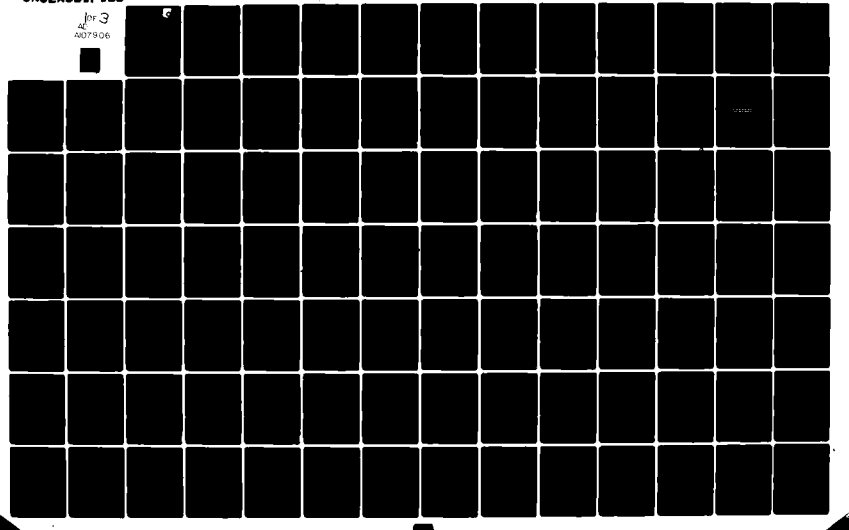
AD-A107 906 TRW DEFENSE AND SPACE SYSTEMS GROUP REDONDO BEACH CA F/G 9/2
DIGITAL AVIONICS INFORMATION SYSTEM (DAIS): DEVELOPMENT AND DEM--ETC(U)
SEP 81 M J COOK, R C MASON, J L STAUTBERG F33615-78-C-1502
UNCLASSIFIED AFWAL-TR-81-1165 NL



LEVEL II

12

AD A107906

AFWAL-TR-81-1165

DIGITAL AVIONICS INFORMATION SYSTEM (DAIS): DEVELOPMENT AND DEMONSTRATION

TRW DEFENSE AND SPACE SYSTEMS GROUP
ONE SPACE PARK
REDONDO BEACH, CALIFORNIA 90278

September 1981

Final Report for Period 1 October 1978 to 31 July 1981

Approved for public release; distribution unlimited

AVIONICS LABORATORY
AIR FORCE WRIGHT AERONAUTICAL LABORATORIES
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture use, or sell any patented invention that may in any way be related thereto.

This report has been reviewed by the Office of Public Affairs (ASD/PA) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.

Boyd E. Holsapple

BOYD E. HOLSAPPLE
PROJECT ENGINEER

L. Daniel Snyder

L. DANIEL SNYDER, Chief
Mission Software & Sys Integration Gp
System Avionics Division

FOR THE COMMANDER

Richard H Boivin

RICHARD H. BOIVIN, Col, USAF
Chief, System Avionics Division
Avionics Laboratory

"If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization please notify AFWAL/AAAS-1, *W-PAFB, OH 45433 to help us maintain a current mailing list".*

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: AFWAL-TR-81-1165	2. GOVT ACCESSION NO.: AD-A107 906
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE *(and Subtitle)*: DIGITAL AVIONICS INFORMATION SYSTEM (DAIS): DEVELOPMENT AND DEMONSTRATION	5. TYPE OF REPORT & PERIOD COVERED: FINAL REPORT 1 Oct 78 - 31 Jul 81
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): M. J. Cook, R. C. Mason, J. L. Stautberg, L. E. Self, R. L. Ellison, M. J. Strathman	8. CONTRACT OR GRANT NUMBER(s): F33615-78-C-1502
9. PERFORMING ORGANIZATION NAME AND ADDRESS: TRW Defense and Space Systems Group, One Spack Park, Redondo Beach, California 90278	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 63243F, 2052 05 18
11. CONTROLLING OFFICE NAME AND ADDRESS: Avionics Laboratory (AFWAL/AAAS), Air Force Wright Aeronautical Laboratory, Wright-Patterson AFB, Ohio 45433	12. REPORT DATE: September 1981
	13. NUMBER OF PAGES: 201
14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)*	15. SECURITY CLASS. *(of this report)*: UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for Public Release; Distribution Unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

Avionics Software	Displays	Multiplex
Avionics Systems	JOVIAL	Processor
Computers	MIL-STD-1553	Simulation
Controls	MIL-STD-1589	
DAIS	MIL-STD-1750	

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

The Digital Avionics Information System (DAIS) represents a significant advance in the technology of avionics system architecture. DAIS is a total systems concept, exploiting standardization, modularity, and application independent executive software to provide a system architecture adaptable to many aircraft, missions, and avionics configurations and fully capable of accommodating new advances in technology. These fundamental system characteristics are described in this report; the specific system features which

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

20. provide these characteristics and attributes are presented.

The DAIS core elements along with support software were integrated into a hot bench support facility. Significant efforts were placed on developing and validating new military standards in the areas of multiplex systems (MIL-STD-1553B), processor instruction set architecture (MIL-STD-1750A), and software (MIL-STD-1589B, JOVIAL). This report describes the integration of the core elements, the validation of the military standards, and the phased demonstration of these military standards in a representative close-air = support scenario.

TABLE OF CONTENTS

TABLE OF CONTENTS (Con't)

TABLE OF CONTENTS (Con't)

TABLE OF CONTENTS (Con't)

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS (Con't)

LIST OF TABLES

GLOSSARY

ACRONYM	DEFINITION
ADI	Attitude Direction Indicator
Alt.	Altitude
AP	Armament Panel
AR	Aiming Reticle
AVSAIL	Avionic System Analysis and Integration Laboratory
BCI	Bus Control Interface (module embedded in the AN/AYK-15A processor)
BCIU	Bus Control Interface Unit
BCM	Bus Control Module
BFL	Bomb Fall Line
BIT	Built-In-Test
BMU	Bus Monitor Unit
CAS	Close Air Support
C&D	Controls and Displays
CBIS	Controls and Backup Instruments System
CCIP	Continuously Computed Impact Point
CIU	Console Intelligence Unit
CPCI	Computer Program Configuration Item
CRT	Cathode Ray Tube
DAIS	Digital Avionics Information System
DEK	Data Entry Keyboard
DISP	Display Process
DMA	Direct Memory Access
DS/MU	Display Switch/Memory Unit
EQUIP	Equipment Process
FIM	Facility Interface Module
FLIR	Forward Looking Infrared Radar
FPM	Flight Path Marker
HARS	Heading Attitude Reference System
HAS	Hardware Architecture Simulation System
HBC	Hot Bench Computer
HSD	Horizontal Situation Display
HSI	Horizontal Situation Indicator
HUD	Head Up Display
ICD	Interface Control Document
ICS	Interpretive Computer Simulation
ILS	Instrument Landing System
IM	Interface Module
IMFK	Integrated Multifunction Keyboard
INS	Inertial Navigation System
IP	Initialization Point
ITB	Integrated Test Bed

ACRONYM	DEFINITION
LDGP	Low Drag General Purpose
LRU	Line Replaceable Unit
LOS	Line of Sight
ME	Message Error; also Master Executive
MFK	Multifunction Keyboard
MMP	Master Mode Panel
MMU	Mass Memory Unit
MPD	Multi-Purpose Display
MPDG	Modular Programmable Display Generator
MTU	Multiplex Terminal Unit
MUX	Multiplex
NAV	Navigation
N.M.	Nautical Miles
OAP	Offset Aiming Point
OEM	Original Equipment Manufacturer
OFP	Operational Flight Program
OPS	Operational Sequencer
OTP	Operational Test Program
PAL	PALEFAC Auxiliary File
PALEFAC	Partitioning Analyzing and Linkage Editing Facility
PCP	Processor Control Panel
PDS	Power Distribution and Control System
PGI	PALEFAC Global Input
PIM	Processor Interface Module
PIO	Programmed Input/Output
PMC	Performance Monitor and Control
PMD	PALEFAC Mission Data
PMI	PALEFAC Module Input
PPI	PALEFAC Partitioning Information Files
PRE	Post Run Editor
RAM	Random Access Memory
RAT	Ram-Air Turbine
ROM	Read Only Memory
ROT	Rough Output Tape
RT	Remote Terminal
SCADU	Super Control and Display Unit
SCU	Sensor Controller Unit
SDVS	Software Design & Verification System
SITC	System Integration and Test Coordination
SLS	Statement Level Simulation
SSDF	Simulated Subsystem Data Formatter
SSIU	Simulated Subsystem Interface Unit
STS	Software Test St[illegible]

GLOSSARY (Con't)

ACRONYM	DEFINITION
TCC	Test Control Center
TCU	Timing and Control Unit--in Remote Terminal
TDM	Time Division Multiplex
T/F	Terminal Failure
T/R	Transmit Receive
URT	Universal Remote Terminal
VSD	Vertical Situation Display

1.0 INTRODUCTION AND SUMMARY

The Digital Avionics Information System (DAIS) represents a significant advance in the technology of avionics systems architecture. DAIS is a total systems concept, exploiting standardization, modularity, and application-independent executive software to provide a system architecture adaptable to many aircraft, missions, and avionics configurations and fully capable of accommodating new advances in technology. The DAIS architecture results in improved reliability and availability of avionics systems while at the same time reducing life cycle costs.

Under the total systems concept, DAIS elements are not dedicated to any one avionics function. Rather, the elements are used to perform the processing and integrate the functions associated with the avionic sensors and subsystems employed for a particular aircraft/configuration/mission. This is achieved in part by the use of AN/AYK-15A general purpose digital processors conforming to MIL-STD-1750A. The processors communicate with each other and other system elements (e.g. C&D subsystem) via a MIL-STD-1553B dual redundant multiplex data bus. Centralized system single point control is performed by a processor-resident software executive. Application Software is structured to provide modularity, reliability, and transferability. Both executive and application software are implemented primarily in a MIL-STD-1589B JOVIAL J73 higher order language. The software modularity is enforced and enhanced by a standard executive-to-applications software interface.

The basic elements of the DAIS architecture which can be reconfigured for various avionics applications are called DAIS core elements (or building blocks). The core elements consist of the DAIS processors, DAIS multiplex, DAIS Mission Software, and DAIS Controls and Displays. Additional elements are the support software elements, namely the JOVIAL compiler, the assembler/linker, and the Partitioning, Analyzing, and Linkage Editing Facility (PALEFAC). The core elements, along with support hardware and software were integrated into a hot bench support facility also known as the Integrated Test Bed (ITB). A postulated 1980's avionics mission was flown using real-time, pilot-in-the-loop simulations to demonstrate the DAIS architecture and concepts. The subject of this report is the description of the functional elements and operational characteristics of the final configuration of the DAIS ITB.

The final ITB configuration is the result of developments performed in the period covered by this report. The baseline for that development was the DAIS ITB as successfully demonstrated in September 1978 and documented in AFAL-TR-79-1027. The substance of the development was the incorporation of a number of system improvements, the integration of second generation core element hardware and the conversion to the newly adopted military standards: MIL-STD-1553B (Multiplex), MIL-STD-1750A (Instruction Set Architecture), and MIL-STD-1589B (Higher Order Language). The effort included the upgrade of all system level specifications to completely and accurately reflect the current system implementation.

This report presents a review of the DAIS architecture with a section on each of the core elements. A section on other key elements and each of the non-real-time support software elements is also included. System operation is described in detail, both normal operation and exception management. Principle

features of the architecture are discussed and the support facility is described. A separate discussion of the software and hardware conversion efforts and integration and testing sequences is presented. The demonstration mission scenarios are reviewed and project management and control procedures are discussed. Support provided to other Air Force programs is identified and a final set of conclusions and recommendations is presented.

Overall, this report presents and describes in detail an avionic system architecture compliant with the most recent military standards, already implemented and demonstrated, which is easily adaptable to many aircraft and missions.

2.0 BACKGROUND

The Digital Avionics Information System (DAIS) has been under development by the Air Force since 1973. Air Force avionics systems in existence in the early 1970's were largely analog systems interconnected by dedicated wiring harnesses. In addition to the weight penalty, these were difficult to maintain, difficult to modify in response to new threat environments, and difficult to upgrade to accommodate new technology. The initial thrust of DAIS was to investigate the use of general purpose digital computers for avionics processing and of multiplexing techniques for data interchange. In addition to these two specific areas of technology, DAIS also addressed the larger problem of providing a system architecture which could adapt available avionics subsystems and sensors to a common, modularly expandable avionics core.

The lack of commonality and compatibility was an area of major concern. The classical approach to avionics provided a system unique to each particular aircraft. Avionics systems tended to be snapshots of technology available at the time of development with little or nor relationship to previous work. The result is that aircraft which are part of the operational fleet at a given time have neither commonality of hardware or software, nor any sort of compatibility. The effects are felt in costs and reliability. Additional front-end design costs are incurred and a less mature product (hardware and software) is obtained on each new development or upgrade/retrofit effort.

Even more serious, from an operational viewpoint, is the limited adaptability of point-designed avionic systems to changing threat environments or expanded mission roles for the weapon system. Past experience has shown that the airframe lifetime is significantly longer than the useful (in the sense of operational capability) lifetime of the avionics. Hence, a typical weapon system is modified many times during the life of the airframe. The Digital Avionics Information System (DAIS) was conceived as a means of avoiding the problems inherent in the classical avionics approach. DAIS considered the avionics job as a hybrid of process control and information management functions and provided a system architecture independent of a particular airframe and, to a large extent, of the technology used to implement the system. An important aspect of the DAIS concept is that it views the avionics as a whole and not as a collection of previously defined systems.

By using this approach, common functions of the avionics system were identified which are required by each of the mission functions. This resulted in an integrated avionics system which accomplishes all of the required mission functions but is partitioned along the lines of processing hardware, software, information transfer, and display and control. A new partitioning of the avionics system was defined so that the mission functions could be mapped onto that partitioning. The next step was to apply some constraints to the evolving system architecture. For reasons of cost, maximum utilization of common hardware was desired. In order to enhance redundancy and error management and recovery, maximum sharing of information and maximum use of shared resources were desired. This was based on the premise that if the necessary information is available system-wide, the system can re-orient the task of selected resources to accomplish critical mission functions when primary resources have failed. One of the few performance improvement constraints was to provide a better method

of interfacing to the pilot. The classical system forces the pilot to process large amounts of raw data as well as make decisions. The DAIS approach was to produce data for the pilot and aid his decision process, thereby reducing pilot workload. The final constraint was to provide an open-ended system capability. The architecture chosen should be capable of handling tasks larger than any anticipated today and should not depend upon today's technology. The system should be able to expand or change by adding or modifying resources without affecting the architecture.

The topology chosen for DAIS was that of a distributed system. This allowed physically distributing the resources to enhance survivability and provided a means for expanding system capability without regard to physical placement of the resources. The architecture chosen for DAIS is a hierarchical system structure operating under centralized control. To the "user", therefore, DAIS appears as a centralized system. The implementation of DAIS required the development of certain building blocks to achieve the design goals. These building blocks are called the DAIS core elements and consist generically of processors, multiplex equipment, control and display subsystem and the software associated with the flight processors.

AFWAL initiated the DAIS program in 1973 with two separate contracts to Texas Instruments (F33615-73-C-1156) and General Dynamics (F33615-73-C-1244) to define and provide the guidelines for the design of the DAIS system. Following these studies, contracts to The Boeing Company (F33615-74-C-1108) and Texas Instruments (F33615-74-C-1023) were let to provide the initial hardware and software specifications for the DAIS core elements, and initial system designs for DAIS.

Subsequent to these studies, AFWAL let contracts to design and develop the core elements as shown below:

Multiplex Equipment	IBM
Processor	Westinghouse
Mission Software	Intermetrics
Controls & Displays	Hughes

Also, AFWAL let a System Integration and Test Coordination (SITC) contract with TRW Defense and Space Systems to support AFWAL in combining these core elements, along with support hardware and software, into an integrated test bed (ITB) or "hot bench".

The initial part of the SITC effort was the development and demonstration of the Hardware Architecture Simulation, a DAIS prototype, to investigate and verify the DAIS architecture. This effort provided confirmation of the feasibility of the DAIS architecture and supplied useful inputs to the concurrent development of the core elements. As the core elements were delivered, they were tested and integrated into the hot bench facility. This effort culminated in the successful demonstration of a Close Air Support mission in September 1978.

AFWAL awarded follow-on contracts to TRW (F33625-78-R-1502) for continued System Integration and Test Coordination (SITC) and to Intermetrics

(F33615-78-R-1542) for Mission Software Enhancements (MSE). Second generation core element hardware was procured as shown below:

Processors (dual procurement)	Westinghouse Sperry-Univac
Multiplex Equipment	IBM

In parallel and interactive with these efforts, the Air Force was pursuing the finalization of revisions to the key avionics military standards. These were approved as follows:

MIL-STD-1553B (Multiplex)	September 1978
MIL-STD-1750A (processor instruction set architecture)	July 1980
MIL-STD-1589B (high order language)	June 1980

AFWAL anticipated the importance of DAIS demonstrating an implementation of these standards and extended the SITC contract to accomplish the necessary conversions. The integration of the second generation core elements and the conversion to the new MIL-STD's was accomplished and a formal demonstration of the DAIS ITB was performed in March 1981. The final configuration of the ITB and efforts leading to it are the subjects of this report.

3.0 DAIS SYSTEM ARCHITECTURE

Historically, avionic systems have been established along semi-autonomous functional areas such as navigation, weapon delivery, stores management, flight control, communications, etc. Each of these functional areas would have an analog or digital system with its own processing, information transfer, control inputs, and display set. There would be an interface between functional areas only as necessary for interaction purposes, usually a one-of-a-kind, non-standard interface. The DAIS concept is that the processing, information transfer, control and display functions be common and service all the previously described functional areas on an integrated basis.

These basic functions are implemented via the core elements which are aircraft and mission independent. The specific avionics capabilities are then achieved by selecting the sensors, weapons, and other subsystems desired and installing the appropriate application software to control and process the avionics data. This functional architecture is represented in Figure 1.

