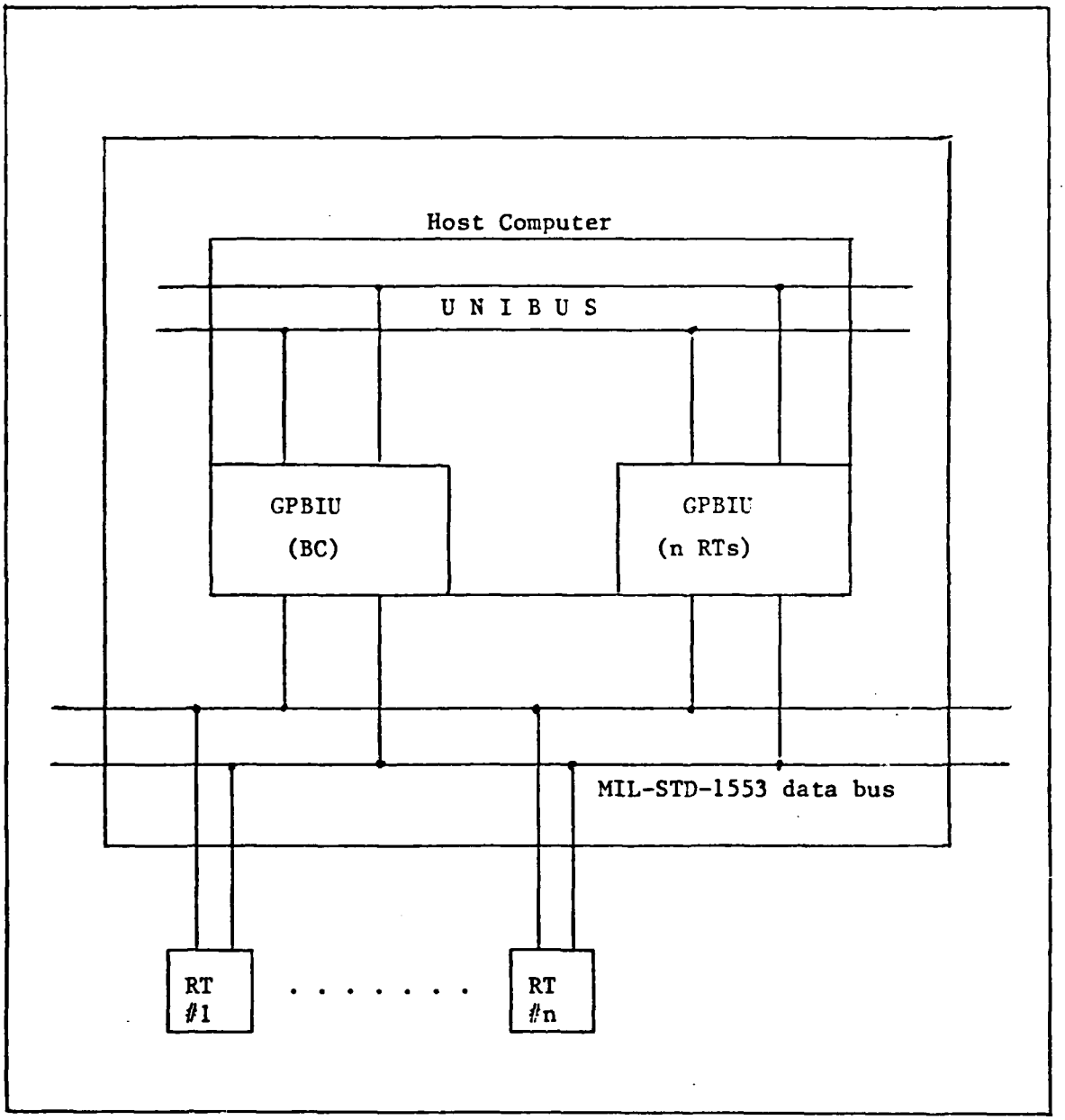


Figure III-19. Broadcast Configuration.

utilizing existing support routines as appropriate. As with the BC-Mode Test Module the operator will have the responsibility and flexibility to introduce the starting values.

Validate Message Traffic

The same guidelines apply here as with the BC-Mode Test Module discussed previously.

Memory Dump (Report Results)

The same guidelines apply here as with the BC-Mode Test Module discussed previously.

Bus Monitor (BM) Mode Test Module

As stated in chapter one, a GPBIU in bus monitor (BM) mode allows that GPBIU to passively monitor traffic on a MIL-STD-1553 data bus. Testing of a GPBIU in this mode is necessary to insure the messages being sent on the bus are the same as those passing thru (being monitored by) the GPBIU in BM mode. If the extension allowing monitoring of the data bus for activity is incorporated in the BM mode, it is very important that what the GPBIU "thinks it sees" is actually what is being transferred across the bus. The reasoning is if the GPBIU is not correctly monitoring the message traffic, it will not be able to correctly activate the backup bus controller in the event of inactivity or when a bus controller retains control of the bus indefinitely (Ref 4:3.37). Testing a GPBIU in BM modes means having various end-to-end tests (see Figures III-20 thru III-22).

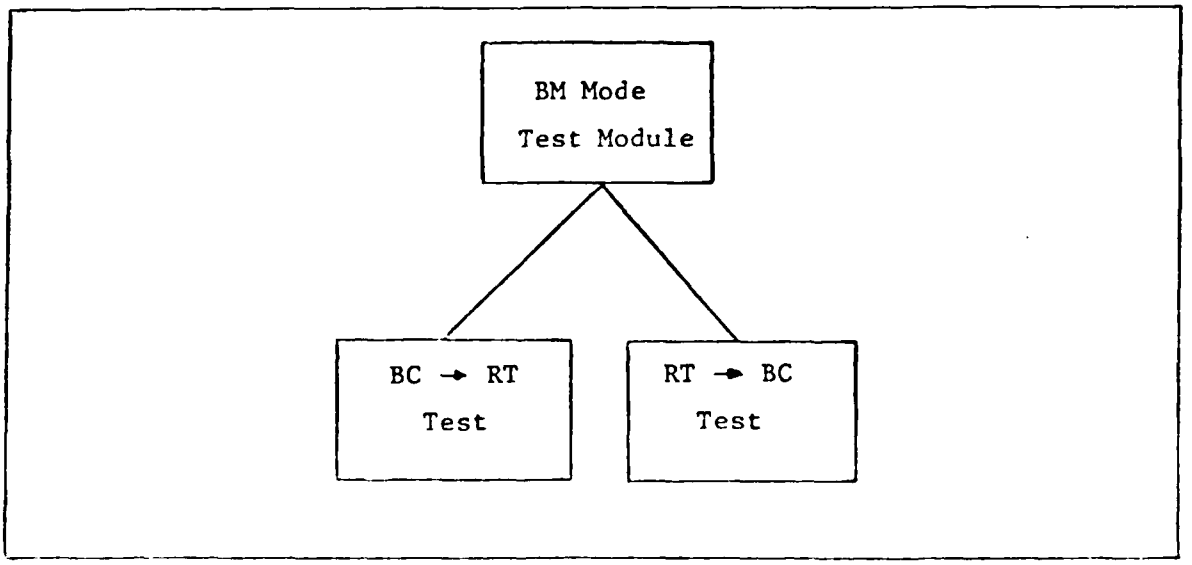


Figure III-20. Bus Monitor (BM) Mode Test.

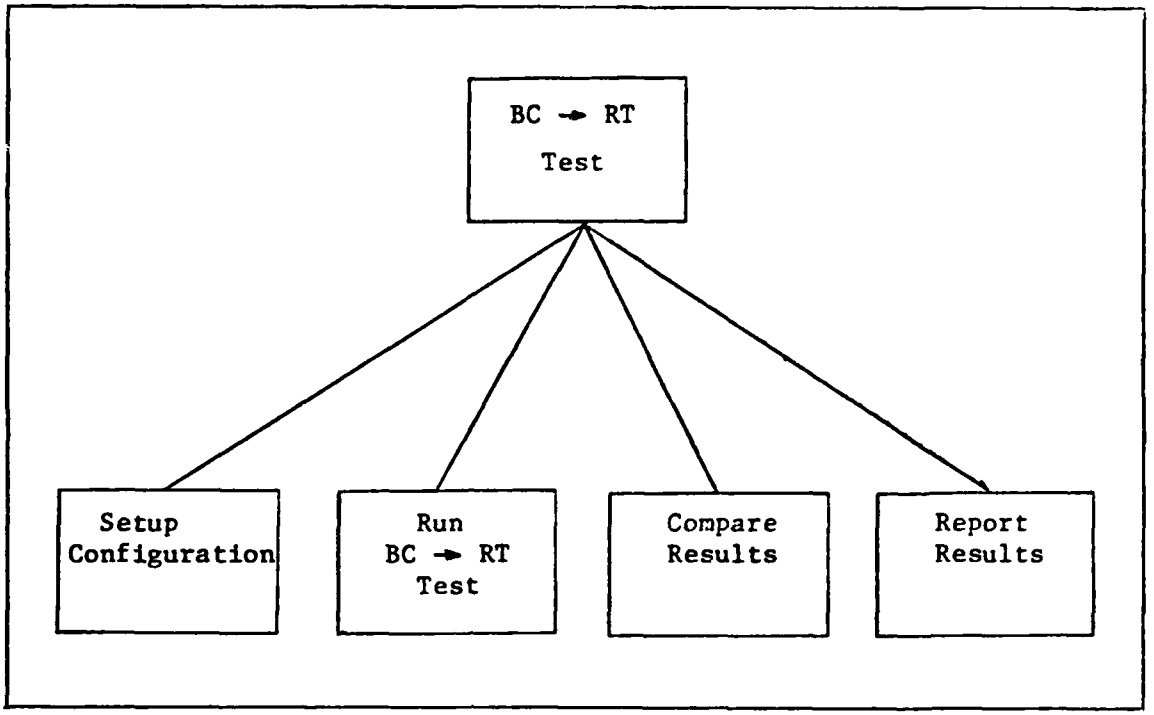


Figure III-21. BC → RT Test.