The purpose of the DAIS concept is to reduce the proliferation and non-standardization of aircraft avionics and permit the Air Force to assume the initiative in the specification of standard avionic systems and interfaces for future Air Force System acquisitions. This approach of isolating the core elements, and separately developing and verifying them should also aid Air Force suppliers by permitting them to concentrate on the specific avionics problem by viewing the core elements as a service facility interconnecting the application software with the sensor/subsystem hardware. The general objectives of the DAIS architectural design are listed in Table 1 along with the design considerations or attributes to help satisfy the objectives. These attributes, such as standardization, redundancy, modularity, on-board test, etc., have been designed into the system at the start as opposed to incorporation at a later stage.

3.1 System Configuration

The DAIS configuration is shown in Figure 2. The MIL-STD-1553B data bus is the central element of the configuration; it provides the communication link between the DAIS processors, Controls and Displays, and aircraft subsystems. The basic core elements or building blocks of DAIS are the DAIS multiplex system (dual bus and remote terminals) DAIS processor/BCMs, DAIS Mission Software, and DAIS Controls and Displays.

The AN/AYK-15A processors include the bus interface/controller function and normally reference is made to a "Processor/BCM" as a functional unit. The Remote Terminals (RT's) receive various analog, digital, and discrete signals from the sensors/subsystems and format these for bus transmission. The RT is designed to accommodate various types of interface modules to provide the proper electrical interface with different sensors. It is programmable to permit mapping of the data between the data bus and the sensors as required for

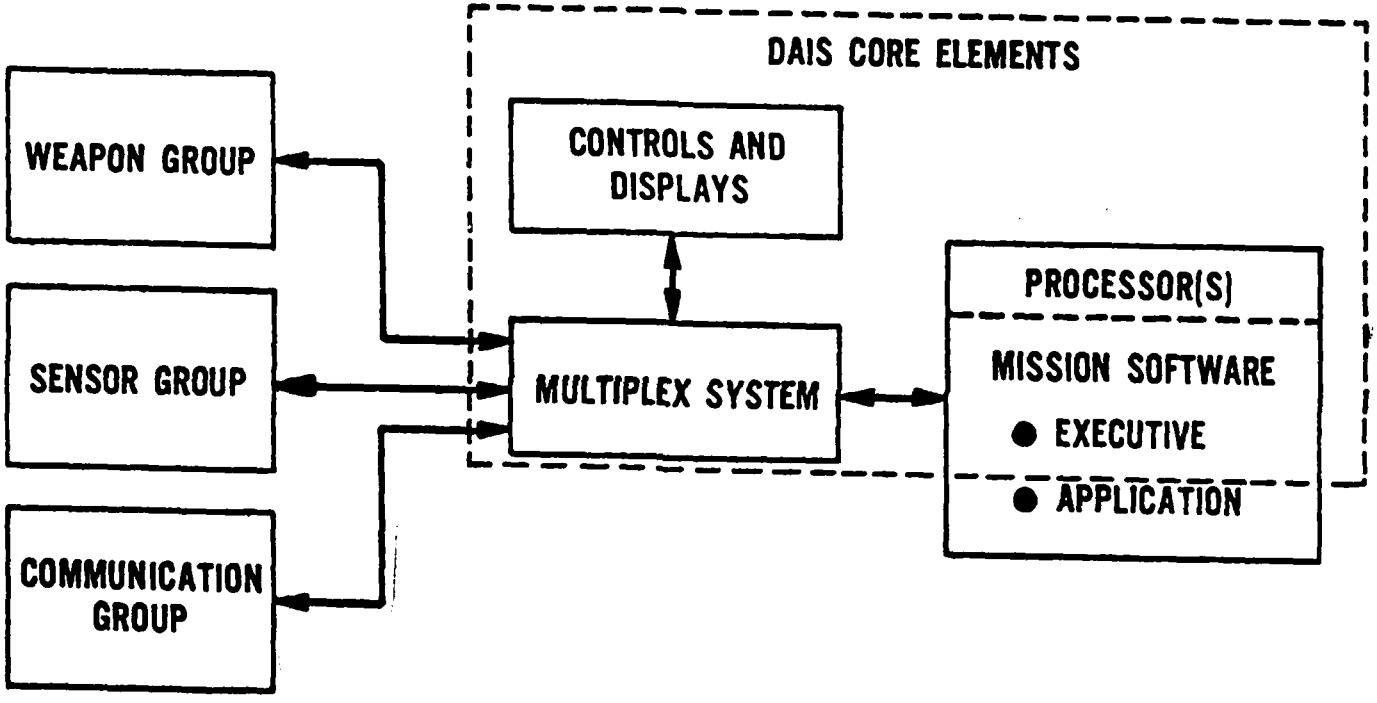


FIGURE 1. DAIS FUNCTIONAL ARCHITECTURE

TABLE 1. DESIGN CONSIDERATIONS UTILIZED IN MEETING DAIS OBJECTIVES

DAIS Objectives	Design Considerations	Cost Impact
Reduce Unnecessary Development Proliferation	• Replication (Core Elements) • Standardization	• Development Cost • Unit Production Cost • Installation Cost
Improve Operational Efficiency and Availability/Maintainability	• Automated Aids • Digital Technology • Redundancy Utilization • On-board Test	• Training Cost • Flight Cost • Reduces Mission Aborts • Maintenance Cost
Improve Flexibility to Changes	• Functional Modularity • System Modularity	• Maintenance Cost • Modification Costs
Maintain Basic Avionics Performance	• Proven Technology	• Development Cost

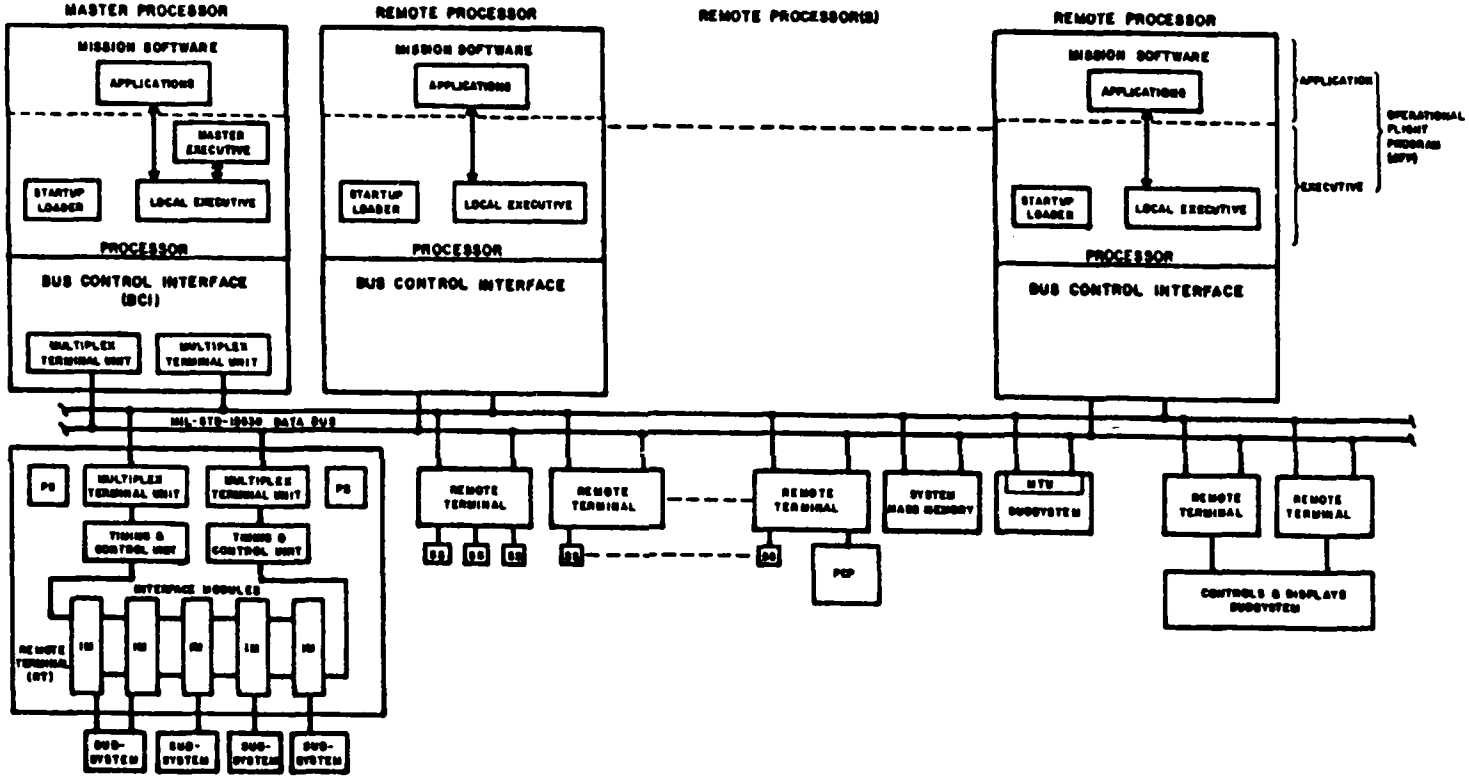


FIGURE 2. DAIS SYSTEM ARCHITECTURE

the specific avionic system configuration. Sensors, control and display equipment, or other subsystems, may interface directly to the bus, or may interface via the Interface Modules of a Remote Terminal (RT). This flexibility is provided within the communications protocol of the MIL-STD-1553B multiplex bus.

In normal system operation, one processor/BCM is designated as master and is in control of the system. All bus commands are issued by this unit under the direction of the Master Executive software resident in the processor memory. The remaining processor/BCMs are designated as remotes and contain mission application software appropriate for the configuration. All processors also contain software known as the Local Executive providing basic real-time task management and utility functions.

The Processor Control Panel (PCP) is interfaced to the multiplex bus via an RT and provides the pilot control over what processors are active in a configuration. The system mass memory is actually a simulation of a bubble memory device with an embedded bus interface. Two independent, fully redundant RTs are used to interface the C&D system to assure availability of this critical pilot interface. Other RTs/Subsystems are added to the configuration as required for a particular mission demonstration. The RTs may also be simulated in the Universal Remote Terminal (URT).

It should be clear that while this specific configuration was implemented in the ITB, the DAIS architecture is applicable to a wide range of configurations. For illustration, a generalized DAIS configuration is shown in Figure 3. The main point is that DAIS provides a set of application-independent core elements. For a specific aircraft application, the system designer is at liberty to configure these building blocks to the specific hardware configuration required. The software can also be partitioned based upon avionic subsystems and mission requirements such as reliability and availability. The support software, J73 Compiler and PALEFAC, provide the system designer and application programmer the tools to develop, test, and partition the application software and automatically generate the executive tables for the specific aircraft configuration.

The number of processors can be reduced or enlarged depending upon the avionics and mission requirements. Application-independent executive software makes the DAIS architecture flexible and adaptable. It can be applied to a broad class of applications, and mission-to-mission changes in a particular aircraft are also easily accommodated. Sensors, weapons, and other subsystems are selected as required for the particular mission. Application modules of the software are selected or developed as required by the subsystems. This basic DAIS architecture reduces unnecessary development duplication of similar tasks every time new systems are developed.

3.2 DAIS Core Elements

The basic elements of the DAIS architecture, which are independent of the aircraft and mission, are known as the core elements and consist of:

DAIS Multiplex (dual bus and RTs)

DAIS Processor/BCI

DAIS Controls and Displays

DAIS Executive Software

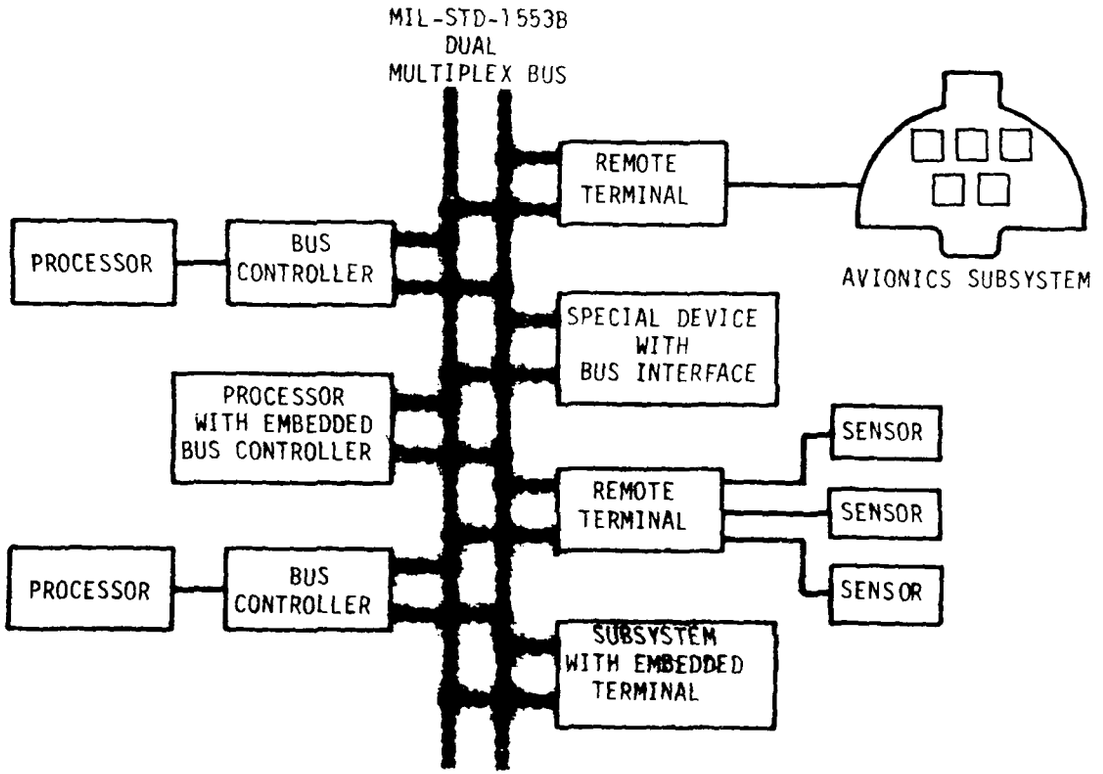


FIGURE 3. GENERALIZED DAIS CONFIGURATION

The system designer must determine the number of processor/BCMs required, but otherwise fully defines system functionality solely with applications software and the suite of avionics sensors/subsystems configured.

The core elements are described in the following sections.

3.2.1 DAIS Multiplex

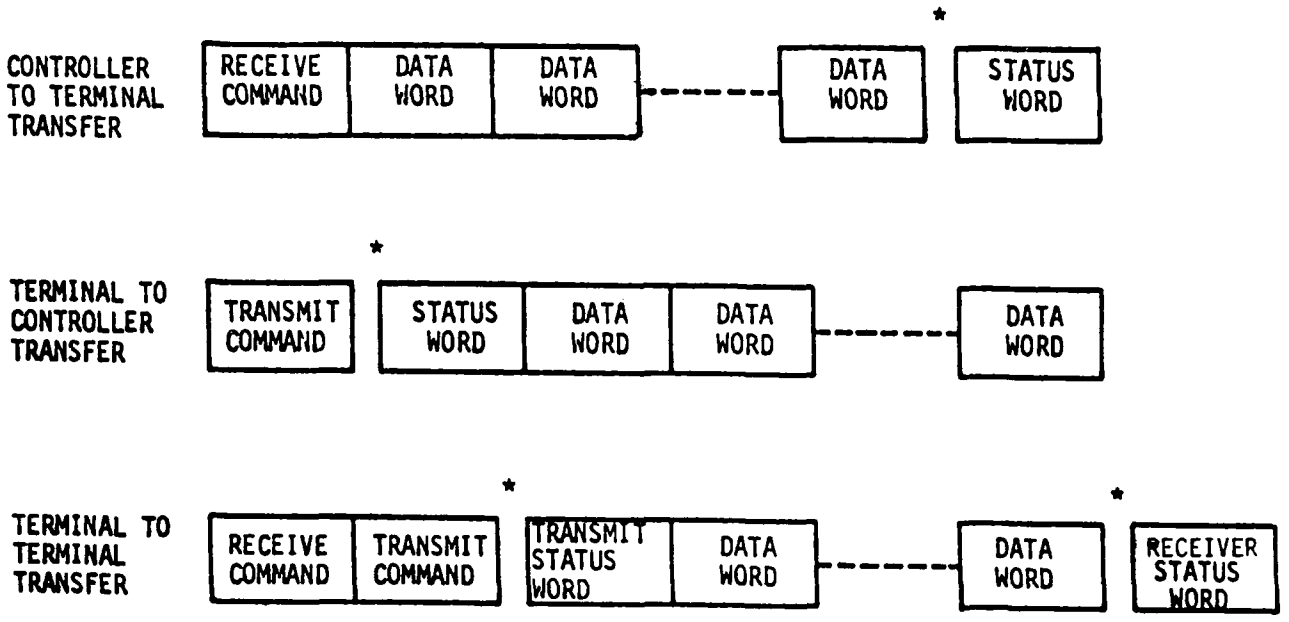
The DAIS multiplex provides a channel for information transfer between the elements within the system. It consists of the Bus Controller Interface Unit (BCIU) or integrated processor/bus controller (AN/AYK-15A). Remote Terminal units (RT), and the multiplex cable assembly (data bus). The message structure, data format, and command-respond protocol is as defined in MIL-STD-1553B. The message format consists of three types of operations as shown in Figure 4 with word formats as shown in Figure 5. The communications protocol is a time division multiplex (TDM), command-response approach with one processor/BCIU controlling the bus traffic at any given time. Each multiplex cable assembly consists of a twisted, shielded wire pair and bus couplers.

Message protocol is the sequence of the bus messages required to perform system and bus control functions (synchronous and asynchronous operations) and error and failure management operations. A set of system control procedures was defined to describe the procedures to perform these operations as summarized in paragraph 4.1. The overall message sequencing is shown in Figure 6.

a. Bus Controller Interface Unit (BCIU). The BCIU provides the interface, control, and data transfer functions between a processor and two data buses. The BCIU operates under control of the Executive software in either a remote mode or master mode.

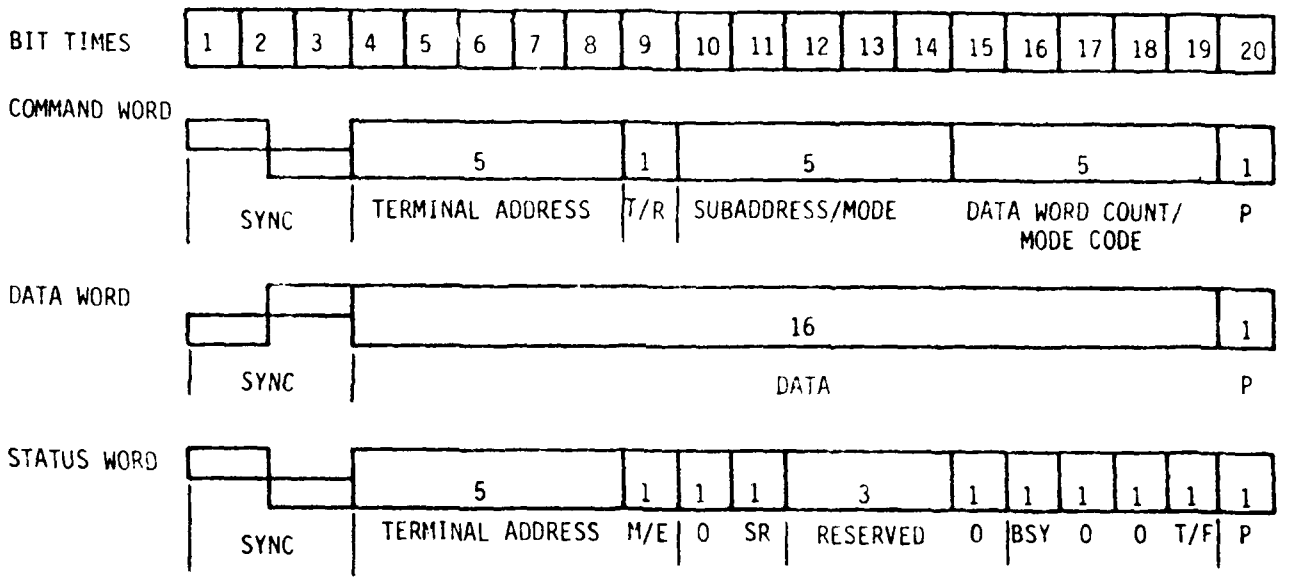
In the remote mode, the BCIU provides the transfer of data to and from the processor based upon commands received from either of the data buses. It provides the status reply on the appropriate data bus, performs special internal operations, and presents interrupts to the processor for a few specified conditions.

In the master mode, the BCIU constructs and issues the bus commands based on a two-word instruction fetched from the processor memory. These instruction words contain not only transmitting-receiving terminal addresses, subaddresses, and word count fields, but also the fields to dictate bus selection, automatic retry options, and BCIU internal operation codes. The BCIU sequentially interprets each instruction pair to determine the action required. Depending on the op-code contained in the instruction pair, the BCIU initiates a bus operation, performs a no-op, or performs a link operation. If a bus operation is indicated, the BCIU controls the message transmission on the data bus. It accesses the next sequential instruction pair when the present operation is completed successfully. If a no-op is indicated, the BCIU accesses the next requested instruction pair. If a link operation is indicated, the BCIU performs no bus operation, but uses the second word of the instruction as the address of the next instruction pair.



*Intermessage Gap Time

FIGURE 4. MIL-STD-1553B MESSAGE FORMATS



NOTE: T/R - TRANSMIT/RECEIVE
P - PARITY
M/E - MESSAGE ERROR
SR - SERVICE REQUEST
BSY - BUSY
T/F - TERMINAL FAIL
0 - ZERO. THESE BITS ARE DEFINED IN MIL-STD-1553B, BUT ARE NOT USED IN DAIS AND ARE ALWAYS ZERO.

FIGURE 5. MIL-STD-1553B WORD FORMATS

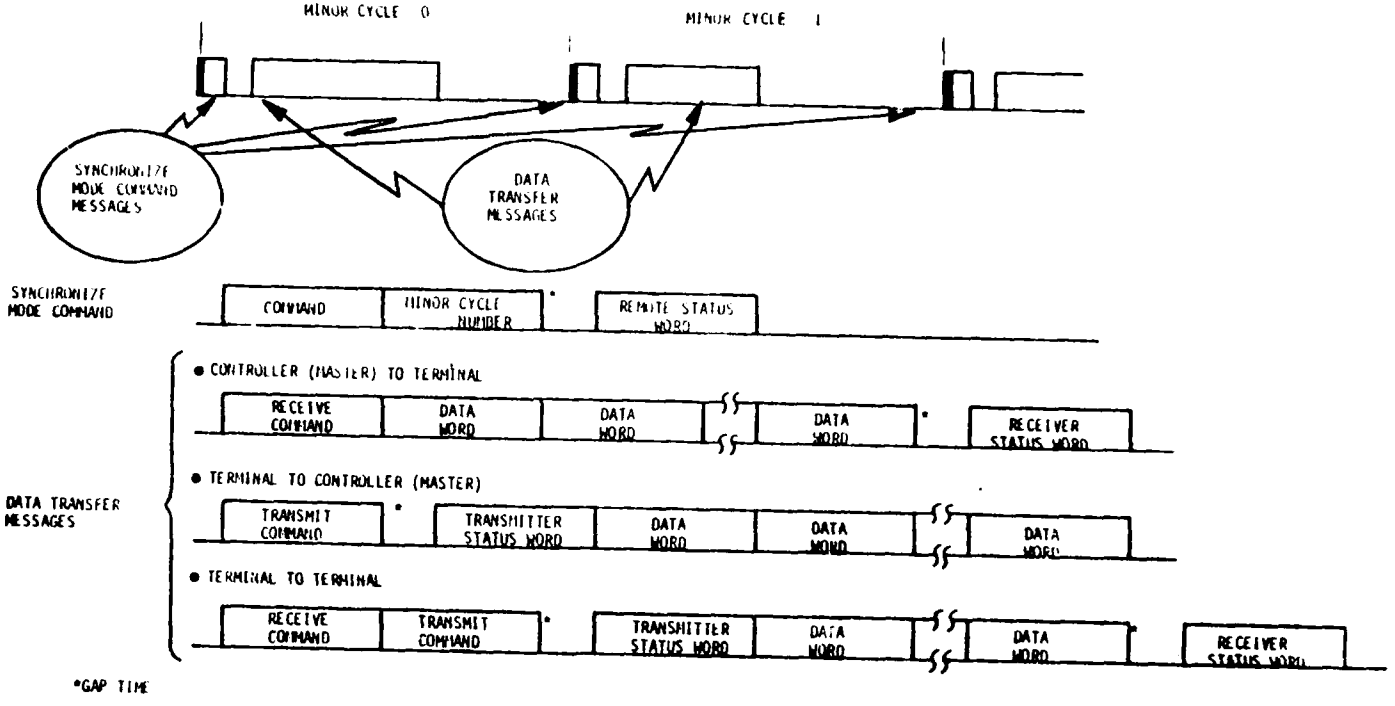


FIGURE 6. MIL-STD-1553B MESSAGE SEQUENCING

Figure 7 identifies the major components that comprise a BCIU. Each Multiplex Terminal Unit (MTU) provides the interface function to one data bus. The Processor Interface Module (PIM) provides the interface function to the processor. The changing of processor types is accommodated by a re-design of only the PIM. The Bus Control Module (BCM) provides the timing, control, instruction decoding, and data transfer routing required to implement the various operations. The BCIU also performs mode code operations to obtain status or built-in-test information for error and failure analysis or to perform special mode operations as defined in paragraph 4.1.2.2.

b. Bus Control Interface Module (BCI). The BCI is embedded in the AN/AYK-15A processor and operates in either of two modes: master or remote. The BCI performs the same duties as the BCIU but provides enhanced status word interpretation and interrupt code generation. Figure 8 shows the major components of the AN/AYK-15A processor/bus controller.

c. Remote Terminal (RT). The Remote Terminal provides the interface between the subsystems and the two data buses. The RT transfers data in both directions between the multiplex bus and subsystem via the Interface Modules. It receives these commands from either of the data buses and provides the status reply on the appropriate bus. The RT performs special mode command operations upon receipt of a mode command.

The RT is modular and programmable. This allows flexible partitioning of the data messages to the appropriate subsystems/sensors and signal conditioning required by different numbers and mixes of subsystem and signals. The RT is composed of Multiplex Terminal Unit (MTU), Timing and Control Unit (TCU), and Interface Modules (IM) as shown in Figure 9. The Multiplex Terminal Unit (MTU) interfaces to the data bus. It transmits and receives information between the data bus and the Timing and Control Unit (TCU) under control of the TCU. The RT is dual redundant with respect to the MTU, TCU, and power supplies.

The Timing and Control Unit (TCU) performs all of the timing, control, buffering, decoding, and checking required to receive or transmit information on the data bus. It transfers that information as outputs or inputs from the RT via the Interface Modules (IM). The TCU contains a programmable device which controls the mapping of each data word in each message to the proper subsystem interface, i.e., the specific IM slot in the RT and channel on that IM. The interface between the TCU and IM is standardized and contains the signals required to allow the TCU to select the individual modules. Therefore, all IM slots accept all IM types.

Each RT will interface with different numbers and mixes of subsystems/sensors signals for each specific system configuration. This is accomplished by inserting the proper number and type of interface modules into the IM slots in the RT housing and by programming the ROM in the TCU. Table 2 lists the typical type of RT Interface Modules provided to interface with the subsystems. If required, a special interface module can be designed to interface with any existing subsystem.

The MTU and TCU functions of the RT can be embedded in a sensor or subsystem so that the interface of the sensor or subsystem is directly with

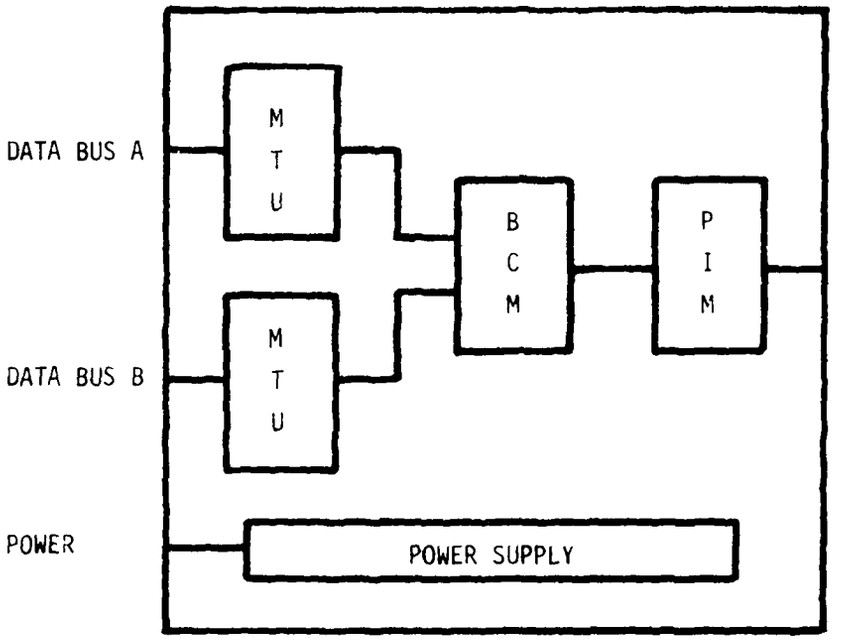


FIGURE 7. BUS CONTROL INTERFACE UNIT

CPU - Central Processing Unit
DMA - Direct Memory Access
EAU - Extended Arithmetic Unit
MTU - Multiplex Terminal Unit
MUX - Multiplex (1553B)

PIO - Parallel Input/Output
PMI - Performance Monitoring Interface
ROM - Read Only Memory
RS-232 - ANSI RS-232 Serial Interface

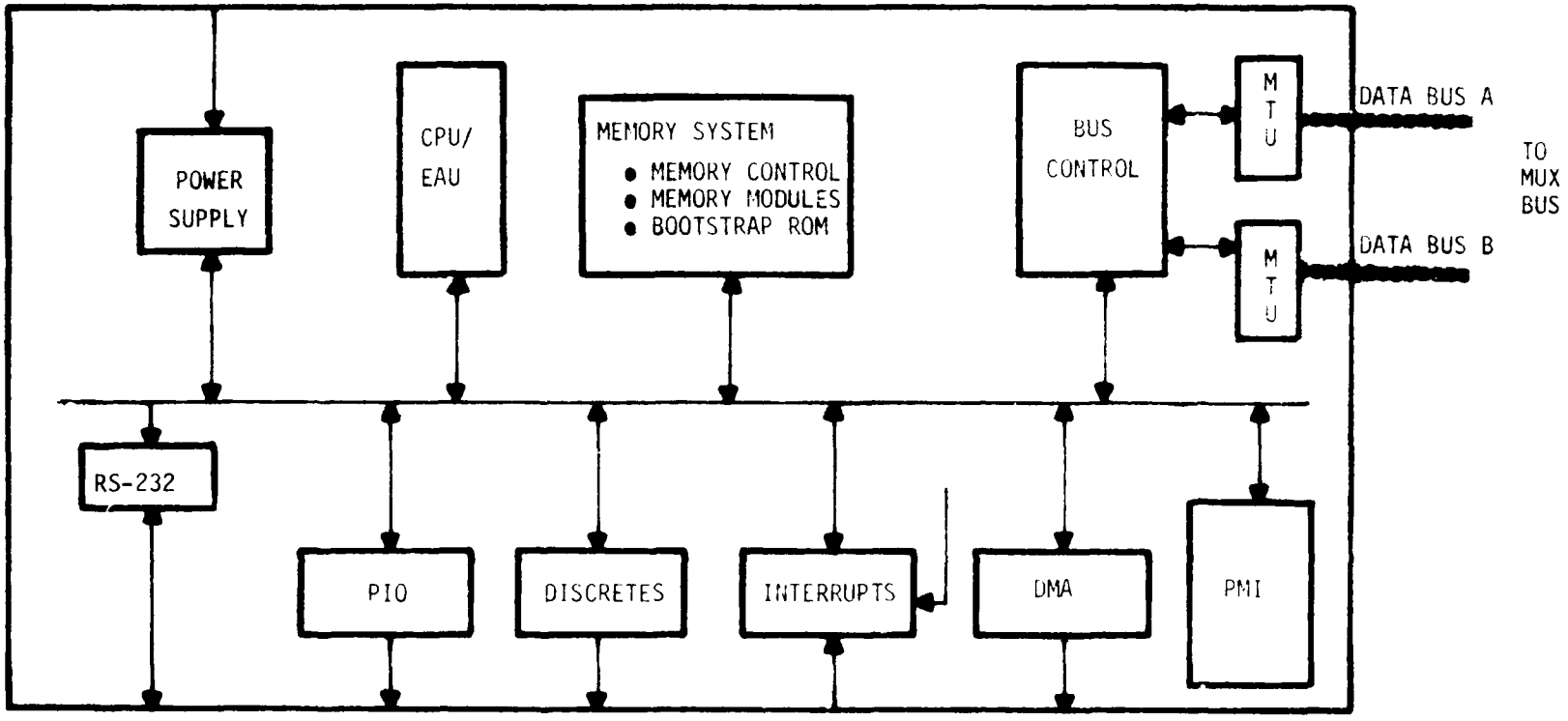


FIGURE 8. DAIS PROCESSOR/BUS CONTROLLER (AN/AYK-15A)

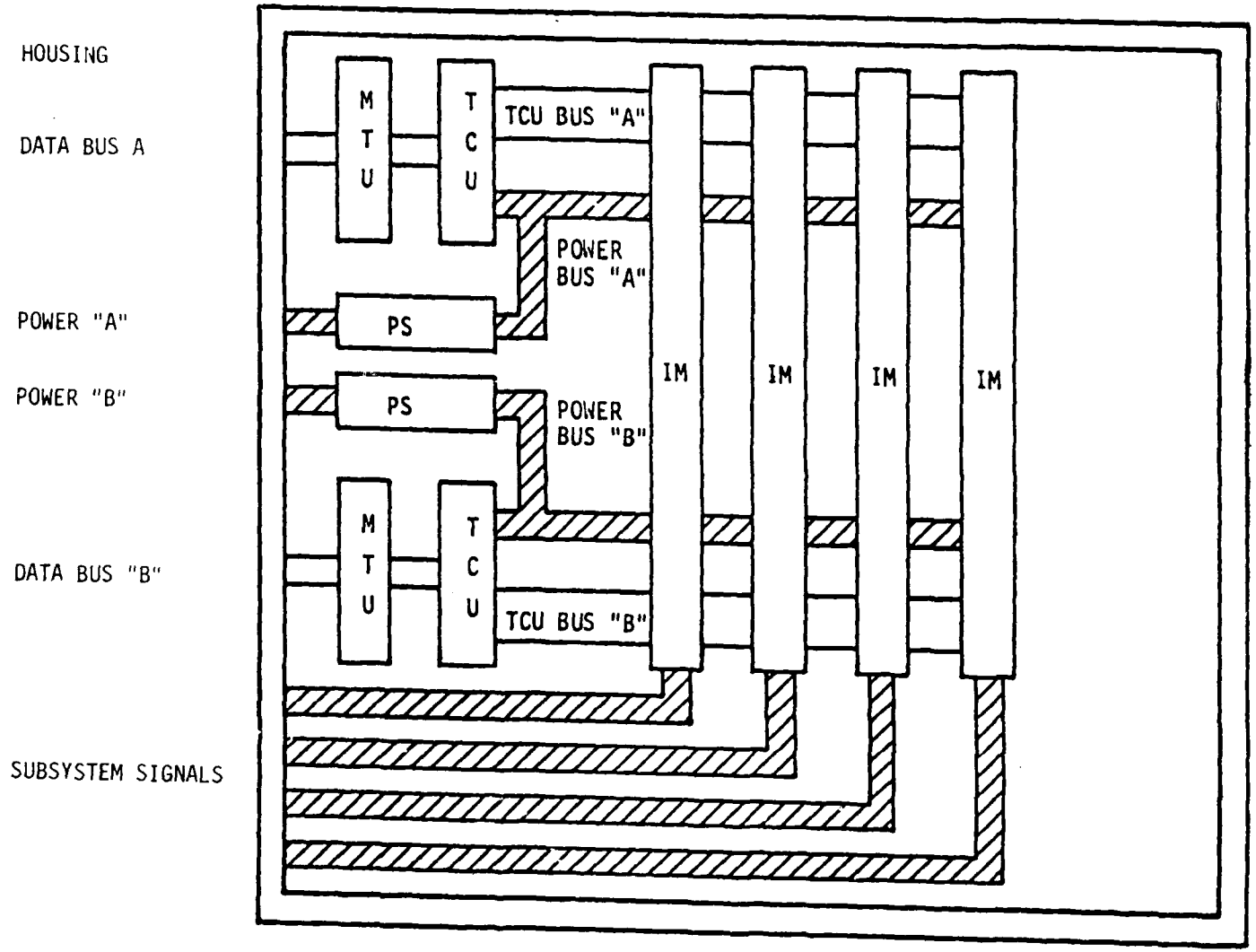


FIGURE 9. REMOTE TERMINAL UNIT

TABLE 2. RT INTERFACE MODULES

IM Type	Channels Per IM	Inputs/Outputs Per Channel	Total Inputs/Outputs Per IM
Differential Discrete Input Module	1	16	16
Differential Discrete Output Module	1	16	16
Single Ended Discrete Input Module	2	16	32
Single Ended Discrete Output Module	2	16	32
Momentary Closure Input Module	1	16	16
Switch Closure Discrete Input Module	1	16	16
Switch Closure Discrete Output Module	1	16	16
A.C. Analog Input Module	8	1	8
A.C. Analog Output Module	8	1	8
D.C. Analog Input Module	8	1	8
D.C. Analog Output Module	4	1	4
Facility Input/Output Module	1	1	1
Synchro Input Module	8	1	8
Synchro Output Module	8	1	8
Serial Digital Input	4	Up to 32 words	
Serial Digital Output	4	Up to 32 words	
Pulse Counter-Torques Module	1	3	3

the multiplex data bus. Functionally, the embedded RT responds to commands received on either data buses in the same way as an RT.