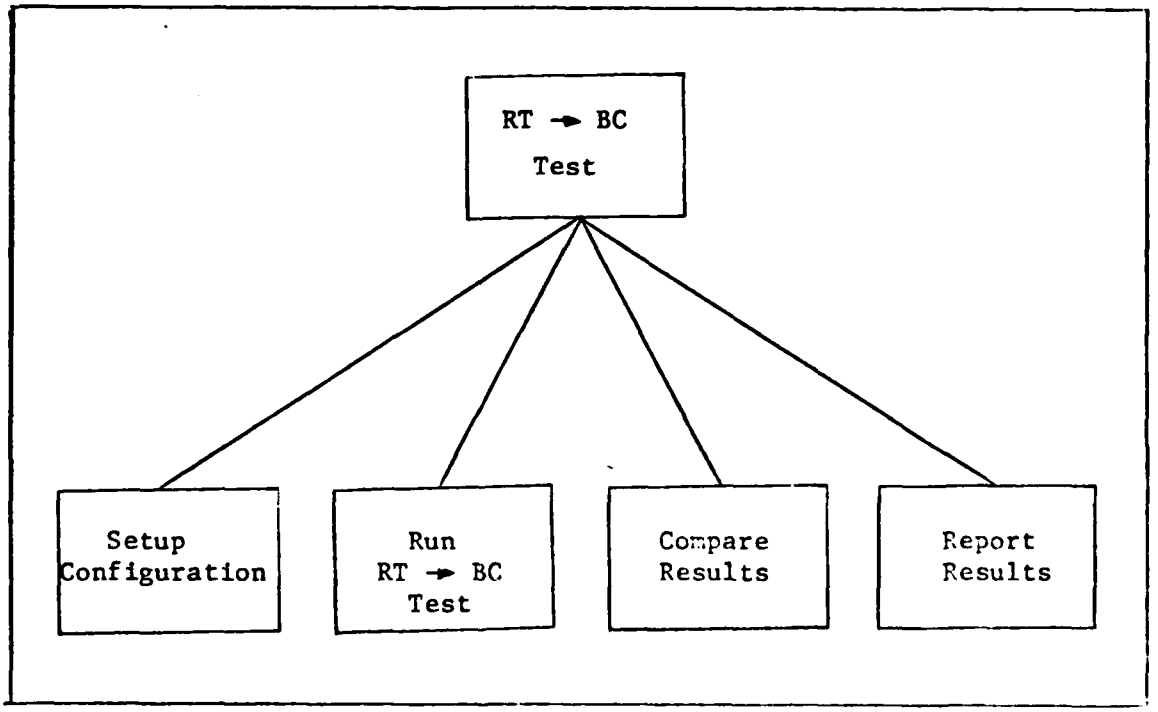


Figure III-22. RT → BC Test.

Configuration Setup

The physical configuration to be accomplished prior to running tests in Bus Monitor mode include those identified in Figures III-9, III-10 or III-19 such that one GPBIU is in BC mode and the other is in BM mode. NOTE: It is necessary to have a separate RT (or another GPBIU) on the MIL-STD-1553 data bus because one GPBIU cannot be used as both an RT and a Bus Monitor when one other GPBIU acts as Bus Controller and they are the only two GPBIUs on the system.

The logical test configurations such as initializing memory addresses, loading those areas with message blocks and issuing appropriate command words are to be accomplished utilizing existing support routines as available. As with the BC-Mode Test Module, the operator will have the responsibility and flexibility to introduce the starting values.

Validate Message Traffic

The same procedures for validating message traffic will be used in the BM-Mode Test as was used in the other tests discussed previously.

Memory Dump (Report results)

The same procedures for obtaining a memory dump or a display at the terminal will be used in the BM-Mode Test as

was used in the other tests discussed previously.

General Comments

Chapter three has identified many logical tests needed to test a GPBIU in various modes of operation on a single host computer. Many schematics of physical configurations and subsystem hierarchy charts were provided for clarity. Although these tests cannot guarantee 100 per cent coverage of the GPBIU circuitry and microcode, they will test a majority of each. Increasing the duration of each test run and utilizing "controlled randomness" to select bit patterns will improve the percentage of circuitry and microcode exercised.

The following chapter proposes a sample design in the form of flowcharts for selected modules in the BC-Mode Test Module.

IV. Sample Design

Introduction

This chapter offers a sample design of the GPBIU-Test-Executive, the Mode-Test-I Executive, BC-Mode Test Module and the BC-Only Test Module. Graphic flowcharts and Program Design Language (PDL) listings (see Appendix C) were requested by ENASF to represent the design because of their widespread use and ease of understanding. The designs of the modules were kept very general and code independent as is the recommended and accepted method of doing designs. This will allow ENASF personnel and future thesis students to easily adapt the proposed designs to their particular design and coding methods. The test procedures themselves are left for further research.

Support for Design

The design proposal is based on stated user requirements for high level preliminary specifications and design. Current ENASF operational support software routines may be incorporated during the detailed design phase of the overall test system. For example, routines for initializing memory areas and establishing the addresses of the command blocks and entering the command word prior to starting a GPBIU will be incorporated into the detailed design where appropriate (see Reference 1 and Appendix A for some existing routines). The proposed high

level design will meet the objective and scope as stated in chapter one as well as the functional requirements for a GPBIU on a single host as discussed in chapter two.

Software Requirements

The main software requirement identified in the initial development of the support software for exercising GPBIUs is that the software be source compatable among all the DEC computers at ENASF (Ref 1). ENASF requests implementation of the eventual GPBIU test system utilize high level language drivers and support routines to provide user-friendly interface on all the DEC systems (Ref 1). Program and module interfaces, parameters and error routines will need to be standardized. Therefore, because of the similarity of interfacing on all DEC systems, the FORTRAN source code is the chosen language for implementation (Ref 1). FORTRAN may be directly transferred from system to system.

Figures IV-1 thru IV-4 show the flowcharts for the selected modules.

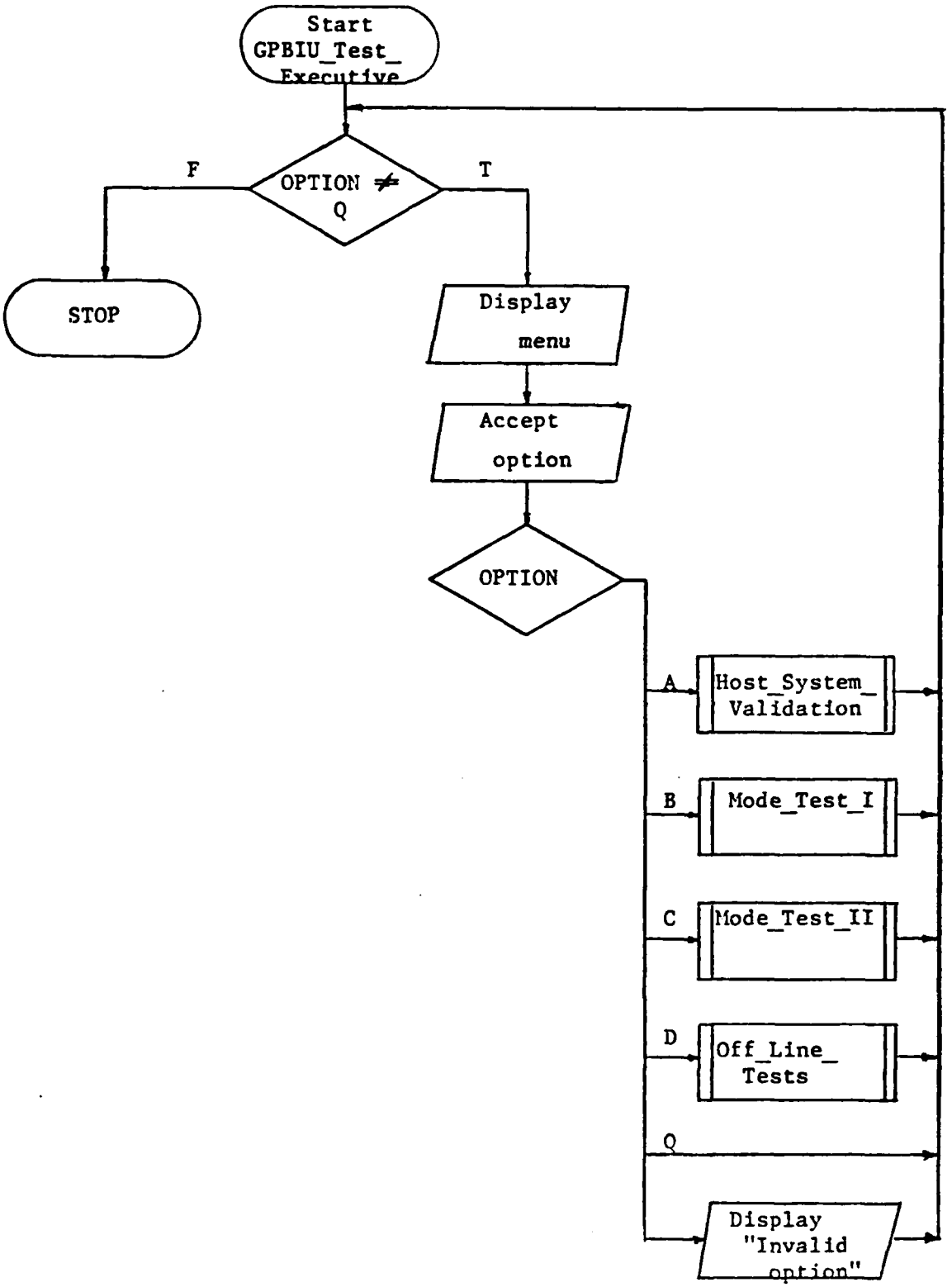


Figure IV-1. GPBIU_Test_Executive Flowchart.