3.2.2 DAIS Processors

The AN/AYK-15 DAIS processors provide computational, control, memory, and input/output capabilities. The DAIS processors are distributed in the architecture and are interconnected through the DAIS multiplex system. In this architecture, the processors address only their own memory units.

The DAIS processors include the following features:

- Central Processor Unit (CPU)
  - 16 general registers
  - Extensive instruction repertoire
  - 379K operations per second throughput based upon a specified benchmark program
  - Floating point capability
  - Direct, indirect, immediate and index addressing modes
  - Programmable interval timers
  - Vectored interrupts
- Memory
  - Expandable random access memory to 65K words (16-bit words), in 16K word memory modules
  - Memory parity and write protect features
- Input and Output
  - Discretes
  - Program controlled
  - Direct Memory Access (DMA)
  - External interrupts

The AN/AYK-15A integrated DAIS processor/bus controller combines the processor and BCIU functions into an integrated unit and incorporates the MIL-STD-1750 instruction set. Figure 8 describes the AN/AYK-15A. This integrated DAIS processor/bus controller includes the following features:

- Processor
  - 16 general registers
  - Extensive instruction repertoire as specified in MIL-STD-1750
  - 400K operations per second throughput based upon specified benchmark program
  - Floating point capability
  - Direct, indirect, immediate, base relative, IC relative, and index addressing modes
  - Programmable interval timers
  - Vectored interrupts
- Memory
  - Expandable random access memory to 65K words (16 bits per word), in 16K word memory modules
  - Memory parity and write protect features
- Input and Output
  - Discretes
  - Program controlled
  - Direct memory access
  - External, encoded interrupts
  - Multiplex data bus
  - Bus controller

3.2.3 DAIS Controls and Displays

The DAIS Controls and Displays (C&D) provide the interface between the pilot and the various avionics. The C&D consists of a set of shared multipurpose data entry devices, control devices, and display devices. The C&D concept is based upon an integrated set of common and shared devices which can be used for most of the avionic functions. Thus, the DAIS Controls and Displays concept embodies the following features:

- Flexibility to accommodate changes in avionic functions and moding as a result of new sensors/subsystems or mission changes without requiring a physical change in the control and display devices.
- Efficient design for centralized system management by the pilot as required for the system application.

- Redundancy if required for specific system application (e.g. CRT's serve as backup to each other, redundant display generators) for backup of mission-critical functions.

- Provisions for mode control and command data from data entry devices and to display devices to be transmitted and received over the multiplex bus to or from mission software.

The set of Control and Display equipment developed for the DAIS demonstration is representative of what one would utilize in the DAIS architecture. An example of the Controls and Displays subsystem configuration is shown in Figure 10. The units or building blocks of the DAIS controls and displays include the following.

a. Modular Programmable Display Generator (MPDG). The MPDG is a programmable graphic display generator. It is capable of being programmed to generate graphics composed of predefined symbols, line segments, and alphanumeric symbols. The MPDG generates symbols for presentation on both raster and stroke written display devices. The MPDG generates the display data for up to four raster display devices via the Display Switch/Memory Unit (DS/MU) which contains the refresh memory for the raster displays. The MPDG also generates stroke (x and y deflection signals, and blank/unblank signals) for one stroke display.

The MPDG generates various forms of symbology including geographic map symbols, overlay symbols, tactical horizontal and vertical situation data, and flight situation and director symbols as programmed in the C&D subsystem application software. This application software executes on the programmable processor in the MPDG.

b. C&D Subsystem Application Software. The C&D subsystem application software consists of the routines and display lists for the display generator in the MPDG. This software generates the display symbology, text formats and pages, map data, etc., required for a specific avionics system application. Mode control and data messages received from mission software via the multiplex system controls the display modes, display assignments, text displays, and variable symbology. The application software is commanded by mission software. It performs the loading of the MPDG programs, text pages, and map data from the C&D Mass Memory Unit (MMU).

The specific application software was developed using an MPDG assembler hosted on a DECsystem-10. The source language consists of the MPDG microprocessor instructions, display generator command formats, and assembler directives.

c. Display Switch/Memory Unit (DS/MU). The DS/MU is used in conjunction with one or two MPDGs to drive the raster display devices. The DS/MU accepts digital data from one or two MPDGs, stores the data in up to six refresh memories, and sends the data to up to four raster displays (525-line or 875-line format).

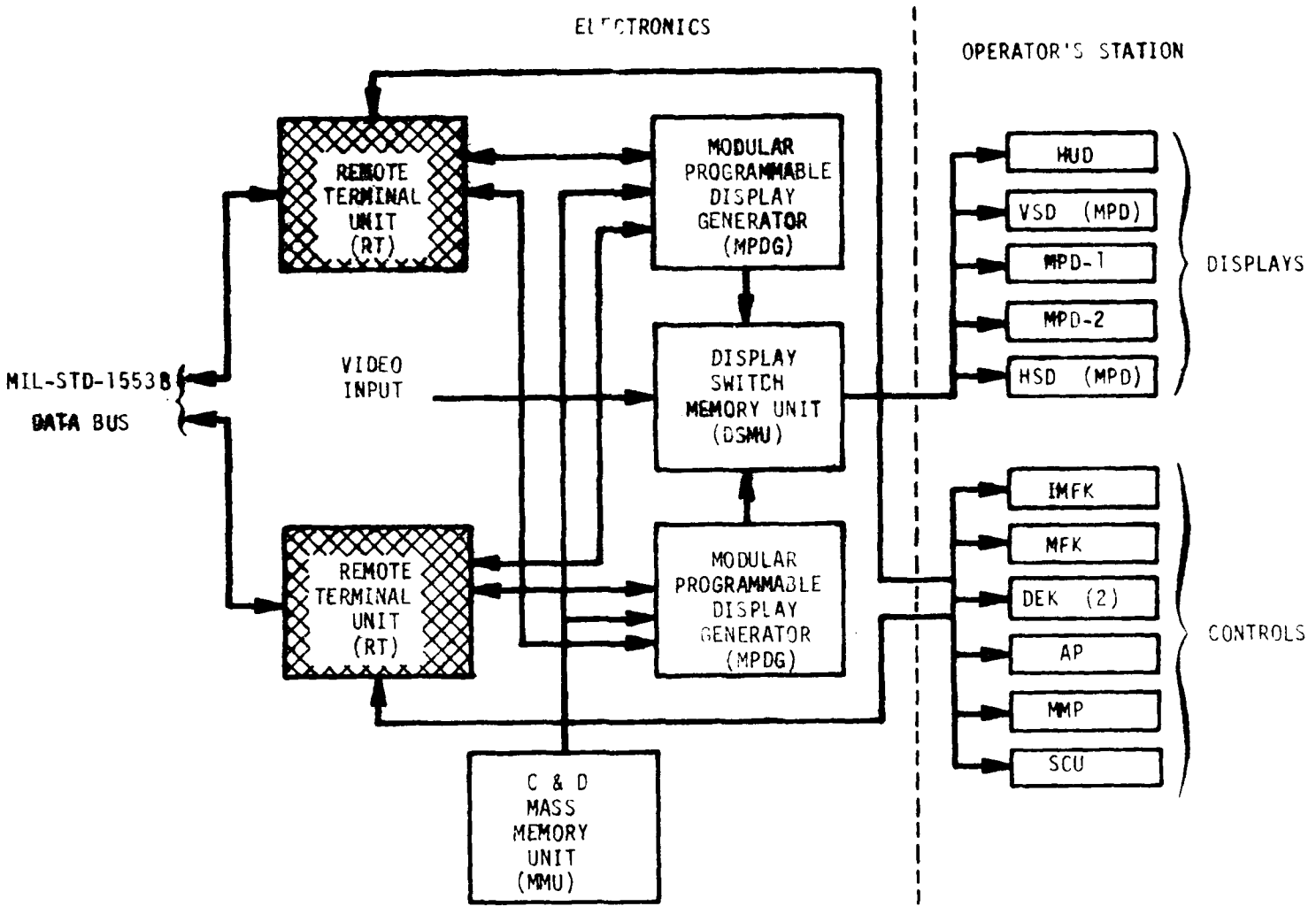


FIGURE 10. CONTROL AND DISPLAY

The raster displays are displayed alone or overlayed with sensor video. The DS/MU provides the capability to route signals from up to 14 sensor or weapon video sources (e.g. TV sensor - 525 line, FLIR sensor - 875 line, weapon station sensors - 525 line) to any of the four raster displays. One external video signal is routed to a single display at one time.

The DS/MU also accepts stroke-generated signals from one or two MPDGs and routes these signals to one of the two stroke displays. Although the DS/MU is a serial element in the system, it is designed to minimize the probability that a single failure will cause complete loss of display capability.

d. Display Devices. The display devices are cathode-ray-tube (CRT) displays which are driven by the MPDG and DSMU as described above. This allows changes or additions to the information displayed to the pilot without physical changes to or movement of the display devices in the cockpit. The CRT display devices consist of raster-written and stroke-written devices as follows:

- Head-Up Display (HUD) - stroke-written
- Multiple Purpose Displays (MPDs) - raster-written

The MPDs can be used for Vertical Situation Display (VSD), Horizontal Situation Display (HSD) and Multi-Purpose Displays (MPDs) as required by the specific avionics systems configuration. All the MPDs are identical except for the legends on the display selection buttons. The Head-Up Display (HUD) presents the symbology (e.g. cues, flight information) in the pilot's forward field-of-view, so out-the-window scenes (e.g. targets, IPs, etc.) can be viewed.

e. Data Entry Devices. The data entry devices provide flexibility in the information which can be entered by the pilot to control and manage the specific avionics system (mission functions, sensors, and subsystems). The design of the data entry devices allows changes or modifications to the avionics functions without physical changes to the cockpit. The output of the data entry device is received and interpreted by Mission Software via the multiplex system. Display responses to the device are generated by mission software via the multiplex system. The type of data entry devices include the following:

1. Integrated Multifunction Keyboard (IMFK). The IMFK consists of a programmable alphanumeric 280 character display, seven master function pushbuttons, and ten submode select pushbutton switches, five on either side of the display. The IMFK display is capable of displaying ten rows of characters divided into three columns (nine characters in left and right columns, and ten characters in center column). The display information is generated by mission software. The pushbuttons generate a request via the remote terminal for the master executive to asynchronously service and transmit the data to the appropriate processor. Application software interprets the pushbutton or switch action and provides the appropriate response.

2. Multifunction Keyboard (MFK). The MFK consists of seven master function pushbuttons, ten submode pushbutton switches, and ten indicator lights. The MFK interfaces with mission software via the RT in the same way as the IMFK. It can be utilized for specific control and display functions or as a backup for the IMFK.

3. Data Entry Keyboard (DEK). The DEK provides the pilot with a means of entering numeric and limited alpha data into mission software via the multiplex data bus. The DEK stores and displays up to ten characters. When the enter key is depressed, a request for service via the RT is made to the master executive. Upon receiving the data, Mission Software interprets the data depending upon the selected submode via the IMFK or MFK.

f. Control Devices. The Controls and Displays subsystem also provides several control devices as follows:

1. Sensor Control Unit (SCU). The SCU provides for selection and position control of a symbol or sensor (X and Y directions). The hand controller is a force control with DC voltage output which is proportional to the force in the X and Y directions. The SCU also provides trigger and switch discrete outputs. These outputs are transmitted to mission software via the multiplex data bus; mission software drives the appropriate symbol or sensor based upon the selected mode. The SCU also provides a control knob to control cursor brightness.

2. Armament Panel (AP). The armament panel provides switches for weapon armament control (bombing, fusing, gun firing, and jettison).

3. Master Mode Panel (MMP). The MMP consists of fifteen master mode switches. Each has an indicator light and HUD-related switches and controls. When a master mode switch is depressed, a request for service will be made to the master executive. Upon receiving the data, mission software interprets and responds to the selected master mode.

The interface between the pilot and the integrated set of Controls and Displays is dependent upon the specific avionic mission and system configuration. The system designer selects the set of integrated Controls and Displays required for the specific avionic system and defines the pilot sequences and display responses required to control and mode the avionics system.

The interface between the Mission Software and the C&D application software in the MPDG is also mission-and application-dependent. This interface is defined by the data bus messages from Mission Software to the MPDG to perform the following functions:

- Load the MPDG
- Control display (VSD, MPD, HAD, & HUD) assignment
- Control display modes
- Provide data to drive the display symbology, e.g. elevation, altitude, heading, etc.
- Provide text data

With increased avionic and mission complexity, the role of the pilot is changing to that of a total system manager. Mission Software can perform the systems management function, collecting information and controlling a functionally integrated set of subsystems and sensors, to reduce the pilot workload. Functions such as subsystem checkout and monitoring, data computation and reduction, routine sequencing within mission modes, and mission and aircraft data storage can be performed by the mission software with minimum pilot intervention. Operation by exception can be designed into the system control and operation.

For example, the pilot can use the integrated Controls and Displays to mode the avionics system through the Mission Software applications functions. He receives display information about the mission and avionics system operations from the software. The control devices permit the pilot to establish the operations to be performed by the software functions. The master modes set the display devices to prescribed formats via Mission Software and the MPDG. The pilot, however, can override these normal operations to either cause alternate modes or activate operations not required by the selected master mode.

Master modes are defined for specific phases of the mission where certain normal operations are active. Upon entering a new master mode, some previous operations are terminated as new ones are activated. Manual modes previously selected by the pilot can be retained if compatible with the new master mode. If the new master mode requires a system that has been "deselected" using the manual mode, the pilot is alerted through display exception messages. The Master Mode Panel, as labeled for the specific mission applications, permits the pilot to select these prescribed master modes. Mission software, in response to the selected master mode, will command the C&D application software to generate formats for the various displays (HUD, VSD, MPD, etc.). The pilot can override the preprogrammed mission functions through the IMFK-displayed pages and side keys which are also handled by mission software. The pilot will also be able to enter information into Mission Software via the DEK. For the DAIS demonstration, a representative set of master modes and mission operational sequences were defined and implemented under control of mission C&D application software. The requirements for these modes are defined in a Pilot/Operational Sequence Interface Control Document (ICD).

3.2.4 DAIS Mission Software

The DAIS Mission Software (MSW) is the set of programs which support and perform the avionics functions required for the DAIS Close Air Support (CAS) mission. To control and perform these functions, the MSW contains both Executive and Application Software.

3.2.4.1 Executive

The executive is the operating system for MSW. It is organized into the local executive and the master executive.

3.2.4.1.1 Local Executive

Each DAIS processor contains a local executive. The local executive software is table-driven. It performs the following functions:

1. Process Control - provides services for the application software to activate and deactivate periodic and non-periodic tasks when appropriate conditions have been met. These conditions are based upon logical settings (on or off) of real-time events.

2. Event Control - provides the mechanism for setting, resetting, and evaluating real-time events which communicate conditions (on or off) signalled between tasks in the same or different processors.

3. Data Control - guarantees integrity of shared data and provides mechanism for transmission and reception of data over the multiplex bus.

4. Processor Initialization - provides initialization for processor power transient recovery, initializes program loads, and performs minor cycle synchronization with the other processors.

3.2.4.1.2 Master Executive

One of the DAIS processors contains the master executive. The master executive is table-driven. It manages the system configuration by performing the following functions:

1. System Synchronization Control - allocates time segments (minor cycle and major cycle) for synchronous messages and performs minor cycle synchronization by transmitting minor cycle master function mode commands to the other processors.

2. Bus Control - controls transmission of synchronous and asynchronous messages over the multiplex data bus.

3. System Error/Failure Management - monitors and analyzes errors and failures on both bus and terminal devices. Initiates message retry procedures to recover from message errors.

4. Configuration Management - detects and isolates permanent device failures, maintains system configuration status, reports failure to the application software, and initiates backup or recovery operation if required.

5. Mass Memory Management - provides for retrieving and storing information from the mass memory on request.

3.2.4.2 Application Software

The application software performs the tasks associated with the avionics requirements of the mission. The application software is divided into eleven functional areas.

- Navigation
- Guidance
- Weapon Delivery
- Position Update
- Display
- Miscellaneous Functions
- Stores Management
- Pilot Control
- System Control
- Common Subroutines
- System Functions

Table 3 lists and briefly describes all the programs in each of the functional areas.