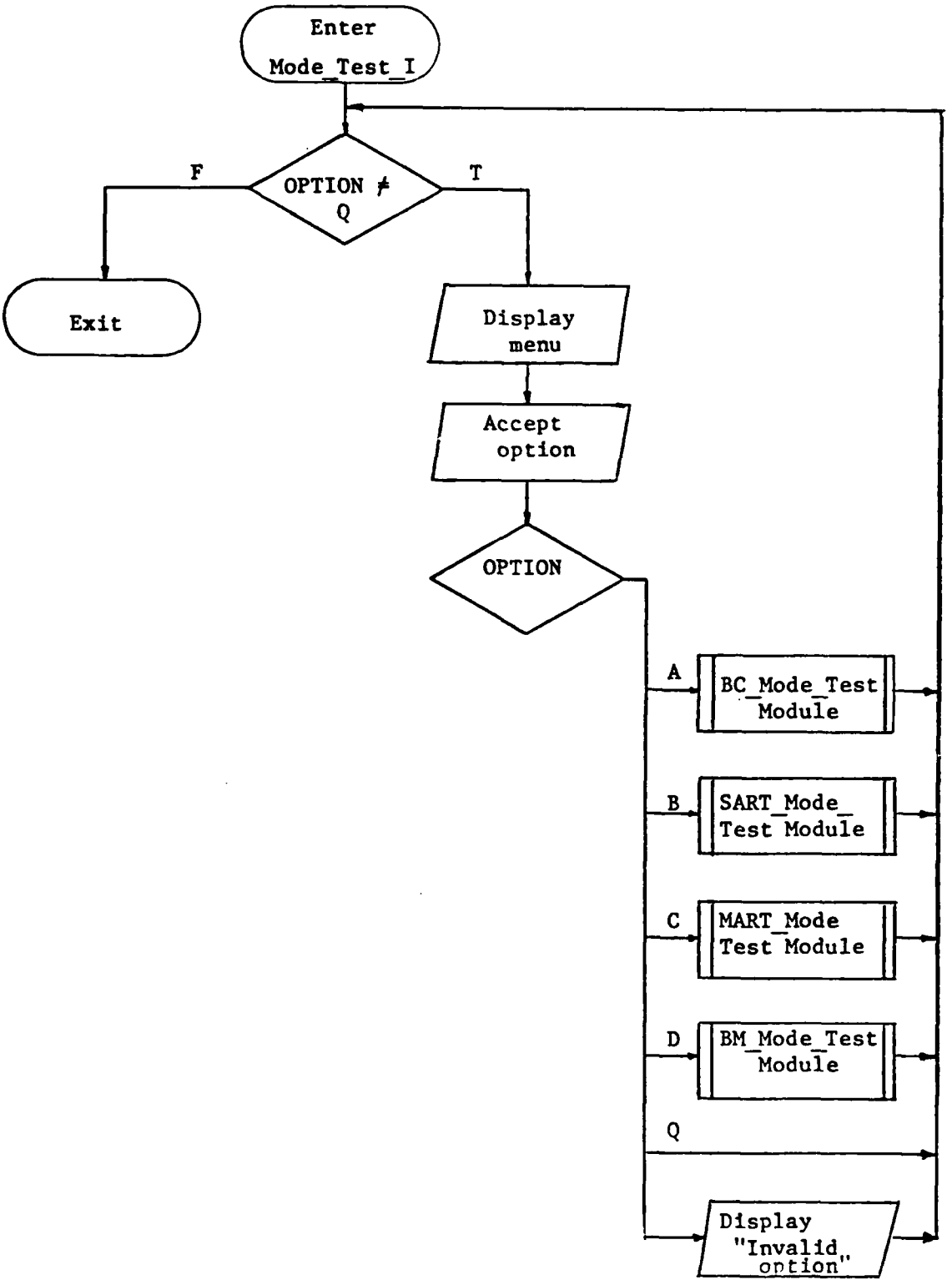


Figure IV-2. Mode_Test_I Flowchart.

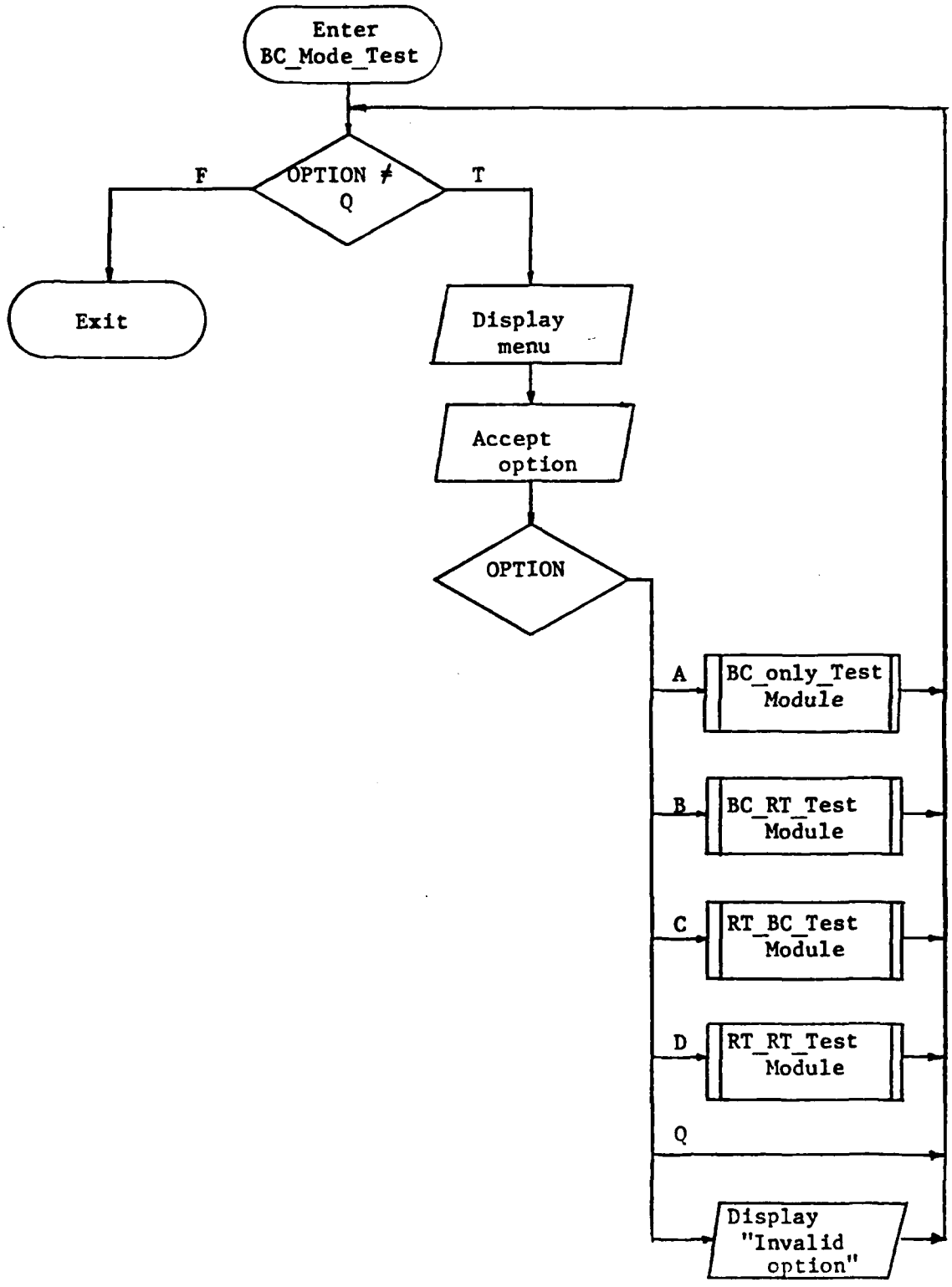


Figure IV-3. BC_Mode_Test Flowchart.

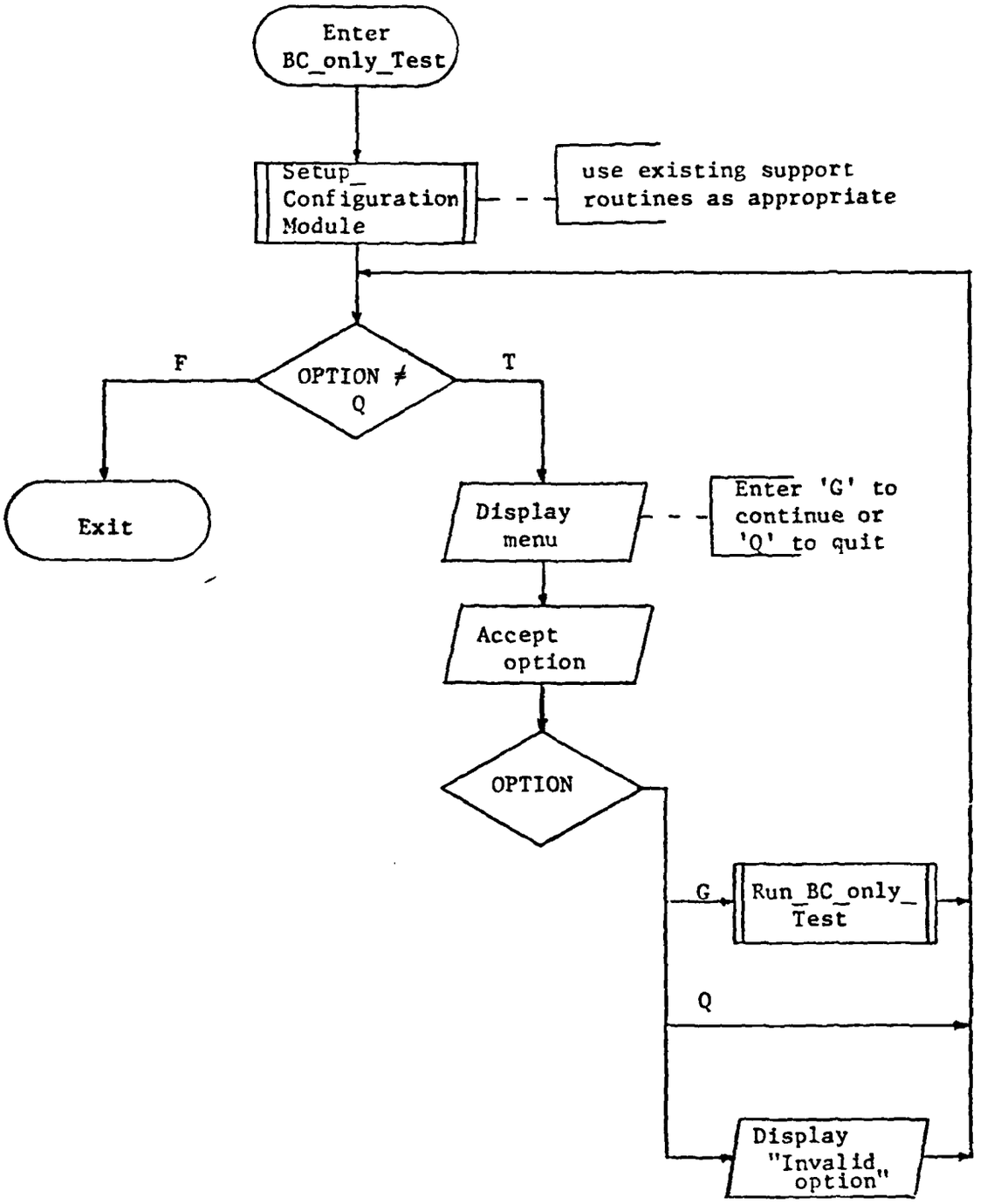


Figure IV-4. BC_only_Test Flowchart.

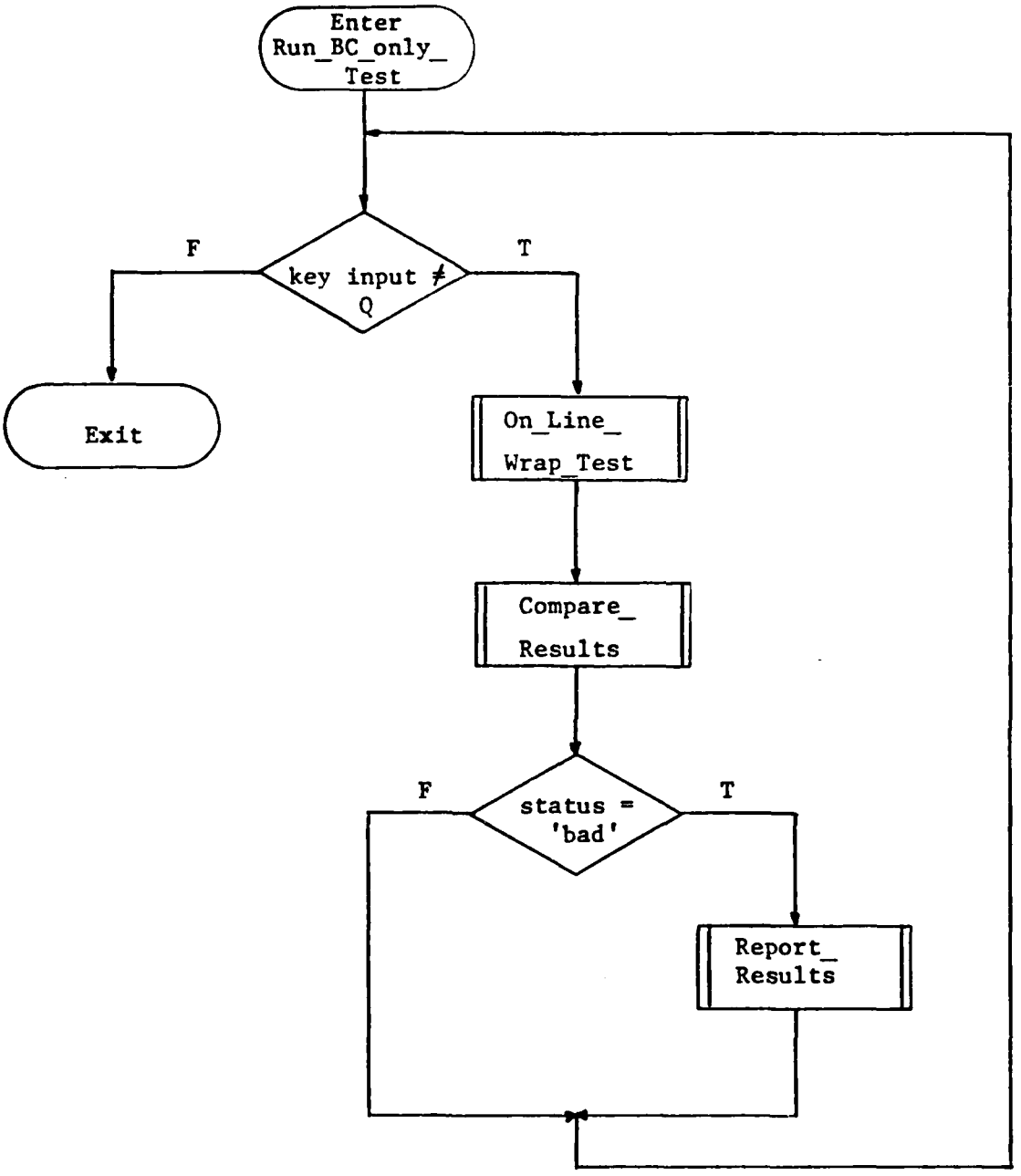


Figure IV-4 (cont). BC_only_Test Flowchart.

General Comments

The sample executive modules and test modules above are recommended as guidelines for future programming efforts. All or a portion of them may be implemented. It is recommended that any existing support routines which are applicable be used wherever appropriate (Appendix A shows many of the modules currently being used to exercise a GPBIU in Bus Controller mode). For example, the Setup-Configuration modules shown throughout chapter three in the hierarchy charts will utilize modules such as Init and Select/Restart and those modules subordinate to them as necessary. Many of the modules in Appendix A are ENASF utility routines for clearing a CRT screen, updating a CRT screen or displaying to the screen. These modules will be utilized as appropriate in the detailed design.

V. Conclusions

Introduction

The approach to developing a GPBIU test system represented by this thesis resulted from an organization's need to determine that a GPBIU is functioning correctly. As stated in chapter one, showing correct GPBIU functioning is very important for the two following reasons: 1) Being able to prove a GPBIU is not the cause of a system failure will prevent invalid service calls to the vendor of the host computer system. This will result in direct dollar savings to the Air Force. 2) Being able thru software testing to prove a GPBIU is not the cause of a system failure or message transmission fault will eliminate the process of having to power down and up again the host system to remove and test a GPBIU off-line everytime something appears abnormal. This will benefit the users by eliminating the requirement to log off the system until the GPBIU is removed or reinstalled. The host system will benefit by eliminating the undesirable action of powering the system down and up so frequently to allow the GPBIU to be removed or reinstalled.

Designing such a test system as required is a non-trivial task. Designing and writing software which tests a piece of hardware is a complex task for someone not familiar with this kind of programming and especially when few hard copy requirements or specifications exist for

producing such software. Much time was spent during this project just to produce some requirements and specifications to precede the preliminary design. Just identifying what tests should be run and generally what areas should be tested is a most challenging problem. I referred in chapter one of this project to Lawrence J. Peters' notion of a "wicked" problem. I found this project to be such a "wicked" problem in that a solution in one area generally resulted in more questions being asked about that area. I tried to answer those questions in general terms so as to produce high level requirements and specifications which could survive as new GPBIUs are developed and a partial sample design of a test system. The requirements, specifications and design are mostly in the form of hierarchy charts and schematics to add clarity to the text. A chart in chapter two was also produced as a summary of the required tests and the areas tested by them.

Findings and Recommendations

The following findings and recommendations are identified as the major thrust for present and near-future GPBIU software test development.

Finding No. 1: Little documentation exists for anyone unfamiliar with the operation and testing of a GPBIU. Most information is contained in the heads of key personnel and technicians. Documentation which is written down is generally technical and complex.

Recommendation No. 1: Existing documentation concerning GPBIU testing must be upgraded as soon as possible. Current ENASF test procedures will have to continue to be used until a more comprehensive GPBIU test software system is implemented. Adding to this recommendation is the fact that key personnel often rotate out of military organizations. In fact, at least one such key individual has already left ENASF during the writing of this project. New documentation is needed to facilitate training of new personnel.

Finding No. 2: Thorough test procedures are non-existant.

Recommendation No. 2: A user guide telling procedurally how and why specific tests are configured and carried out is needed as part of the upgrade in documentation.

Finding No. 3: Accomplishing the required overall software test system will require a progression of follow-on thesis efforts.

Recommendation No. 3: It is recommended that an immediate follow-on thesis be accomplished to refine the preliminary requirements, specifications and design presented in this thesis and to accomplish the detailed design and implementation of the MODE-TEST-I portion proposed in this thesis. It is also recommended that other theses be accomplished to do the preliminary and detailed designs and implementation and integration of the other

subsystems (HOST-SYSTEM-VALIDATION and MODE-TEST-I1). These theses may be accomplished in succession or concurrently by individuals or concurrently by a team of individuals working closely with one another and with ENASF. ENASF should have the option of specifying how they prefer these projects be carried forward.

Finding No. 4: As stated in chapter four, the major software requirement is source compatability among the DEC computers at ENASF. Currently, if a GPBIU is mounted on the VAX 11/780, the VAX must be brought down and the GPBIU removed to a dedicated PDP 11/55 for testing.

Recommendation No. 4: ENASF has determined the FORTRAN language offers the required user friendliness and transportability among their systems. It is recommended an on-line test be developed to run on the VAX. This may be a modified version of the existing off-line test on the PDP 11/55 or may be a new test development but it should run on all systems.

Finding No. 5: Individuals writing software for the GPBIU software test system must become familiar with the interfaces among and between a MIL-STD-1553 data bus and the DEC computer systems to include memory referencing and configuration. The VAX 11/780 interface is slightly different than the PDPs interface to memory.

Recommendation No. 5: The GPBIU Programmer's Reference Guide (Reference 1 in the Bibliography) provides insight into the problem of interfacing to memory on the

various systems. It has been upgraded during this writing. Also, since the FORTRAN programming language will be used, individuals doing the coding must be familiar with utilizing common blocks to reference data locations in memory. The MIL-STD-1553 Multiplex Applications Handbook will also be of help.

Finding No. 6: A new GPBIU is currently under development.

Recommendation No. 6: Utilizing existing GPBIUs and recommending and writing software procedures for testing GPBIU operation will lead to much new and valuable information being learned about GPBIUs. Writing such test procedures for upgraded GPBIUs will be an evolving process based on where we came from and where we're going in respect to their development and how people understand their operation. The concept of the GPBIU as a piece of hardware is not so hard to grasp but the writing of software programs which must deal with various interfaces and cover such a broad range of applications is very complex work. Software tests should be written for the existing GPBIUs and the proposed requirements and specifications updated as new GPBIUs are developed. Only in the detailed design of the test software should the new specifics of any new GPBIU have to be a new major accomplishment.