3.2.4.2.1 Navigation

The navigation software computes aircraft position and velocity and related information used by the pilot and by avionics algorithms. The Navigation programs include three navigation modes (INS, Dead Reckoning, and TACAN), associated programs, and programs to interface with the appropriate avionic sensors.

3.2.4.2.2 Guidance

The guidance software computes flight path deviations and handles pilot requests. It calculates and displays to the pilot information needed for steering. The guidance software provides the six steering modes: Command Track, Command Nav, Command Heading, Command Altitude, ILS, and TACAN.

3.2.4.2.3 Weapon Delivery

The weapon delivery programs perform the calculations to aid the pilot in releasing weapons. Variables such as ejection angles, range-to-target, miss distance, stick length, and bomb-fall-line are computed and displayed. Two delivery modes, CCIP/Auto and CCIP/Manual, are available.

3.2.4.2.4 Position Update

The position update programs in conjunction with the weapon delivery range algorithms allow the pilot to update the aircraft position based on known waypoint or target positions. Three position update modes are available: INU HUD update, Flyover update, and FLIR/Laser update.

3.2.4.2.5 Display

The display software presents information to the pilot in the cockpit. It controls data sent to the MPDG's and the IMFK and MFK. It provides specific display messages to drive the symbols and graphics on the DAIS displays.

TABLE 3. APPLICATION SOFTWARE PROGRAMS

FUNCTIONAL AREA	APPLICATION SOFTWARE	DESCRIPTION
Navigation	SPØ1NAVCONTROL	Navigation controller
	SPØ2NAVPARMS	Calculates flight parameters from INS data
	SPØ3NAVINS	Interfaces with INS and computes wander angle, latitude, inertial velocity
	SPØ4TACANNAV	Calculates latitude, longitude, velocities using TACAN
	SPØ5DRNAV	Implements Dead Reckoning navigation
	SPØ6ALIGN	Controls timing of INS alignment
	SPØ7NAVDATA	Computes transformation matrix
	SPØ8TACSEL	Selects TACAN station to be used in TACAN Area Navigation
	QPØ1INSOUTMODE	Outputs mode and power to INS
	QPØ2INSOUTDATA	Outputs barometric altitude to INS
	QPØ3INSOUTUPDATE	Outputs direction cosines and longitude to INS
	QPØ4INSINDATA	Inputs INS data
GUIDANCE	SP1ØSTEERCONTROL	Guidance controller
	SP11STEER	Main task of steering function
	SP12DESTINATION	Changes command nav steering destination
	QP15TCNOUT	Outputs mode and channel select to TACAN
	QP16TCNIN	Inputs TACAN range, bearing, range rate, status
	QP2ØILSOUT	Outputs Power and frequency to ILS
	QP21ILSIN	Inputs glideslope and localizer deviation
WEAPON DELIVERY	SP4ØATTVEL	Calculates air mass velocity and ejection angles
	SP41WDCONT	Weapon Delivery controller
	SP42AIMING	Controls aiming reticle
	SP43FLR	FLR (Laser Ranger) pointing angles
	SP44BALLISTICS	Ballistics algorithm
	SP45WPNRANGE	Sets steering modes and in-range flags
	SP46SC	Solution cue position
	SP47BFL	Bomb fall line parameters
	SP48PD	Positions the pull-up cue
	SP5ØCCIP	Solution cue for CCIP manual
	SP51WDINIT	Initializes weapon delivery software

TABLE 3. APPLICATION SOFTWARE PROGRAMS (CON'T)

FUNCTIONAL AREA	APPLICATION SOFTWARE	DESCRIPTION
WEAPON DELIVERY (CON'T)	SP52RANGING	Computes target range
	SP53RELEASE	Time-to-release
	QP5ØLSRINRNG	Inputs range and status from Laser Ranger
	QP51LSROUTDATA	Outputs data to Laser Ranger
	QP55SCUIN	Converts SCU data into internal format
POSITION UPDATE	SP9ØPOSUPDATE	Coordinates pilot slewing the aiming reticle and designating to perform a position update
	SP91POSUPDATECONTROL	Position update controller
DISPLAY	DPØ1MMPLIGHTS	Changes MMP light state
	DPØ2IMFKLIGHTS	Lights the IMFK Brute Force keys
	DPØ3MFKLIGHTS	Lights the MFK Brute Force keys
	DPØ4PAGE	Displays entire IMFK/MFK page
	DP21MPDG	Synchronous DISP for messages to MPDG
	DP23MPDGDISPLAYASSIGN	Formats and sends data to MPDGs
	DP24MPDGWPTINIT	Initializes waypoints for MPDG
	DP25MPDGTABLDISP	Sends message to MPDG1
	DP3ØFLASHSYMBOLS	Controls flashing
	DP52STORESTAT	Changes MPDG stores option format
	DP6ØBLINKFD	Blinks the flight director
	DP51MAPUPDATE	Determines which map data blocks are needed
	QP71SCUDEKOUT	Output lamps, DEK activity, display power
MISCELLANEOUS FUNCTIONS	SP2ØADCOMP	Air Data computations
	SP21WIND	Computes wind velocity
	QP25RDRALTOUT	Outputs power and test bits to Radar Altimeter
	QP26RDRALTIN	Inputs Altitude and Status from Radar Altimeter
	QP4ØADIN	Inputs Air Data
	QP43UHFOUT	Outputs frequency and mode to UHF
	QP49GEARIN	Inputs aircraft status
STORES MANAGEMENT	SP7ØSTORESCON	Stores management controller
	SP71STORESCON1	Deactivates weapons
	SP72MK82CON	Controls MK-82
	SP76PJETT	Performs programmed jettison
	SP77EJETT	Performs emergency jettison
	SP78STORESINIT	Brings loaded stores to quiescent state
	QP6ØARMIN	Inputs armament switches
	QP8ØMK82	MK-82 control
	QP81STA1REL	Station 1 Release

TABLE 3. APPLICATION SOFTWARE PROGRAMS (CON'T)

FUNCTIONAL AREA	APPLICATION SOFTWARE	DESCRIPTION
PILOT CONTROL	CPØ3MMPREQPROC	Handles MMP Pilot requests
	CPØ4IMFKREQPROC	Interprets IMFK key inputs
	CPØ5MFKREQPROC	Interprets MFK key inputs
	CPØ7HANDLER	Carries out actions when IMFK side key is pressed
	QP7ØDEK1IN	Inputs data from DEK1 and DEK2
SYSTEM CONTROL	CPØ1MASTSEQ	Initializes application software
	CPØ2CONFIG	Sets up application software configuration
	CP11WOGSIMKEY	Weight-off-gear simulated key producer
	CP12FLYTOSIMKEY	Fly-to-point simulated key producer
	CP13WHLSIMKEY	Wheels down simulated key producer
	CP14FLAPSIMKEY	Flaps set simulated key producer
	CP15MMPILLSIMKEY	MMP illegal simulated key producer
	CP16MMPNIMPLSIMKEY	Non-implemented simulated key
	CP17WONGSIMKEY	Weight-on-gear simulated key producer
	CP18OFFFPSIMKEY	Off-flight-plan simulated key producer
	CP19TO2DIMWPTSIMKEY	To 2 dimensional waypoint simulated key producer
	CP2ØOUTBNDSIMKEY	Outbound intercept simulated key producer
	CP3ØWTOFFGEAR	Take-off/Climb simulated key producer
	CP4ØMMCOPY	Copies Master Mode
	CP5ØDISPCONTROL	Activates CPØ7
COMMON SUBROUTINES	ALTSET	Formats reference pressure for IMFK
	BINARY	Performs binary search
	CNTRST	Formats weapon station information
	CONVER	Converts radians to degrees and seconds
	DEGREE	Converts radians to degrees
	DENORM	Converts floating point to fixed point
	ELEVDH	Formats IMFK legend
	FLOATF	Converts floating point to ASCII
	FLYTOF	Formats the fly-to display
	FUZING	Formats the fuzing for display
	ILSCHN	Formats the ILS CHNG IMFK legend
	INTERV	Formats the interval for the IMFK

TABLE 3. APPLICATION SOFTWARE PROGRAMS (CON'T)

FUNCTIONAL AREA	APPLICATION SOFTWARE	DESCRIPTION
COMMON SUB-ROUTINES (CON'T)	JETPAG	Formats the jettison programs
	LATLON	Converts radians to character display
	LIMIT	Limits an integer to a specified range
	LIMITF	Limits a floating-point to a specified range
	MAGVAR	Formats the magnetic variation
	MODEFO	Formats the weapon delivery mode
	MTXV	Multiplies matrix transpose times a vector
	MXV	Multiplies matrix times vector
	MXM	Multiplies matrix by matrix
	NORMAL	Normalizes a fixed-point number
	OFFSET	Formats the offset for display
	QUANID	Formats the quantity/weapon ID
	REVERS	Reverse the bit pattern
	SMOOTH	Performs first-order lag smoothing
	STATIO	Formats the stations
	TCNCHN	Formats the TCNCHNG legend
	TRANSP	Performs matrix transpose
	S15ENC	Converts integer to ASCII
	UHFCHN	Formats the UHFCHNG data
	VADD	Adds two vectors
	VDOT	Performs dot product
	VUPDATE	Performs vector operation
	VXV	Multiplies two vectors
	WINDSE	Formats the windset display
SYSTEM FUNCTIONS	SIN	Sine
	COS	Cosine
	SIND	Sine (degrees)
	COSD	Cosine (degrees)
	ASIN	Arc sine
	ACOS	Arc cosine
	ATAN	Arc tangent
	SINH	Hyperbolic sine
	COSH	Hyperbolic cosine
	TANH	Hyperbolic tangent
	MAX	Maximum
	TAN	Tangent
	LN	Natural logarithm
	LOG	Logarithm
	SQRT	Square root
	EXP	Exponential
	ATAN2	Two-argument arc tangent

3.2.4.2.6 Miscellaneous Functions

The miscellaneous functions include wind and air data calculations. They also provide interfaces with the air data sensors, the radar altimeter, the UHF radio, and the gear.

3.2.4.2.7 Stores Management

The stores management software maintains inventories of the weapons on board the aircraft. It keeps track of the number and type of weapons as well as their state.

3.2.4.2.8 Pilot Control

The pilot control software responds to the pilot's requests and invokes the appropriate application functions that generate information for displays or for controlling avionic support subsystems.

3.2.4.2.9 System Control

The system control software schedules and controls the application software.

3.2.4.2.10 Common Subroutines

The common subroutines (COMSUB's) are a set of reentrant programs. They perform mathematical operations such as matrix multiplication as well as data manipulations such as reformatting.

3.2.4.2.11 System Functions

The system functions include the mathematical and trigonometric functions such as sine and square root.

3.2.4.3 Mission Software Architecture

The mission software is modular and flexible. It can accommodate mission-to-mission changes as well as changes in the complement of avionic sensors. It allows for modification of both the executive and the application software which greatly facilitates implementing the DAIS architecture on other aircraft.

The application software is organized in a hierarchical control tree structure. Each program is either a controller or a calculator. A controller schedules a particular set of calculators. Each controller is, in turn, scheduled by a controller at a higher level.

The master sequencer is at the top of the application software hierarchy. It controls the request processors, configurator, and subsystem status monitor. This master sequencer is only required in the master processor.

The configurator schedules the remaining tasks. It defines the set of application functions to be invoked by request from the request processors or the subsystem status monitor. It also generates a new set of available functions upon significant changes in mission phasing or equipment moding. This set of functions is based on a global knowledge of overall system status.

The structures described above are further divided into modular components. Each component is a closed program, that is a program with only one entry and one exit point. In addition, each program is under control of a program at the next higher level of the hierarchy. Also, the components isolate separate hardware functions; this localizes the impact of equipment changes and allows for mission-to-mission system changes.

Structured programming techniques are used to increase reliability and understanding and to eliminate intricate logic that is difficult to validate and verify. Only closed logic structures--logic structures which have a single entry point and a single exit point--are employed in the construction of program components. To further emphasize standardization and understanding, the higher order language JOVIAL J73 is used for development of the Mission Software. JOVIAL is not used for those portions of the program where computer time must be minimized or a high degree of numerical accuracy is required or interface with processor or bus controller hardware is performed.

3.3 Other Elements

3.3.1 System Mass Memory

This optional subsystem supports the DAIS capabilities to load/reload program modules. In the ITB, it is actually a simulation of a bubble memory device. The simulation is implemented in the universal remote terminal (URT).

A fixed RT address is required for the system mass memory in order for it to be accessed by the bootstrap routine during startup. The system mass memory has four basic operations, as follows:

Address Select - to set a starting address for the mass memory and specify if synchronous or asynchronous operations are to be performed. Also to set read mode or set write mode.

Address Retrieve - to obtain the current (i.e. next to be accessed) mass memory address and settings of the Read/Write and Sync/Async flags.

Read - to transfer data from the mass memory and advance the current address by the number of words transferred.

Write - to transfer data to the mass memory and advance the current address by the number of words transferred.

The format of the messages associated with each of the basic functions is shown in Figure 11. In this example, Receive (T/R=0) subaddress 17 is used for Address Select; Transmit subaddress 17 is used for Address Retrieve; Transmit subaddress 18 is used for Read; and Receive subaddress 18 is used for Write. These are, of course, arbitrary choices that may be changed by the system designer. The significant point is that the startup program must know the bus address of the mass memory and the subaddress of each function.

These messages have some bit definitions unique to the system mass memory. The sync/async bit in the address select command tells the unit whether or not the Last Command Word is to be included as the first data word of a Read operation.

The read/write bit, also in the Address Select message, specifies whether subsequent operations will be writing data to or reading data from the system mass memory. This information is required to properly manage data buffers internal to the interface. It also provides a first level of security, reducing the chances of inadvertent data destruction.

To meet the multiplex bus response times, a buffer interface is necessary. Further, the buffer size is selected so that the time to load the buffer from the memory device is less than the time to transmit the buffer contents over the multiplex bus. A standard double buffering technique can then support a continuous data flow and no delays need be encountered after the initial access.

The bits defined in the status word provide the Master Executive or Startup Loader with information on why the most recent operation was not successful. These are interpreted as follows:

ME - Message Error - indicates data was not received correctly off the bus. The data has not been accepted, and the address has not been advanced.

BSY - Busy - indicates the data transfer could not take place; e.g., buffer full, access in progress.

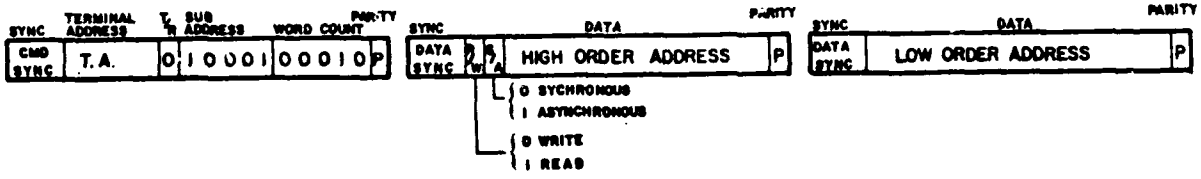
TF - Terminal Fail - indicates an unrecoverable device error occurred on the most recent operation.

The system mass memory is not to be confused with the C&D mass memory unit (MMU). The MMU is a component of the C&D subsystem though a system designer may opt to implement these logically distinct units on a single physical device. (In this case, the interface unit would be required to resolve any access conflicts internally in a manner transparent to the bus operations.)

Alternatively, a specific implementation may choose to economize by omitting the system mass memory, thereby foregoing the inflight reload capability.

ADDRESS SELECT

S.A.=17
T/R=0
WC=2
DATA WORDS CONTAIN R/W FLAG,
SYNC/ASYNC FLAG, AND ADDRESS

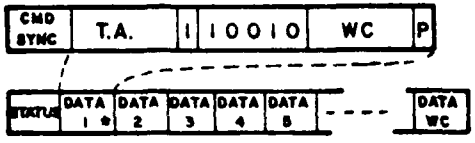


ADDRESS RETRIEVE

S.A.=17
T/R=1
WC=3
DATA AS ABOVE

FIGURE 11. SYSTEM MASS MEMORY MESSAGE FORMATS (Page 1 of 2)

READ
(DATA TRANSMIT)
S.A. = 18
T/R = 1
WC = VARIABLE



* FOR ASYCHRONOUS OPERATION DATA 1 IS LAST COMMAND WORD, I.E., ECHO OF TRANSMIT COMMAND, WITH DATA SYNC INSTEAD OF COMMAND SYNC

WRITE
(DATA RECEIVE)
S.A. = 18
T/R = 0
WC = VARIABLE

STATUS WORD

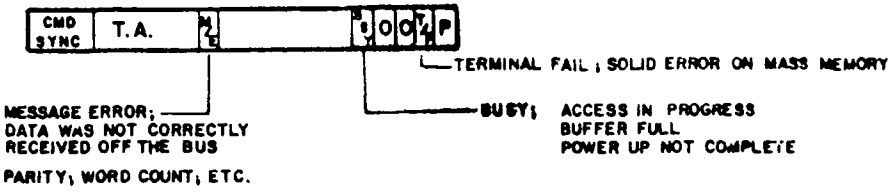


FIGURE 11. SYSTEM MASS MEMORY MESSAGE FORMATS (Page 2 of 2)

3.3.2 Processor Control Panel (PCP)

The Processor Control Panel provides switches to separately enable power to each side (A and B) of the multiplex system and to each of the processors. Two toggle switches and two push buttons provide the pilot control over system load and system restart operations.

The advisory caution panel (ACP) provides a display of hardware and software status during the startup process. This is shown in Figure 12. The two panels are normally referenced together as the PCP/ACP. The lamp test button on the PCP provides a local test of all lights and backlighted push buttons on both panels.

3.4 Non-Real-Time Support Software

The DAIS non-real-time support software provides the tools to develop, generate, test, and partition the mission software for the specific avionics configuration. The non-real-time support software includes the JOVIAL Compiler and the Partitioning Analyzing and Linkage Editing Facility (PALEFAC).