Finding No. 7: Exhaustively testing a GPBIU is not feasible since the fault dictionary for the implemented

word size would be too large.

Recommendation No. 7: The test software should utilize a form of controlled random testing to hold certain areas fixed while increasing the frequency of change in other areas. Also, selected patterns and boundaries should be chosen for testing. For example: all zeroes, all ones, alternating zeroes and ones, left half zeroes - right half ones, etc. These tests should be under a controlled loop to allow the time periods of tests to be extended to offer greater confidence that a GPBIU is functioning properly. This may be accomplished by having software programs check the keyboard for a termination character each time through the loop.

Closing Remarks

The GPBIU test software system will evolve through many phases as more of the system is defined and implemented. The system must be adaptable to changes in future GPBIU development. For these reasons, this thesis proposed high level requirements and specifications necessary to test a GPBIU. The detailed design will be somewhat dependent upon the specific generation GPBIU but it should also be developed to be as flexible to changes as possible. Also, the software must be user friendly and include complete documented user guides and references to facilitate training of new personnel assigned to work with the GPBIUs. Data dictionaries must be developed. These

will describe how, why and where data elements are used. Modules must also be documented in module data dictionaries which include such information as module number, what the module does, where the module is called from and who it calls. (NOTE: Module numbers are nonexistant in the RTX system of Appendix A. This made it very difficult to determine if all modules which exist are covered in the diagram.) Inadequate documentation has been the downfall of many systems. Future thesis efforts must incorporate deliberate accomplishment of providing well documented software systems in accordance with today's accepted methods of software design.

Bibliography

1. GPBIU Programmers Reference Guide, Sgt Jim Bennett, Air Force Systems Command, Aeronautical Systems Division (ENASF), Wright Patterson Air Force Base, Ohio, 45433, June, 1983.

2. Konno, Leslie T., Bus Interface Unit Design for the Distributed Processor/Memory System , Master's Thesis. Wright-Patterson Air Force Base, Ohio, Air Force Institute of Technology, December, 1977.

3. MIL-STD-1553B, Aircraft Internal Time Division Command/Response Multiplex Data Bus, Department of Defense, 21 September, 1978.

4. MIL-STD-1553 Multiplex Applications Handbook, Final Technical Report, Air Force Systems Command, Aeronautical Systems Division (ENASF), Wright Patterson Air Force Base, Ohio, 45433, 1 May, 1980.

5. Porter, David R. and Willis M. Muska. "Optical Data Bus Receiver for MIL-STD-1553B," IEEE, NAECON, 1319-1324, 1981.

6. Sanducci, Anthony F. "MIL-STD-1553 Bus Controller and Multiplex Remote Terminal," IEEE, NAECON, 1311-1318, 1981.

7. VAX 11/780 Architecture Handbook , Digital Equipment Corporation, 1977.

8. Yourdon, Edward and Larry R. Constantine, Structured Design , New York: Yourdon Press, 1978.

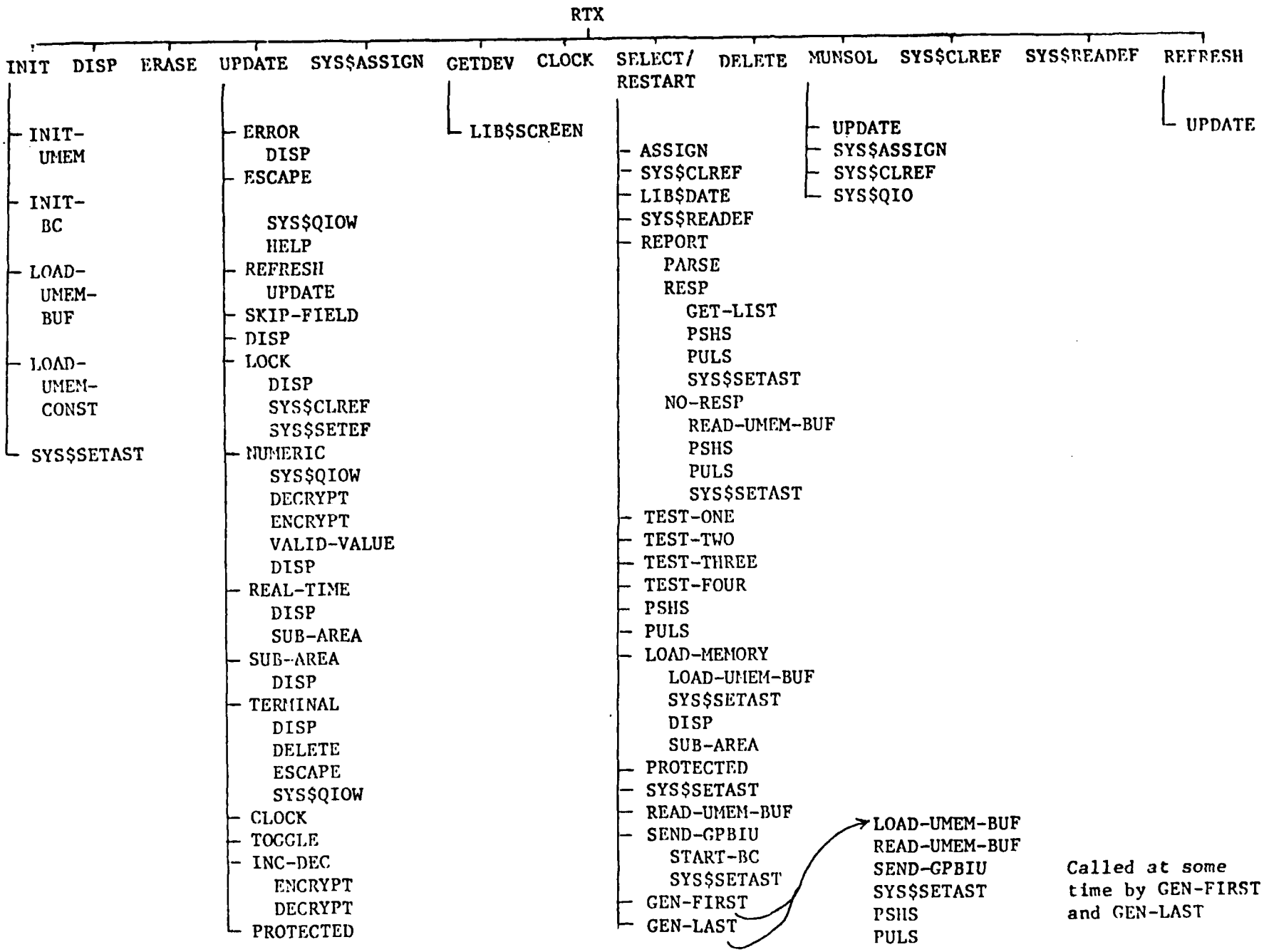
9. Peters, Lawrence J., Software Design: Methods & Techniques , New York, Yourdon Press, 1981.

10. GPBIU Internal Memoranda, Air Force Systems Command, Aeronautical Systems Division (ENASF), Wright Patterson Air Force Base, Ohio, 45433.

Appendix A

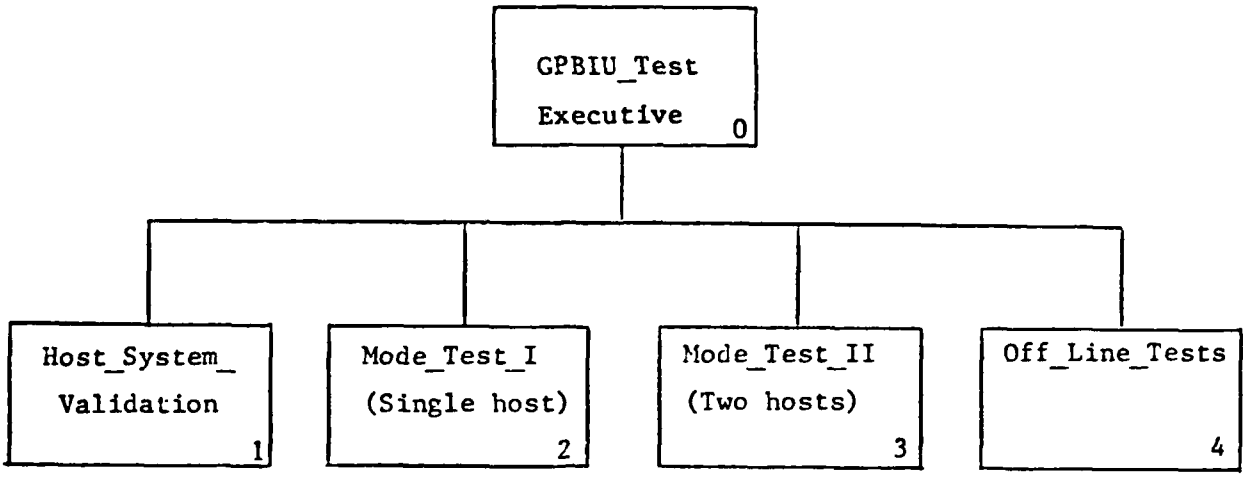
Existing Hierarchical Chart for RTX

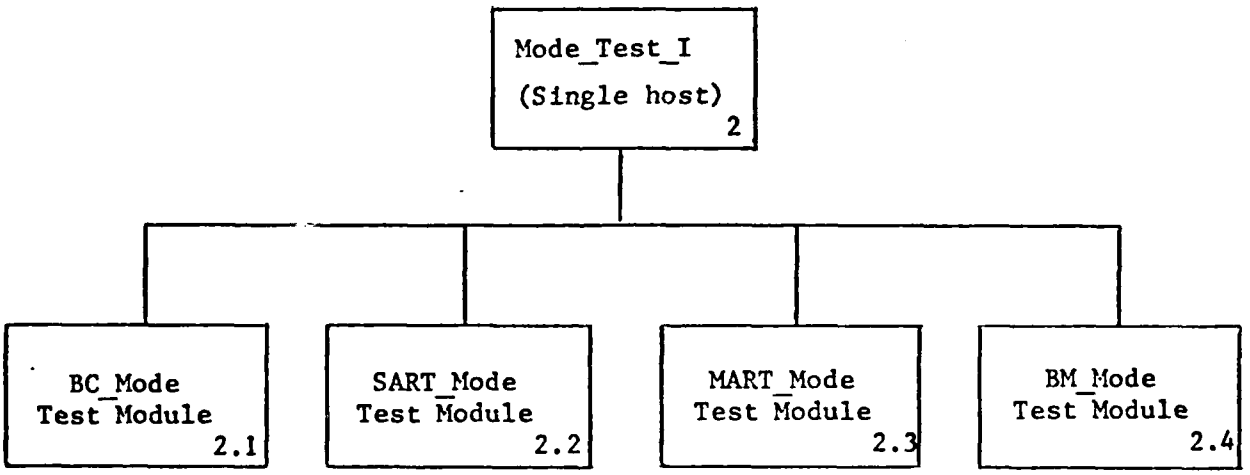
(Remote Terminal Test)