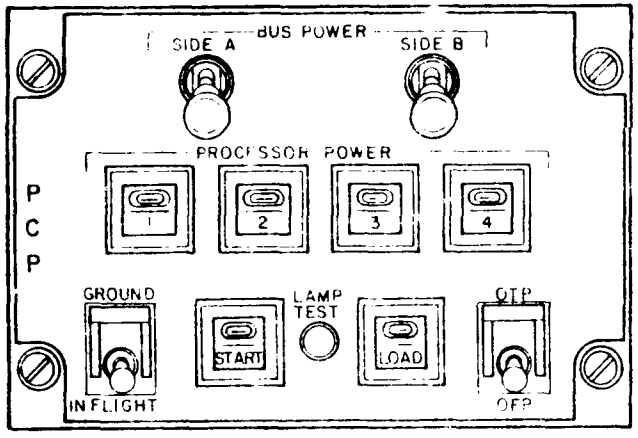
3.4.1 JOVIAL Compiler

The JOVIAL Compiler is a program which translates high-order-language source code into object code. Two versions of the JOVIAL language and, therefore, two versions of the JOVIAL compiler were used in developing the DAIS mission software. Mission Software was first written in JOVIAL J73/I. In 1980, all Mission Software was converted to JOVIAL J73. Both versions of the compiler serve the same function and fill the same role in the DAIS system architecture.

The JOVIAL compiler generates object modules which can be loaded and executed on specified target computers (such as the DAIS processors or the DECsystem-10 computer). The JOVIAL language is machine-independent. It uses self-explanatory English words and familiar notation for algebraic and numerical computations and logical operations. The JOVIAL compiler provides a common, powerful, and easily understandable programming language for a wide range of applications. The compiler provides a capability for sharing data among interdependent programs or tasks through a communication pool (compool).

The JOVIAL compiler is designed to be easily retargeted to produce object code for a new and unspecified computer. It is also designed to be easily rehostable to another host computer. Initially, the compiler was hosted on the DECsystem-10 computer system and targeted for the DAIS processor and DECsystem-10.

The structure of the compiler is shown in Figure 13. The system programmer generates the source input and compool input files. The programmer then compiles the source program using the compiler which produces source listings, cross references, object listings, and the relocatable object files for the selected target computer.



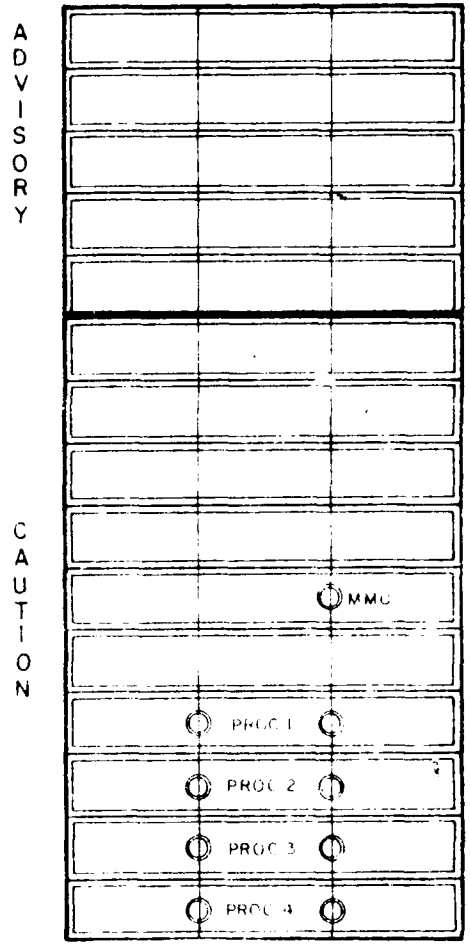


FIGURE 12. PCP/ACP.

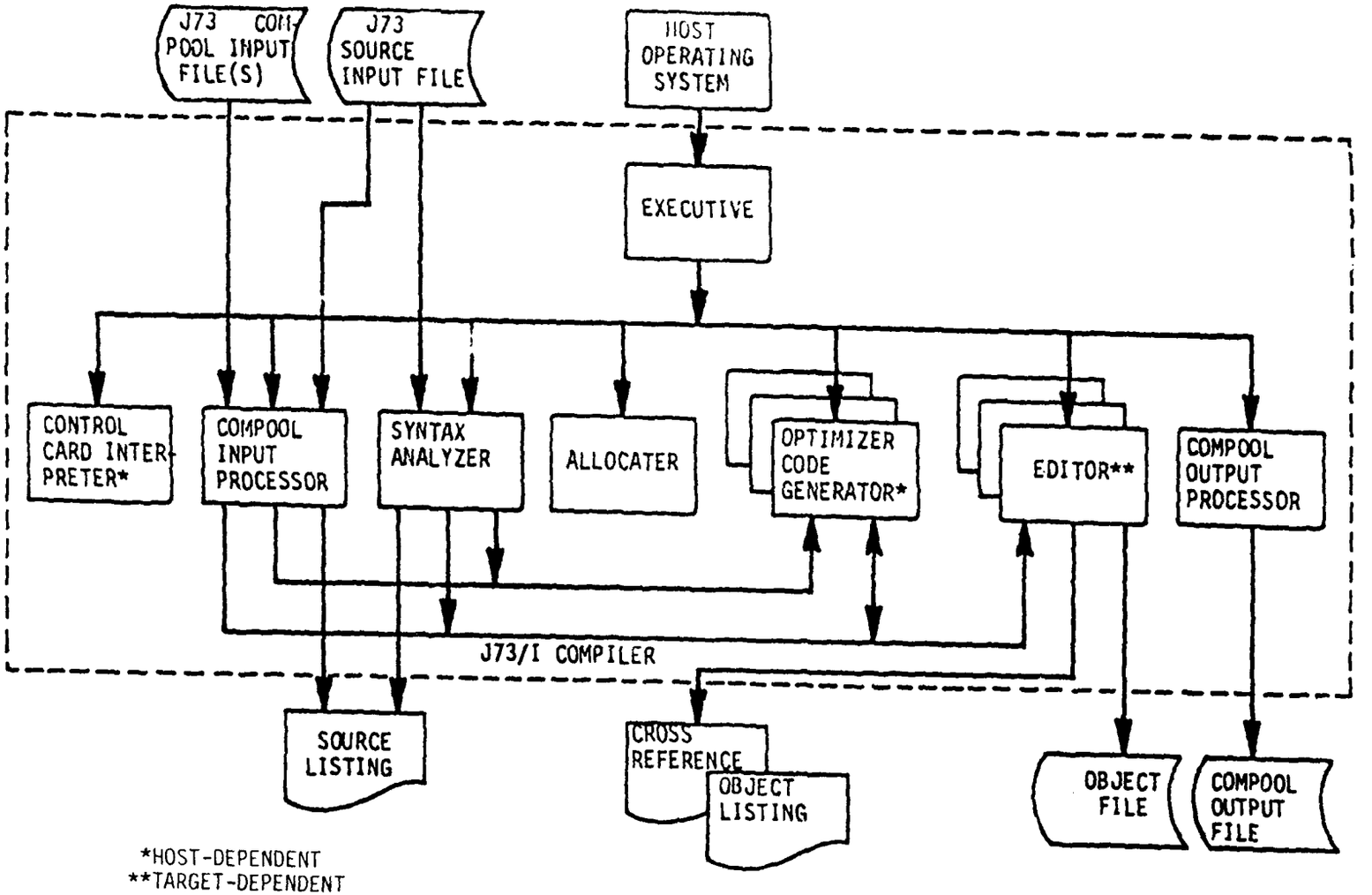


FIGURE 13. J73 COMPILER STRUCTURE

The output of the compilation is a relocatable object file. In order to execute the program on the DECsystem-10, the object file must be converted to core image form using the DECsystem-10 (LINK-10) linker/loader. In order to execute the program on the DAIS processor, it must be converted to core image form using the HBC linker/loader.

The JOVIAL Programmer User's Manual provides definition of the J73 language, properties, and statements including examples of how to write a J73 program. The guide also presents a syntactic description of the language elements with examples and necessary information for the user to write, compile, load and execute programs on the target computers.

3.4.2 Partitioning Analyzing and Linkage Editing Facility (PALEFAC)

PALEFAC is a non-real-time support software tool for use by the system designer and application programmer to:

1. Build the data tables for the DAIS local and master executives.

2. Provide bus analysis and module partitioning information.

3. Produce the executive tables in J73 source code.

4. Generate linker command files for each DAIS processor for the DECsystem-10 Linker or DAIS Processor Linker.

PALEFAC is used as a stand-alone tool. PALEFAC consists of three basic programs, as shown in Figure 14: PALEFAC pre-processor, PALEFAC main program, and PALEFAC module input (PMI) decoder.

PALEFAC is used to build the mission software load modules for each of the DAIS processors. Inputs to PALEFAC are the Application Software modules including the executive service requests generated by the application programmer in J73 source code. The pre-processor reads each application module and creates a record for each application module in the PMI file.

The system designer prepares a PALEFAC Global Input (PGI) file based upon the specific system configuration, partitioning of application tasks to each processor, and bus messages to each terminal. This would include:

a. Bus Messages

- To/from data block name or Terminal Address/ Subaddress
- Cycle (period and phase) for synchronous transmissions
- Activity request identification for asynchronous transmission

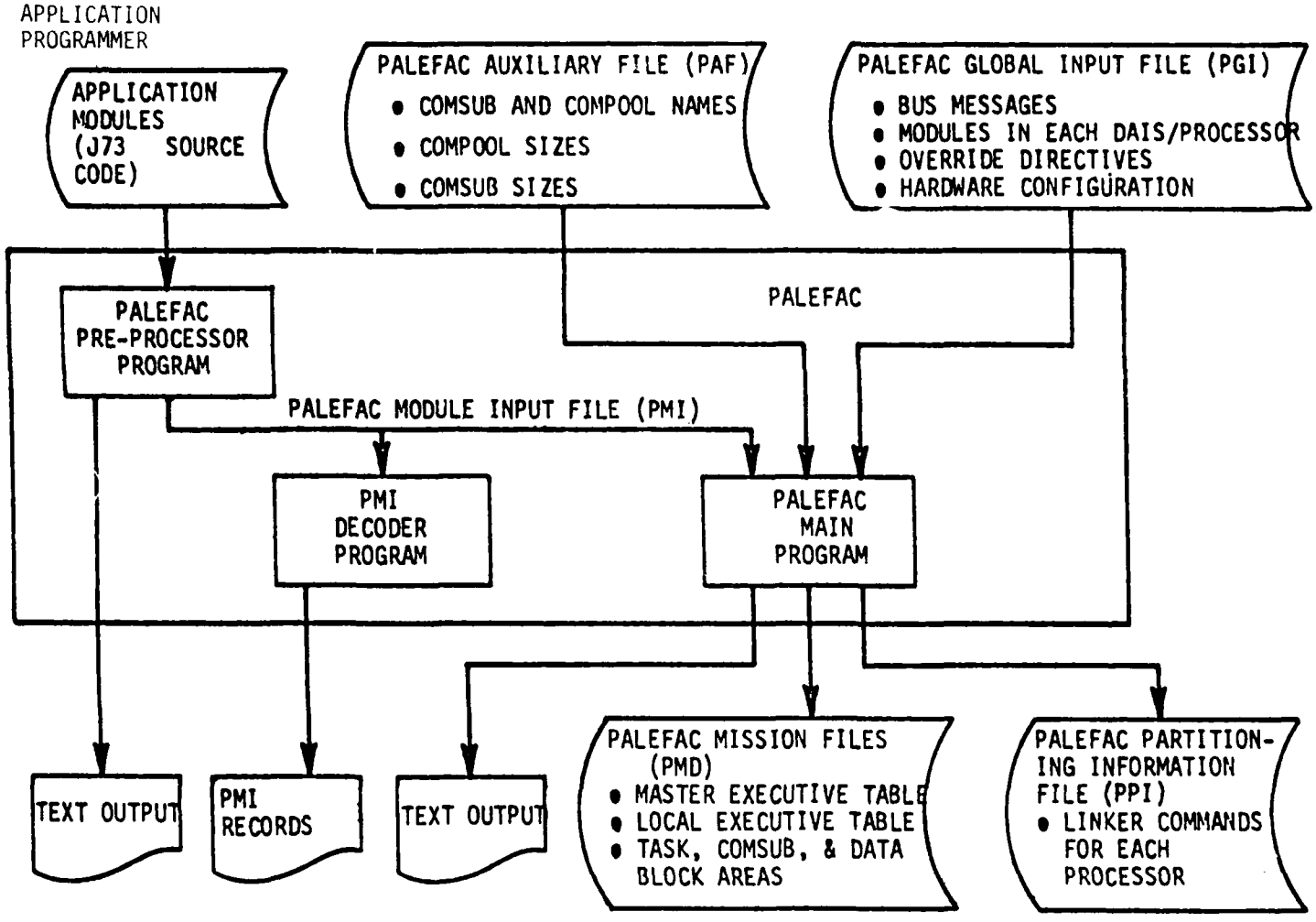


FIGURE 14. PALEFAC OVERVIEW

- Word count
- Class for message error retry

b. List of all tasks or modules for each processor

c. Override directives

d. Number of processors in the system configuration and address of each terminal

The system designer also generates the PALEFAC Auxiliary File (PAL) for common subroutines (comsubs) and common communication data blocks (compools).

The PALEFAC main program builds the executive tables and linker command files based upon the system designer specified configuration. The output of PALEFAC is the PALEFAC mission files (PMD), PALEFAC Partitioning Information Files (PPI), and test output files shown in Figure 11. The PMD files contain all the executive tables including the bus control tables in J73 source code. The PPI files contain the linker commands for each processor and specify all the mission software modules (e.g. executive modules, application tasks, comsubs, and executive tables) in order for the linker to produce the load modules for each processor. PALEFAC also produces text output files listing all the PALEFAC input files, the output files, and error messages.

3.4.3 Assemblers/Linker

ALAP is a retargetable assembler written in Fortran IV and hosted on the DECsystem-10 computer at the AF Wright Aeronautical Laboratory. The assembler can be retargeted to a variety of machine languages. Its function is to translate assembly language source into object modules which can be loaded and executed on a specified target computer (e.g. AN/AYK-15 processors, AN/AYK-15A processors, DECsystem-10 computer) or into relocatable files for further processing by LINKS.

LINKS is a target-independent linker written in Fortran IV and hosted on the DECsystem-10 computer at the AF Wright Aeronautical Laboratory. Its function is to combine relocatable files produced by the ALAP assembler and/or the JOVIAL compiler into a single object file ready for execution on a specified target computer.

4.0 DAIS SYSTEM CHARACTERISTICS AND FEATURES

The performance characteristics of DAIS include system functional control characteristics as well as system architecture features and attributes included in the design of DAIS core elements. These sections address the system control features and the primary characteristics which are unique and important to the DAIS concept.

4.1 System Control Procedures

The DAIS architecture and integrated system design result in a special set of functional requirements, designated system control procedures, which include system startup/restart, control of the bus communication, and utilization of redundancy in the event of core element failures. The system and bus control functions are distinguished from mission avionic functions which support the specific mission tasks. The mission avionic functions are composed of navigation, guidance, weapon delivery, communications, target acquisition and track, stores management, and subsystem management.

Another important group of functions, on-board tests, utilize both hardware and software to isolate in-flight failures. This allows utilization of redundancy, maintains a complete updated status of every equipment on the system, and provides isolation of failure to LRU levels with a minimum of auxiliary group equipment.

4.1.1 System Startup/Restart Operations

The overall objective of the system startup/restart procedures is to give the pilot control of the system configuration and to provide a verification of the hardware and software prior to entering normal system operations.

The pilot controls the system via the Processor Control Panel (PCP). The power enable switches allow him to remove a processor from the system by powering it down. The GROUND position of the startup switch permits him to specify an initial startup as opposed to a restart, indicated by the INFLIGHT position. The START button permits him to request the use of software already loaded, while the LOAD button allows him to force the reload of all software. Of course, the LOAD function is supported only when a system mass memory is configured in the system.

In addition to the pilot-initiated restarts, the system will, under certain limited circumstances, perform an automatic restart. It is essential that an automatic restart have minimal impact on continued system operation. This consideration defines the need for a "warm start" initialization of software as opposed to the "cold start" associated with normal system startup.

"Cold start" assumes a fully operational configuration. It involves the complete initialization of executive and application software and the assignment of initial values to events and data.

"Warm start" continues the current operational status. A partial initialization of the executive is performed, current data is maintained, and application tasks and events are assigned to a known, predetermined state.

The various startup/restart cases are summarized in Table 4. As is evident from this table, it is important the pilot set the startup switch before powering up the processors. Otherwise, the power-up would look like a transient recovery. Conversely, the pilot must return the switch to the INFLIGHT position at the completion of a normal startup. Failure to do so would allow a transient to bring the system to a stop.

4.1.1.1 Normal System Startup

Each processor is equipped with a read only memory (ROM)*, which is enabled when the power up condition is detected. The ROM contains sufficient instructions for the processors to perform self tests, determine the integrity of loaded software and arbitrate control of the multiplex bus. The ROM may contain additional functions as desired for any specific implementation, but normally considerations of stability and size would indicate only essentials should be in the ROM. The minimum ROM contents are dictated by the requirement that the ROM be capable of starting or restarting a system without a system mass memory.

As each processor senses power-up, it enables the ROM and begins executing at location zero. Memory protection is established, interrupts are cleared, and self tests are performed on the processor/BCM.

At the completion of self tests, software is verified and one processor is established as controller of the multiplex bus. The other processors remain in remote mode, prepared to transmit the results of self tests and software verification. The controller polls all possible processors for this information to determine the configuration to be run. A processor is included in the configuration if it passed self test and it successfully communicates on the multiplex bus.

Software verification/loading is the final phase of the startup process. Based on the configuration which is operational, the system loader accesses a startup configuration table to determine the load module required in each processor. If the required software is not present, it is loaded from system mass memory, and the verification process repeated. When it is confirmed that all processors are properly loaded, the controller directs each processor to perform initialization. The controller then enters its own initialization and normal system operation is begun.

4.1.1.2 System Warm Start

System Warm Start is designed primarily as a response to a system-wide power transient. The objective of a warm start is to resume operations as quickly as possible. The procedure starts what it finds and verifies in the processors' memory and does not perform reloading. The pilot will be alerted if a runnable set of software is not found.