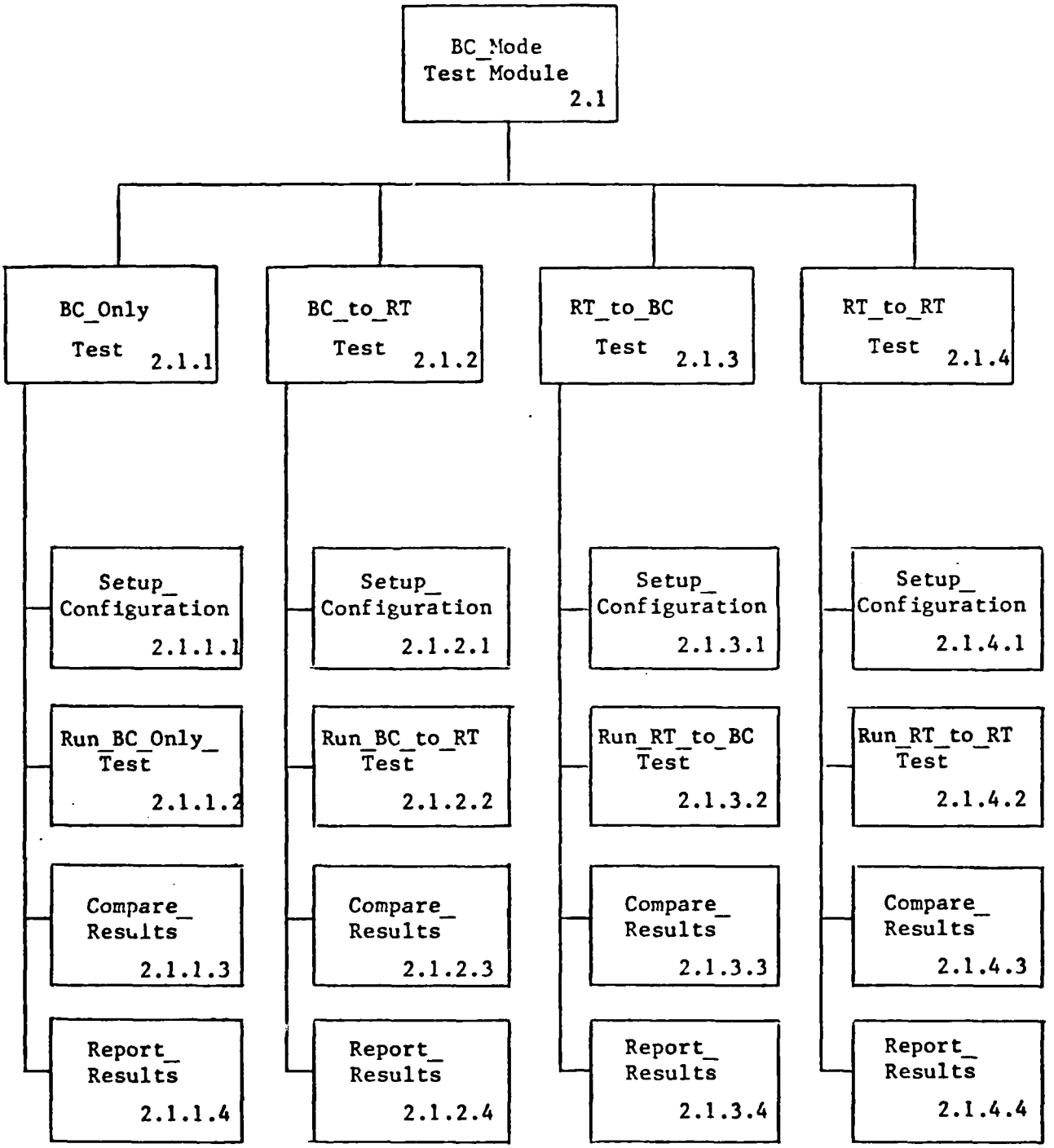

RTX
INIT
DISP
ERASE
UPDATE
SYS$ASSIGN
GETDEV
CLOCK
SELECT/ RESTART
DELETE
MUNSOL
SYS$CLREF
SYS$READEF
REFRESH
INIT-UMEM
INIT-BC
LOAD-UMEM-BUF
LOAD-UMEM-CONST
SYS$SETAST
ERROR
DISP
ESCAPE
SYS$QIOW
HELP
REFRESH
UPDATE
SKIP-FIELD
DISP
LOCK
DISP
SYS$CLREF
SYS$SETEF
NUMERIC
SYS$QIOW
DECRYPT
ENCRYPT
VALID-VALUE
DISP
REAL-TIME
DISP
SUB-AREA
SUB-AREA
DISP
TERMINAL
DISP
DELETE
ESCAPE
SYS$QIOW
CLOCK
TOGGLE
INC-DEC
ENCRYPT
DECRYPT
PROTECTED
LIB$SCREEN
ASSIGN
SYS$CLREF
LIB$DATE
SYS$READEF
REPORT
PARSE
RESP
GET-LIST
PSHS
PULS
SYS$SETAST
NO-RESP
READ-UMEM-BUF
PSHS
PULS
SYS$SETAST
TEST-ONE
TEST-TWO
TEST-THREE
TEST-FOUR
PSHS
PULS
LOAD-MEMORY
LOAD-UMEM-BUF
SYS$SETAST
DISP
SUB-AREA
PROTECTED
SYS$SETAST
READ-UMEM-BUF
SEND-GPBIU
START-BC
SYS$SETAST
GEN-FIRST
GEN-LAST
UPDATE
SYS$ASSIGN
SYS$CLREF
SYS$QIO
UPDATE
LOAD-UMEM-BUF
READ-UMEM-BUF
SEND-GPBIU
SYS$SETAST
PSHS
PULS
Called at some time by GEN-FIRST and GEN-LAST