*The ROM, in effect, overlays the first four thousand words of memory while invoked. This is a feature of the AN/AYK-15A processor and, since it is the preferred implementation, the startup procedures are described in this context. With the AN/AYK-15, or other processor not equipped with a ROM, the startup software must be permanently preserved in low memory and memory protected except as required when modifying interrupt vectors.

TABLE 4. START/RESTART CASES/OPTIONS

	PCP SWITCH SET TO GROUND	PCP SWITCH SET TO INFLIGHT
PCP START BUTTON DEPRESSED	NORMAL STARTUP • LOAD ONLY IF REQUIRED • COLD INITIALIZATION	SYSTEM RESTART (PILOT) • VERIFY RESIDENT SOFTWARE • NO RELOAD • WARM START
PCP LOAD BUTTON DEPRESSED	NORMAL STARTUP • FORCE RELOAD • COLD INITIALIZATION	SYSTEM RELOAD • FORCED RELOAD • COLD INITIALIZATION
POWER TRANSIENT	WAIT FOR START/LOAD	SYSTEM RESTART (TRANSIENT) • VERIFY RESIDENT SOFTWARE • NO RELOAD • WARM START

As power is restored to each processor, it will enter the ROM, perform self tests, verify software and build startup status information as in normal startup. The absence of any bus activity permits one of the processors to assume control to manage the warm start. The fact that it is a warm start is determined by the INFLIGHT position of the start switch and the absence of the discretes for both the start button and the load button.

The startup configuration table defines the acceptable configurations for a warm start in their order of preference. This information, along with the module ID's required for an initial load, is kept in the fixed information table in each processor to facilitate quick transient recovery. If the procedure cannot find the configuration matching the initial load (which by definition is the most preferred configuration for a warm start) it then goes through the restartable list trying to find a configuration in which each of the load module ID's can be found in some processor.

Should no restartable load set be found, the pilot is alerted by the master caution lamp. The startup process is then recycled to the point of reading the PCP prior to identifying the configuration. The procedure will, therefore, automatically detect a processor that recovers from a power transient much slower than the others. It will also respond to new direction from the pilot via the PCP.

4.1.2 Normal System Operation

In normal system operation one of the processors, designated the master processor, contains the master executive and controls the bus. Each of the remaining processors is designated a remote processor and each contains a local executive. Application software is distributed among all the processors.

4.1.2.1 Bus Control Operation

Bus control operation is concerned with the control of the bus message and sequences as defined in MIL-STD-1553B and as shown in Figure 6. The control of bus traffic operations provides for major/minor cycle synchronization, which is maintained via transmission of minor frame (synchronize mode command) messages to each remote processor, and provides for both synchronous and asynchronous bus communication.

The synchronous bus traffic is controlled by a predefined bus command list. Asynchronous messages, which are predefined, are scheduled by the master executive based upon requests from terminals or application tasks. In addition, the bus control operations themselves may generate asynchronous messages in managing exception conditions.

In general, asynchronous operations are given priority over synchronous operations. A more complete breakdown of the priorities (1 is the highest priority, 6 is the lowest) is included in the summary of bus message operations, Table 5.

Message protocol is defined as that sequence of bus messages required to perform the normal multiplex synchronous and asynchronous operations and to support the error detection and recovery process. In order to insure

TABLE 5. BUS MESSAGE OPERATIONS (SHEET 1 OF 4)

BUS MESSAGE OPERATIONS	PRIORITY	FUNCTION	MECHANISM
1. Minor Cycle Message (To Remote Processor)	3	• Signals Remote Processors to start a new minor cycle. • Minor Cycle is the shortest cyclic period & major frame is the longest period a synchronous action can have.	• Master initiates minor cycle sync messages upon expiration of minor cycle timer & completion of synchronous bus messages for the last minor cycle. • Master selects instruction list for the current minor cycle.
2. Synchronous Messages	5	• Periodic messages transmitted in a specific minor cycle as determined by period & phase. Period is the message iteration or sample rate. Phase is the displacement from the first minor cycle.	• Master BCI executes synchronous bus list for the specific minor cycle.
3. Asynchronous Messages (General)	4	• Asynchronous bus messages are initiated upon a request from one of the system sources: remote terminal/subsystem, or application task. Types of asynchronous messages are identified below.	• Master suspends lower priority bus list and executes a predefined bus instruction set for the asynchronous message.

TABLE 5. BUS MESSAGE OPERATIONS (SHEET 2 OF 4)

BUS MESSAGE OPERATIONS	PRIORITY	FUNCTION	MECHANISM
3A. Interprocessor Async Message	4	• Transfer asynchronous data or executive service requests between processors.	• Local executive (if not in master processor) signals Master by placing Asynchronous Request Vector (ARV) in status code register. BCI "or's" ARV into Status Word Service Request (SR) Bit. Master detects SR Bit in status and fetches ARV via Transmit Vector mode command. If the local executive is located in the master processor, the ARV is passed directly. The ARV points to a predefined bus list which is executed to effect the message transmission. The first data word of the message is the Async ID used to identify the message to the receiving processor.
3B. Critically Timed Messages (To Remote Terminal)	2	• Application tasks in any processor request a message be transmitted to a specific remote terminal at a specified future time.	• When critical timer expires, Master suspends lower priority bus list and sends predefined message. If application task is in remote processor, a prior interprocessor asynchronous message to the master is required to identify the critically timed message and time it is to be sent.

TABLE 5. BUS MESSAGE OPERATIONS (SHEET 3 OF 4)

BUS MESSAGE OPERATIONS	PRIORITY	FUNCTION	MECHANISM
3C. Remote Terminal-Requested Asynchronous Message (From Remote Terminal)	4	• Asynchronous request from Remote Terminal is initiated by subsystem for transmission of data to a processor.	• Subsystem flags request to RT which sets service request bit in the status word and the activity bit in the activity register. Master sends Transmit Vector Mode Command to retrieve activity register. Master executes bus instructions to transmit the data followed by a Synchronize Mode Code. The first data word transmitted is the RT Last Command Register which identifies the RT asynchronous message.
3D. Processor to Remote Terminal Message	4	• Application task in processor requests a message be transmitted to a specific remote terminal.	• Master executes predefined bus instruction set. If application task is in a remote processor, local executive signals Master Executive with SR Bit/Vector Request sequence to identify instruction set. No Async ID word is necessary.

TABLE 5. BUS MESSAGE OPERATIONS (SHEET 4 OF 4)

BUS MESSAGE OPERATIONS	PRIORITY	FUNCTION	MECHANISM
4. Message Error and Terminal Failure Analysis Operation	1	• Transmission of various mode commands to support message error analysis and terminal exception/BIT word analysis.	• Master executes assorted Mode Code Operations to confirm the error and determine status of receiver and/or transmitter.
5. Status Polling Messages	6	• Periodically poll each terminal for status and asynchronous message requests.	• When Master Executive has no other messages to be transmitted, a predefined bus list of Transmit Status Mode Command will be initiated. Each terminal will be polled at least once each major frame or at a periodic rate as required for asynchronous requests.

minimum processor executive and multiplex bus overhead required to implement the message protocol, two concepts are introduced: mode commands and system control procedures (Figure 15).

System control procedures define the message protocol for the multiplexed data bus system. They serve to provide the executive software designer with the logical decisions required to manage normal system operation. They also define the required system flow to detect and recover from random bus message errors and selective terminal failure modes. The system control procedures serve to link the executive mission software, bus message protocol, and the terminal-to-subsystem interface management.

A time slice approach where messages occur on a minor cycle/major cycle basis is used to control bus message operations. Messages that are transmitted at specific iteration rates within a major cycle are defined as synchronous messages. Each synchronous message is assigned a period and phase for transmission. The number of minor cycles per major cycle is specified by the system designer. An example is 128 minor cycles every second or major cycle. Messages that are to be transmitted on a demand or request basis and not at a specific phase and period are designated as asynchronous messages. Table 5 summarizes the bus message operations.

Bus control operations provide for major/minor cycle synchronization and for synchronous and asynchronous bus message transmissions as shown in Figure 16. The synchronous bus messages are controlled by a predefined bus instruction list. The bus instruction list is part of the Master Executive preloaded tables. It contains an instruction pair for each synchronous message that is transmitted on the data bus. Since not every message is transmitted every minor cycle, the instruction lists are organized so that the instructions for all messages with the same period and phase are grouped into a block. Each block ends with a link to the block for the next higher message rate with the appropriate phase. In this way a static list can be employed. No dynamic linking is required each minor cycle; the Executive just starts the bus at the correct place in the list.

Each data word on the bus is transferred to/from the processor's memory in real time via a direct memory access channel. The address at which each message block begins is constructed by combining the current base address, the transmit/receive bit, and the subaddress into the pointer address, and then fetching the contents of the address via DMA. Consecutive data words are stored in contiguous memory locations until the word count for that particular message is exhausted. For all receive operations, each message block is preceded by a generated tag word which contains the current minor cycle number, word count of the received message, and a message error flag.

Asynchronous messages are scheduled by the master executive based upon requests from remote terminals, application tasks, or the master executive to perform message error or terminal failure operations. Subaddress 30 is dedicated to asynchronous receive messages. After receiving an asynchronous message, the BCM interrupts the processor and the local executive updates the pointer word for storing data before restarting the BCM. A transmit asynchronous message is sent via subaddress 29 and does not cause an interrupt.

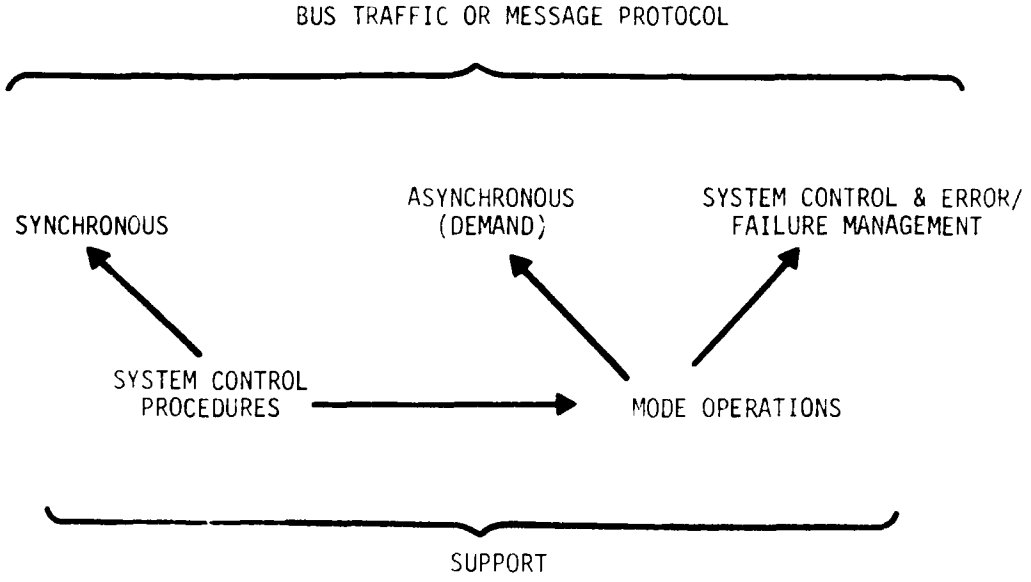


FIGURE 15. BUS OPERATIONS

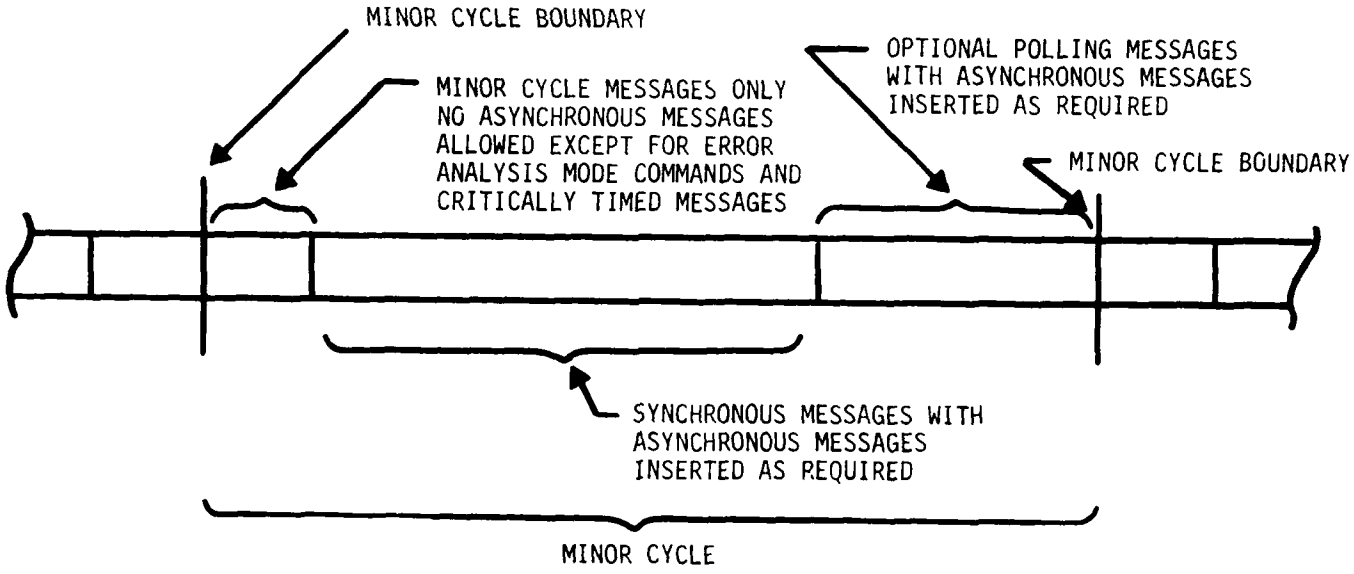


FIGURE 16. MESSAGE TRANSMISSION DURING A MINOR CYCLE

The first word of all asynchronous messages is used to designate the source of the message. This asynchronous message identifier is used by the local executive to process the data or identify the executive services action required.

There are several types of asynchronous bus message operations based upon the resource making the request and protocol to process the request as defined in Table 5. Upon receiving the request, the master executive causes the BCM to halt at the end of the current message operation. It then executes a predefined or generated bus instruction pair to initiate the asynchronous message operation. Figure 17 shows the message protocol sequence for a subsystem connected to an RT requesting service. Asynchronous operations are used for pilot panel inputs and weapon release messages, as examples. Figures 18 and 19 show asynchronous message protocol sequences for interprocessor operations. Interprocessor messages are used to signal master mode changes between application tasks in different processors and steering/guidance waypoint passage, as examples.

4.1.2.2 Mode Command Operations

The mode command operations are indicated by using a subaddress of Ø or 31 with the specific mode code in the word count field. Table 6 specifies the mode commands required to support normal operation and error/failure management. The MIL-STD-1553B Mode Codes have the following uses in DAIS.

a. Mode codes 2, 18, and 19 are used to support detection and recovery from bus message errors.

b. Mode codes 2, 3, 8, 18 and 19 are used to support terminal failure analysis.

c. Mode codes 3, 4 and 5 are used to control the redundant buses.

d. Mode codes 1, 16 and 17 are used in asynchronous bus operation.

e. Mode codes 0, 6, 7, 20, and 21 are not used.

Table 6 indicates which mode command operations are required in a remote device to make it compatible with the DAIS system control procedures and master executive.

Table 7 shows the use of the mode commands in relationship to the system control functions.

4.1.3 Application Software Executive Services

The DAIS Local Executive provides certain services to the Application Software. It defines both software elements and real-time statements. The software elements allow program and data modules to be classified according to their function. The real-time statements operate on the software elements.