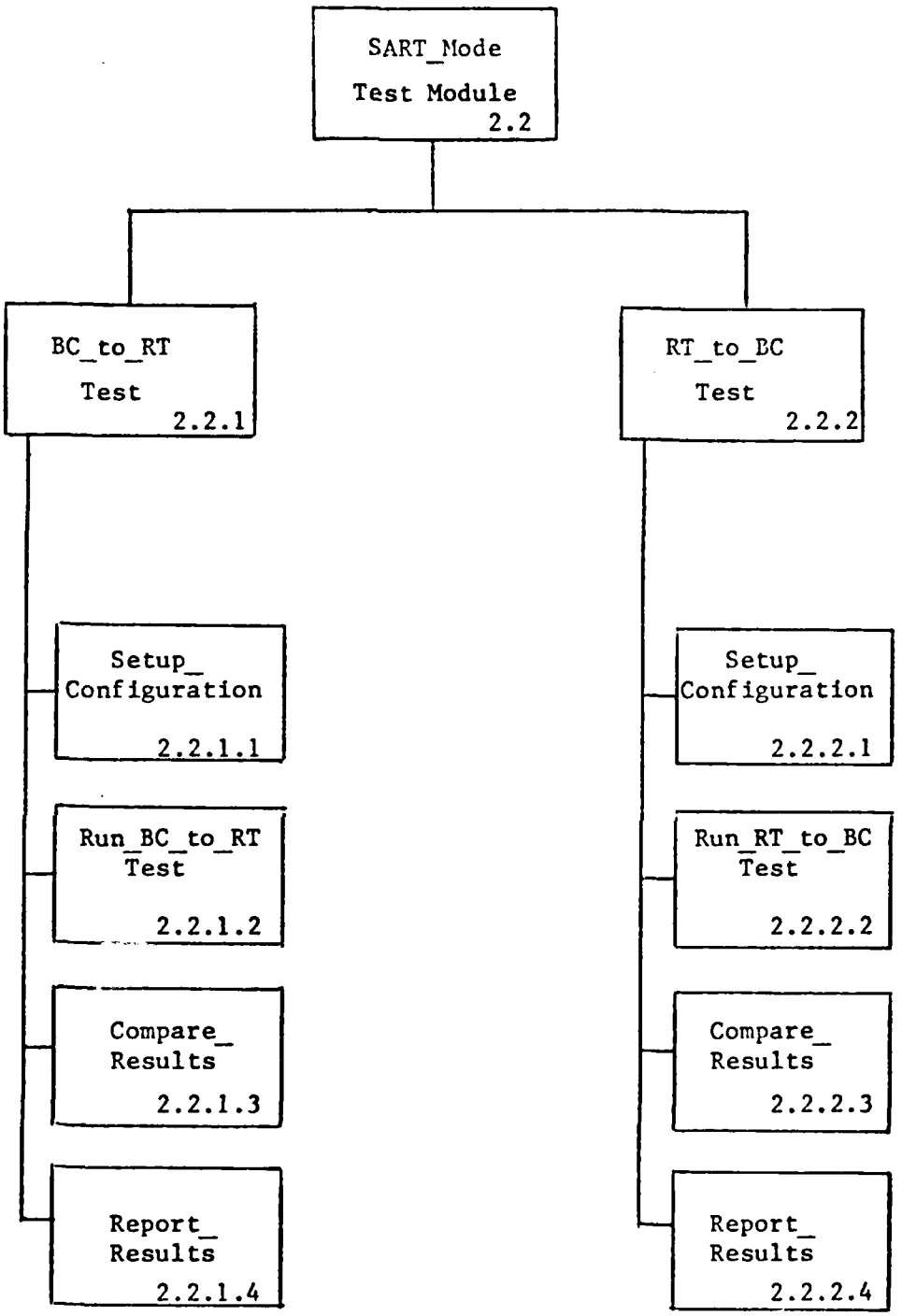
Appendix B

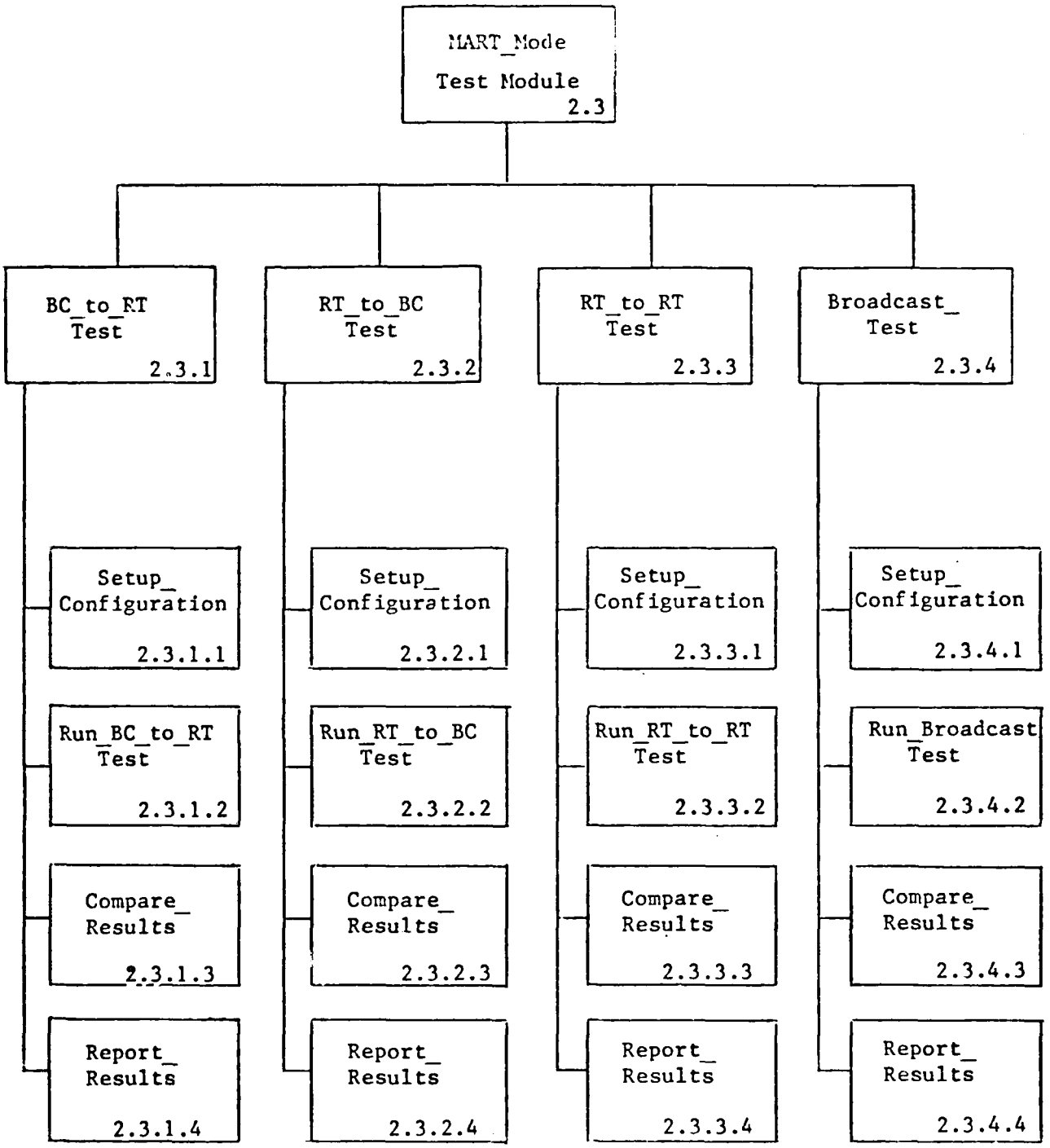
Summary Hierarchy Charts for Proposed GPBIU Test System


GPBIU_Test
Executive
0
Host_System_
Validation
1
Mode_Test_I
(Single host)
2
Mode_Test_II
(Two hosts)
3
Off_Line_Tests
4

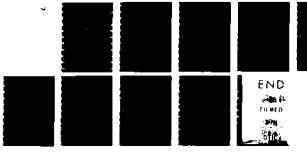

Mode_Test_I
(Single host)
2
BC_Mode
Test Module
2.1
SART_Mode
Test Module
2.2
MART_Mode
Test Module
2.3
BM_Mode
Test Module
2.4

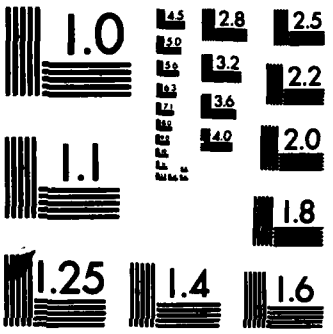

BC_Mode
Test Module
2.1
BC_Only
Test 2.1.1
BC_to_RT
Test 2.1.2
RT_to_BC
Test 2.1.3
RT_to_RT
Test 2.1.4
Setup_
Configuration
2.1.1.1
Run_BC_Only_
Test
2.1.1.2
Compare_
Results
2.1.1.3
Report_
Results
2.1.1.4
Setup_
Configuration
2.1.2.1
Run_BC_to_RT
Test
2.1.2.2
Compare_
Results
2.1.2.3
Report_
Results
2.1.2.4
Setup_
Configuration
2.1.3.1
Run_RT_to_BC
Test
2.1.3.2
Compare_
Results
2.1.3.3
Report_
Results
2.1.3.4
Setup_
Configuration
2.1.4.1
Run_RT_to_RT
Test
2.1.4.2
Compare_
Results
2.1.4.3
Report_
Results
2.1.4.4

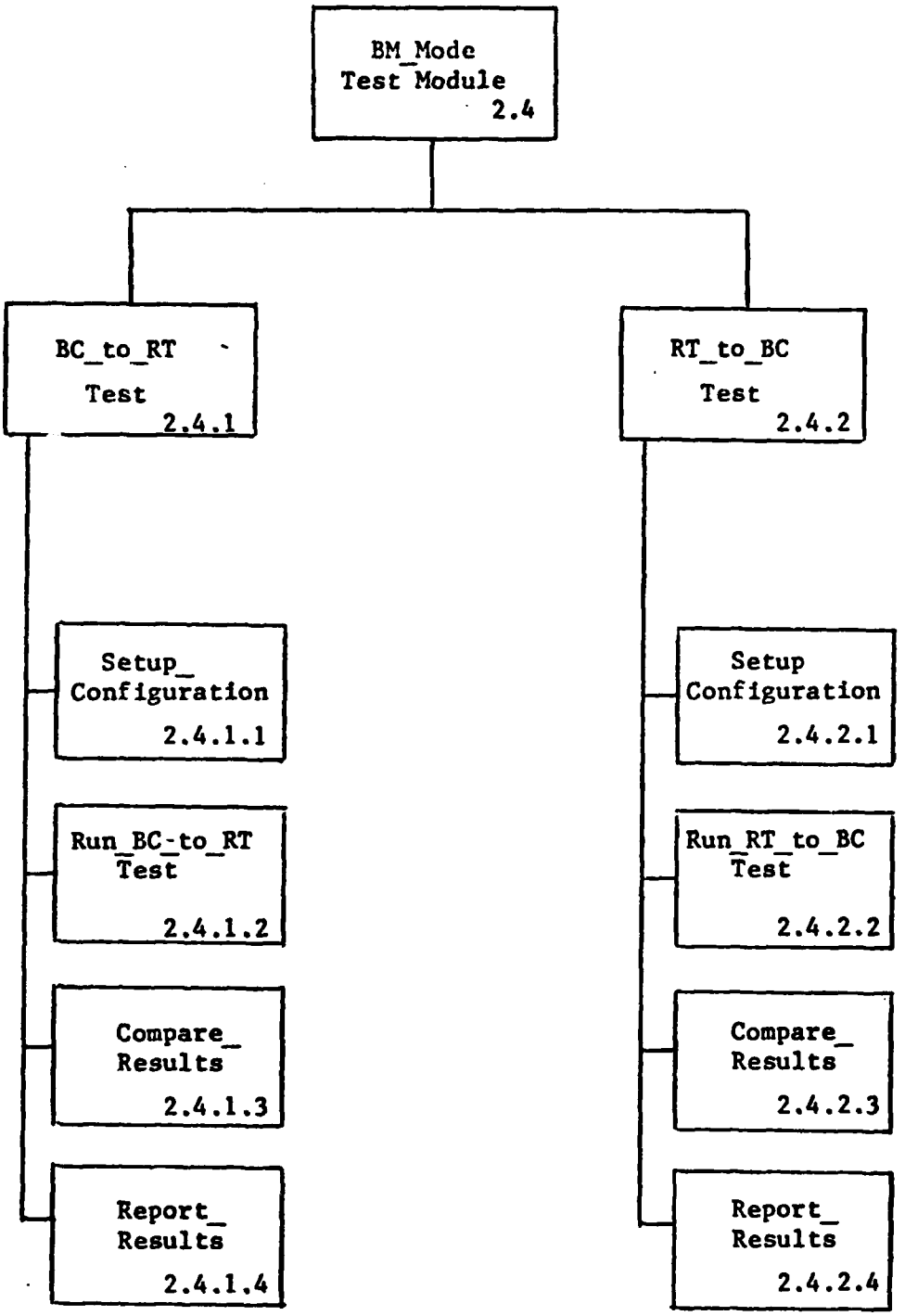

SART_Mode
Test Module
2.2
BC_to_RT
Test
2.2.1
RT_to_BC
Test
2.2.2
Setup_
Configuration
2.2.1.1
Run_BC_to_RT
Test
2.2.1.2
Compare_
Results
2.2.1.3
Report_
Results
2.2.1.4
Setup_
Configuration
2.2.2.1
Run_RT_to_BC
Test
2.2.2.2
Compare_
Results
2.2.2.3
Report_
Results
2.2.2.4


MART_Mode
Test Module
2.3
BC_to_RT
Test
2.3.1
RT_to_BC
Test
2.3.2
RT_to_RT
Test
2.3.3
Broadcast_
Test
2.3.4
Setup_
Configuration
2.3.1.1
Run_BC_to_RT
Test
2.3.1.2
Compare_
Results
2.3.1.3
Report_
Results
2.3.1.4
Setup_
Configuration
2.3.2.1
Run_RT_to_BC
Test
2.3.2.2
Compare_
Results
2.3.2.3
Report_
Results
2.3.2.4
Setup_
Configuration
2.3.3.1
Run_RT_to_RT
Test
2.3.3.2
Compare_
Results
2.3.3.3
Report_
Results
2.3.3.4
Setup_
Configuration
2.3.4.1
Run_Broadcast
Test
2.3.4.2
Compare_
Results
2.3.4.3
Report_
Results
2.3.4.4

AD-A138 005 GENERAL PURPOSE BUS INTERFACE UNIT (GPBIU) SYSTEM TEST 2/2
SOFTWARE DESIGN(U) AIR FORCE INST OF TECH
WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI.. R A LINDSEY
UNCLASSIFIED 16 DEC 83 AFIT/GCS/EE/84D-64 F/G 9/2 NL






BM_Mode
Test Module
2.4
BC_to_RT
Test
2.4.1
RT_to_BC
Test
2.4.2
Setup_
Configuration
2.4.1.1
Run_BC-to_RT
Test
2.4.1.2
Compare_
Results
2.4.1.3
Report_
Results
2.4.1.4
Setup
Configuration
2.4.2.1
Run_RT_to_BC
Test
2.4.2.2
Compare_
Results
2.4.2.3
Report_
Results
2.4.2.4

Appendix C

Program Design Language for Sample Design Flowcharts in Chapter IV

Program Design
Generated on a VAX 11/780
at ASD/ENASD
WPAFB,OH 45433

****
*
* GPBIU Test System PDL *
*
* Nov 1983 *
*
* PDL 03.06 *
*
****

GPBIU_TEST_EXECUTIVE DESCRIPTION BLOCK

SEGMENT NAME: GPBIU_TEST_EXECUTIVE

PURPOSE: Executive for GPBIU Test System

AUTHOR: Rodger A. Lindsey, Captain, USAF

CALLING SEQUENCE: HOST_SYSTEM_VALIDATION
MODE_TEST_I
MODE_TEST_II
OFF_LINE_TESTS

INPUTS: Select option from menu

OUTPUTS:

GPBIU_TEST_EXECUTIVE

```
START
DO  WHILE OPTION IS NOT EQUAL TO Q
    DISPLAY MENU
    ACCEPT OPTION FROM KEYBOARD
    CASE  OF OPTION
        A:
        CALL  HOST_SYSTEM_VALIDATION MODULE
        B:
        CALL  MODE_TEST_I MODULE
        C:
        CALL  MODE_TEST_II MODULE
        Z:
        CALL  OFF_LINE_TESTS MODULE
        Q:
        NO OP
        OTHERWISE:
        DISPLAY INVALID OPTION MESSAGE
    ENDCASE
ENDDO
END
```

MODE_TEST_I DESCRIPTION BLOCK

SEGMENT NAME: MODE_TEST_I

PURPOSE: Controls testing of GPBIU on a single host.

AUTHOR: Rodger A. Lindsey, Captain, USAF

CALLING SEQUENCE: BC_MODE_TEST
SART_MODE_TEST
MART_MODE_TEST
BM_MODE_TEST

INPUTS: Select option from menu

OUTPUTS:

MODE_TEST_I

```
ENTER
DO  WHILE OPTION IS NOT EQUAL TO Q
    DISPLAY MENU
    ACCEPT OPTION FROM THE KEYBOARD
    CASE  OF OPTION
        A:
        CALL  BC_MODE_TEST MODULE
        B:
        CALL  SART_MODE_TEST MODULE
        C:
        CALL  MART_MODE_TEST MODULE
        D:
        CALL  BM_MODE_TEST MODULE
        Q:
        NO OP
        OTHERWISE:
        DISPLAY INVALID OPTION MESSAGE
    ENDCASE
ENDDO
RETURN
```

BC_MODE_TEST DESCRIPTION BLOCK

SEGMENT NAME: BC_MODE_TEST

PURPOSE: Controls testing of GPBIU in Bus Controller mode.

AUTHOR: Rodger A. Lindsey, Captain, USAF

CALLING SEQUENCE: BC_ONLY_TEST
BC_RT_TEST
RT_BC_TEST
RT_RT_TEST

INPUTS: Select option from menu

OUTPUTS:

BC_MODE_TEST

```
ENTER
DO  WHILE OPTION IS NOT EQUAL TO Q
    DISPLAY MENU
    ACCEPT OPTION FROM THE KEYBOARD
    CASE  OF OPTION
        A:
        CALL  BC_ONLY_TEST MODULE
        B:
        CALL  BC_RT_TEST MODULE
        C:
        CALL  RT_BC_TEST MODULE
        D:
        CALL  RT_RT_TEST MODULE
        Q:
        NO OP
        OTHERWISE:
        DISPLAY INVALID OPTION MESSAGE
    ENDCASE
ENDDO
RETURN
```

BC_ONLY_TEST DESCRIPTION BLOCK

SEGMENT NAME: BC_ONLY_TEST

PURPOSE: Places the GPBIU in Bus Controller mode, and calls the routine to do the internal wrap test.

AUTHOR: Rodger A. Lindsey, Captain, USAF

CALLING SEQUENCE: RUN_BC_ONLY_TEST

INPUTS: Operator inputs addresses to establish message blocks.

OUTPUTS:

BC_ONLY_TEST

```
ENTER
CALL  SETUP_CONFIGURATION MODULE
DO  WHILE OPTION IS NOT EQUAL TO Q
    DISPLAY MENU
    ACCEPT OPTION FROM KEYBOARD
    CASE  OF OPTION
        G:
        CALL  RUN_BC_ONLY_TEST MODULE
        Q:
        NO OP
        OTHERWISE:
        DISPLAY INVALID OPTION MESSAGE
    ENDCASE
ENDDO
RETURN
```

RUN_BC_ONLY_TEST DESCRIPTION BLOCK

SEGMENT NAME: RUN_BC_ONLY_TEST

PURPOSE: Routes data from output to input ports of a GPBIU. Compares the resultant message block with the original block and reports differences.

AUTHOR: Rodger A. Lindsey, Captain, USAF

CALLING SEQUENCE: ON_LINE_WRAP_TEST
COMPARE_RESULTS
REPORT_RESULTS

INPUTS: Utilizes data established in the SETUP_CONFIGURATION module of the BC_ONLY_TEST module.

OUTPUTS: Memory dump or terminal display when a faulty transfer is discovered.

RUN_BC_ONLY_TEST

```
ENTER
DO  WHILE KEYBOARD INPUT NOT EQUAL TO Q
    CALL  ON_LINE_WRAP_TEST
    CALL  COMPARE_RESULTS
    IF  STATUS IS 'BAD'  THEN
          CALL  REPORT_RESULTS
    ENDIF
ENDDO
RETURN
```

VITA

Rodger A. Lindsey was born in Villa Rica, Georgia, on 10 January 1953 to Herbert C. and Ezma I. Lindsey. After graduation from high school in 1970, he attended Auburn University for one year, Marietta-Cobb Area Vocational School for one year and spent three years in the United States Army Security Agency before graduating from Georgia Southern College in Statesboro, Georgia in 1979 with a Bachelor of Science Degree in Management. He then attended the United States Air Force's Officer Training School and receieved his commission in September 1979. For the next two years and nine months he instructed students in the Computer Systems Development Officer Course (5131 AFSC awarding course) at Keesler Air Force Base, Biloxi, Mississippi. In June 1982 he entered the Air Force Institute of Technology for a Master's Degree in Computer Systems.

Permanent Address: c/o Herbert C. Lindsey
Route 2
Dallas, Georgia 30132

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION: UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S): AFIT/GCS/EE/83D-64	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION: School of Engineering; 6b. OFFICE SYMBOL (If applicable): AFIT/EN	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State and ZIP Code): Air Force Institute of Technology, Wright-Patterson AFB, Ohio 45433	7b. ADDRESS (City, State and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION: Air Force Systems Command, Aeronautical Systems Division (ENASF); 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State and ZIP Code): Wright-Patterson AFB, Ohio 45433	10. SOURCE OF FUNDING NOS. — PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT NO.
11. TITLE (Include Security Classification): see box 19	
12. PERSONAL AUTHOR(S): Rodger A. Lindsey, Capt, USAF	
13a. TYPE OF REPORT: MS Thesis; 13b. TIME COVERED: FROM _____ TO _____	14. DATE OF REPORT (Yr., Mo., Day): 1983, Dec. 16; 15. PAGE COUNT: 100
16. SUPPLEMENTARY NOTATION	Approved for public release: IAW AFR 190-17. LYNN E. WOLAVER, Dean for Research and Professional Development, Air Force Institute of Technology (ATC), 7 Feb 84
17. COSATI CODES — FIELD / GROUP / SUB. GR.	18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number): GPBIU as interface between host computer and MIL-STD-1553 data bus; validate message traffic; test system requirements functional requirements; preliminary subsystems requirements

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Title: GENERAL PURPOSE BUS INTERFACE UNIT (GPBIU) SYSTEM TEST SOFTWARE DESIGN (UNCLASSIFIED)

This report outlines preliminary high level system requirements, specifications and sample design for developing a software test package for functionally testing a General Purpose Bus Interface Unit (GPBIU). These tests are needed to verify GPBIUs are functioning properly because they are used in direct support of standards testing of avionics interconnected subsystems (radar, navigation, target identification, sensors, displays, etc.).

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT: UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION: UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL: HAROLD W. CARTER, Lt Col, USAF	22b. TELEPHONE NUMBER (Include Area Code): (513) 255-6913; 22c. OFFICE SYMBOL: AFIT/ENG

DD FORM 1473, 83 APR — EDITION OF 1 JAN 73 IS OBSOLETE.

Box 18 (cont.)

and specifications; summary flowcharts and PDL; summary hierarchy charts; software requirements; bus controller mode; single and multiple address remote terminal modes; bus monitor mode.


FIL