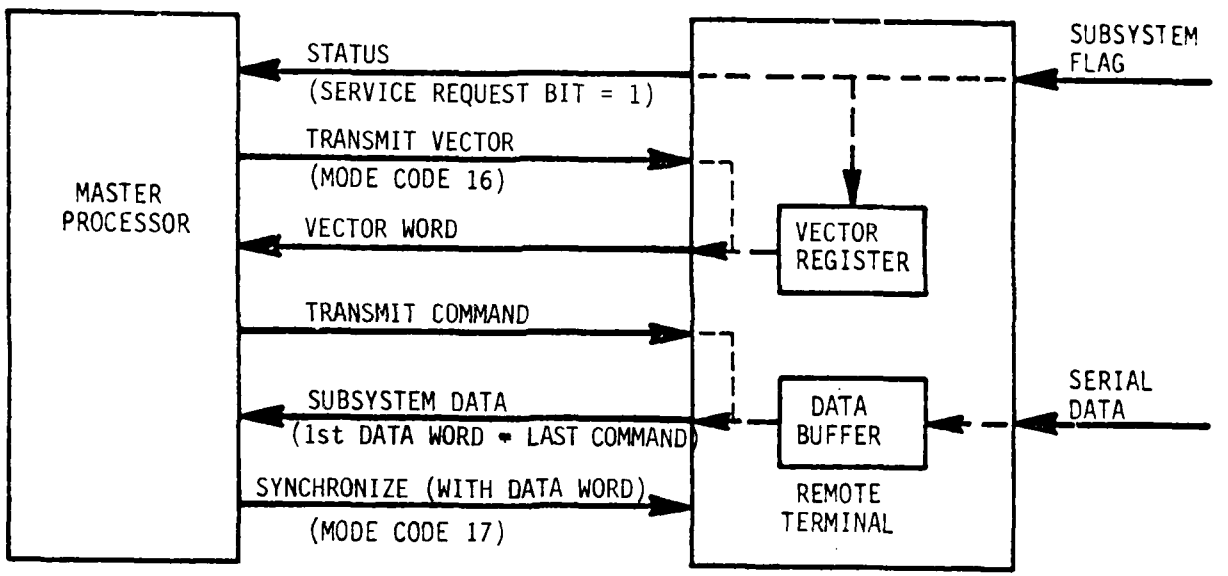


FIGURE 17. ASYNCHRONOUS MESSAGE PROTOCOL - REMOTE TERMINAL REQUEST

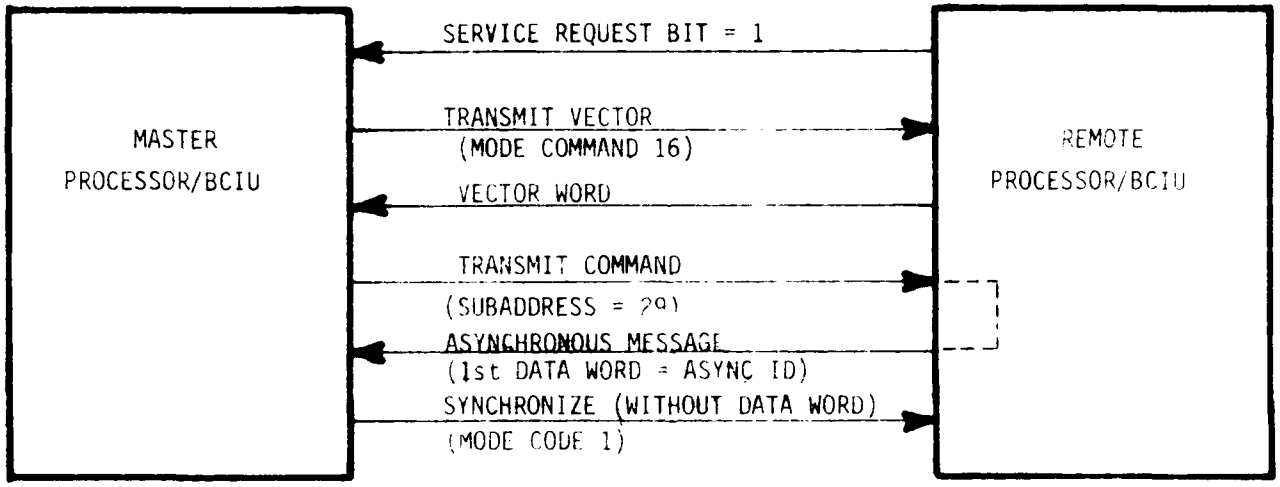


FIGURE 18. ASYNCHRONOUS MESSAGE PROTOCOL - INTERPROCESSOR MESSAGE, EXAMPLE #1

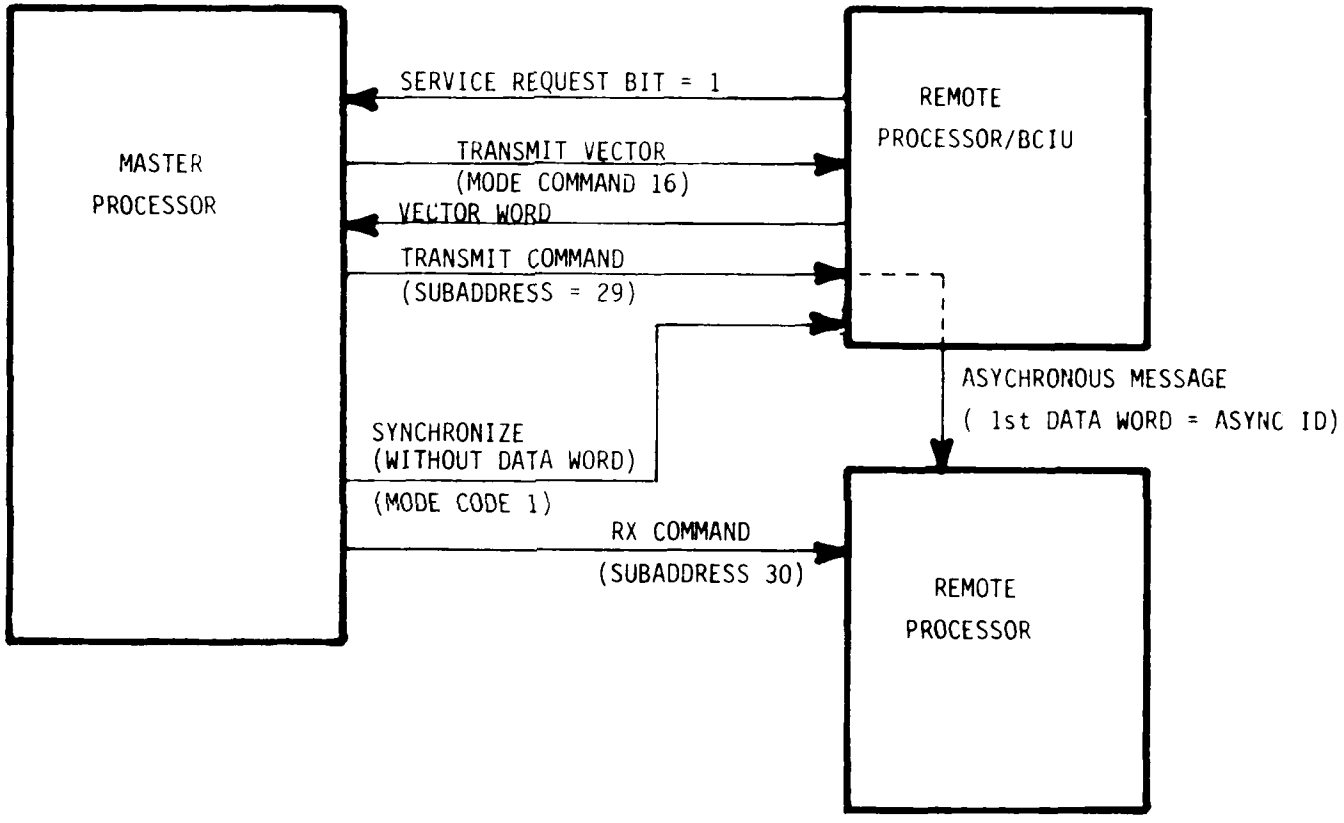


FIGURE 19. ASYNCHRONOUS MESSAGE PROTOCOL - INTERPROCESSOR MESSAGE, EXAMPLE #2

TABLE 6. MIL-STD-1553B MODE CODE COMMANDS

MODE CODE NUMBER	DESCRIPTION	PROTOCOL	OPERATION
1	Synchronize	No data	Synchronize terminal
2	Transmit status word	No data	Current terminal status
3	Initiate self-test	No data	Initiate self test in terminal
4	Transmitter shutdown	No data	Disable redundant bus transmitter
5	Override transmitter shutdown	No data	Enable redundant bus transmitter
8	Reset remote terminal	No data	Reset terminal to initialized (Power up) state
16	Transmit vector	1 data word, TERM-TO-CONT	Current service request information
17	Synchronize	1 data word, CONT-TO-TERM	Data word contains synchronize information
18	Transmit last command	1 data word, TERM-TO-CONT	Last valid terminal command
19	Transmit BIT word	1 data word, TERM-TO-CONT	Current built-in-test register

CONT = controller
TERM = terminal

TABLE 7. MIL-STD-1553B MODE CODE RELATIONSHIP TO SYSTEM FUNCTIONS

DAIS SYSTEM FUNCTION / SUPPORTING MODE CODES	0 NOT USED	1 SYNCHRONIZE	2 TRANSMIT STATUS	3 INITIATE SELF TEST	4 SHUTDOWN BUS	5 SHUTDOWN OVERRIDE BUS	8 RESET TERMINAL	16 TRANSMIT VECTOR	17 SYNCHRONIZE	18 TRANSMIT LAST COMMAND	19 TRANSMIT BIT	6, 7, 9-15, 20-31 UNDEFINED
NORMAL SYSTEM INITIALIZATION		X	X			X						
MINOR CYCLE SYNCHRONIZATION									X			
SYNCHRONOUS MESSAGE OPERATION		X										
ASYNCHRONOUS MESSAGE OPERATION	X	X						X	X			
ERROR/RETRY MANAGEMENT		X	X			X				X	X	
POWER CONTROL			X	X	X							

The Application Software elements which are recognized by the local executive software are Tasks, Comsubs, Compool Blocks, and Events, defined as follows:

a. Tasks are programs or processing modules which can either be controllers or calculators. Application software is organized in tasks (program modules) with each performing a function as required for the specific system application. Each task contains executable code and local data.

b. Comsubs are common subroutines which differ from tasks in that they are required to be re-entrant. Comsubs can be used by many tasks, so that one task using a particular comsub can be suspended in the middle of the comsub routine by another task using the same comsub.

c. Compool Blocks provide data communication between tasks and the external world or equipment. A buffer system is provided by the use of global copies and local copies to prevent a compool block from being read when it has been partially updated. Every task, except privileged tasks, interfaces with a local copy while performing its calculations. The local copy is updated from the global copy with the READ statement; the local copy updates the global copy by the write statement. A special statement (TRIGGER) is used for a critically timed compool block.

In a privileged task, communication is allowed directly with the global copy in the processor in which the task resides. The privileged task cannot be interrupted by another task or by the Executive.

Since the READ, WRITE, and TRIGGER statements invoke Executive services that operate in the Privileged Mode, this guarantees a task will not reference partially updated data.

d. Events are data items used to communicate real-time conditions signalled by a process. An event is either "ON" or "OFF".

Real-time statements are used by tasks to control the state of other tasks, control values of events, and control reference to compool blocks. They are J73-defined statements which the application programmer uses during the development of mission software as follows:

Schedule Statement
Cancel Statement
Terminate Statement
Wait Statement
Signal Statement
Read Statement
Write Statement
Trigger Statement
Broadcast Statement

The first four statements are used to control task states. These states and the method of transitioning from one state to another state are shown in Figure 20.

a. Schedule Statement. Schedule statements are used to place an uninvoked task into an invoked state. An invoked task becomes active and dispatchable when the required event conditions are satisfied. An invoked

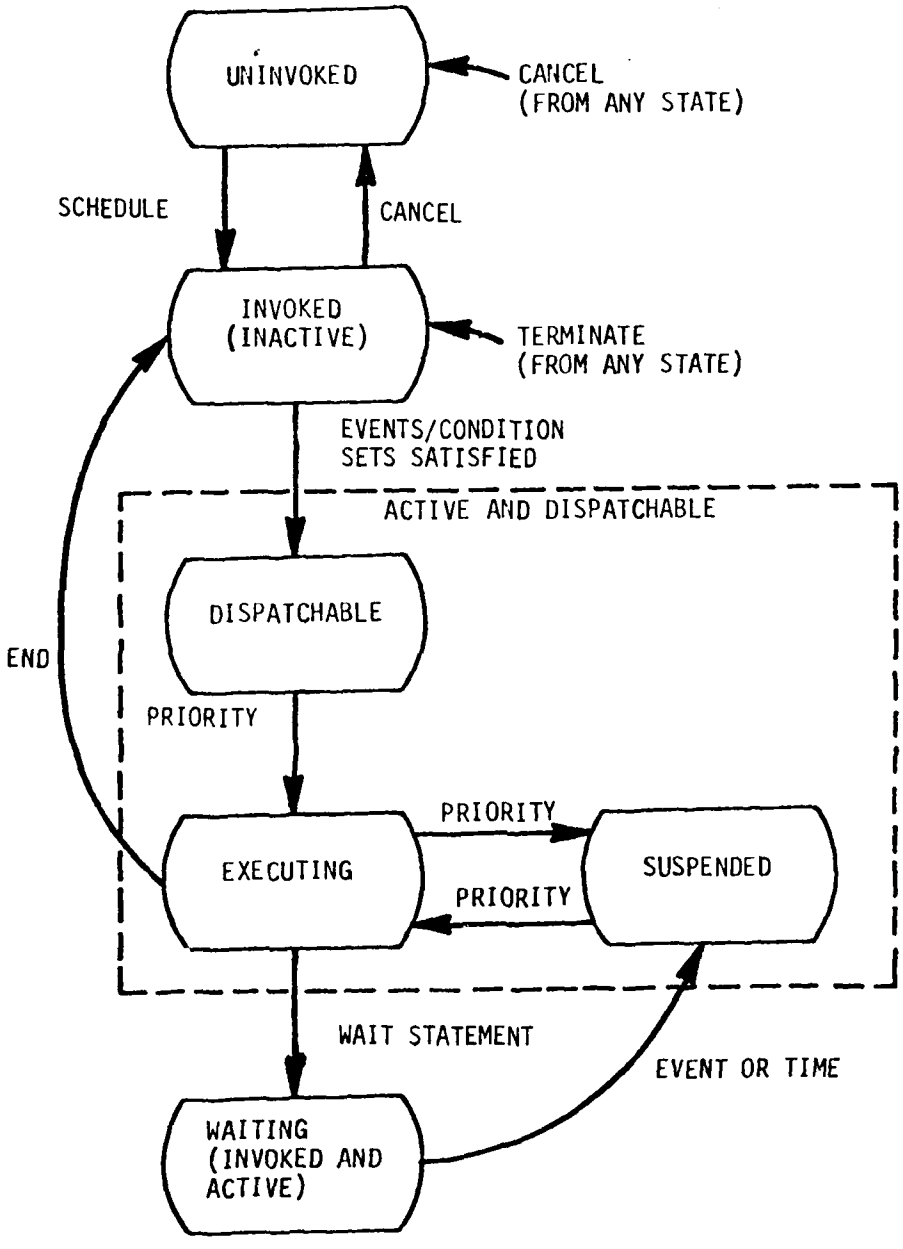


FIGURE 20. TASK STATE DIAGRAM

task is placed in execution by the Executive if it has the highest priority among the tasks which are dispatchable.

A schedule statement includes the name of the task to be scheduled, the mode of task (privileged or normal), priority of the task, and event conditions to be satisfied.

b. Cancel Statement. The Cancel Statement is used by one task to place another task in the Uninvoked State. If a task is cancelled, all the lower branches or sons of the tasks in the control tree structure are cancelled.

c. Terminate Statement. The Terminate Statement is used by one task to place another task in the inactive, but invoked, state. If a task is terminated, all the lower branches or sons of the tasks in the control tree structure are also terminated.

d. Wait Statement. Wait Statements are used by tasks to place themselves into the Wait State pending occurrence of one of the following conditions:

1. Absolute Time
2. Relative Time
3. Latched Event
4. Unlatched Event

For the latched or unlatched events, the task remains in the Wait State until the specified event, either on or off, occurs.

e. Signal Statement. Signal Statements are used by tasks to set a specified event to a specified value, either on or off.

f. Read Statement. Read Statements are used by tasks to copy the values of specified compool blocks (global copies) into a local copy to be used by the task.

g. Write Statement. Write Statements are used by the tasks to copy the values of specified local copy into the specified compool block (global copy). The Write Statement causes global copies in other processors or virtual copies in remote terminals to be updated and causes signalling of an event associated with the compool block.

h. Trigger Statements are used by a task to send a specified critically-timed compool block (data) to a specified remote terminal at a specified time. This causes a critically-timed asynchronous bus message operation to be performed where the local copy of a specified compool block is sent to a global copy in the Master Processor and where, at the specified time, the data is transmitted to the specified remote terminal.

i. Broadcast Statement. The Broadcast Statement is used only by a privileged mode task. After updating a global copy of a compool block, the broadcast statement is used to cause the asynchronous update of remote and/or virtual copies of the compool block.

4.1.4 Error and Failure Management

Bus message error management and terminal failure management are performed by the master BCM and the master executive. In addition, each terminal on the multiplex data bus provides mechanisms for detecting and recording message error conditions and terminal failures. Message error conditions and terminal failure conditions are recorded in the Built-In-Test (BIT) word which can be obtained by the master with a mode command. Also, each terminal records in a register (Last Command) the last valid command received by that terminal which also can be obtained by the master with a mode command.

The master executive and BCM centrally manage the message protocol to recover from message errors and reconfigure for terminal failures via the following activities.

a. Monitor the message sequence to determine successful or unsuccessful completion.

b. Analyze error response and determine type of retry procedure for unsuccessful messages.

c. Perform retry of unsuccessful message, first on the same data bus.

d. Analyze error response if not successful and retry the unsuccessful message on redundant data bus.

e. Perform failure analysis if retry of message on alternate bus is unsuccessful.

f. Update the system configuration tables based upon the error conditions and failures.

The remote devices on the multiplex bus collaborate in message error detection by validating each received word. A word is invalid if it contains:

- Parity error
- Incorrect or missing sync
- Incorrect bit count
- Invalid Manchester format

A remote receiver will not respond with a status word after detecting a message error. As message data is received by a BCM, it is stored in the processor memory via DMA. If received unsuccessfully, the message error bit in the tag word is set. As message data is received by an RT, the data is temporarily stored. If received unsuccessfully, the data is not transferred to the subsystems.

Message Error analysis and retry procedures are performed by the Master Processor/BCM. Error analysis is based on:

- The bus instruction which caused the message to be sent.

- The status word(s) received from the transmitting and/or receiving terminal involved in the operation, and

- The interrupt code register (ICR) and internal status register (ISR) in the Master BCI.

The method of retry is based on specifications of the system designer, the type of message operation, and the manner in which the error is detected. The majority of messages are simply retried the number of times specified in the individual bus instruction, and then if necessary retried again on the redundant multiplex bus. The system designer may specify that a message requires a careful retry. This is done if a subsystem would be degraded or function improperly if sent repeated data. In this case, the message error management must first confirm that the data was not properly received before repeating the message.

The type of message operation influences the message error management in several ways. Synchronous messages may normally be repeated immediately. A second message simply overwrites the first and the message data is not scheduled for processing until a later minor cycle anyway. If one assumes that the source of a message error is of short duration and occurs randomly, it is easy to see this is the most effective approach. The BCI can recover the error automatically and an interrupt of the processor is not even required. Asynchronous messages to a remote processor require careful retry. Such messages are separately buffered and queued as received and thus risk duplication if repeated. An asynchronous message from a remote processor can be retried immediately since the remote does not perform buffer management/queue update processing until after the synchronize mode command is received.

The method of error detection is principally a question of whether the master is explicitly aware of the message error. This is the case when a data error is detected while the master is receiving data (i.e. RDE indication in a terminal to controller operation). In this instance, no further analysis is needed. When the primary error detection is by a remote device, the situation is less clear. In accord with MIL-STD-1553B, remote devices, upon detecting an error, do not return any status. Subsequent mode code operations are necessary for the master to acquire and examine the status word associated with the most recent operation.

However, the status word error bits in the ISR do not necessarily imply a message error. They indicate the absence of a good status, which could simply mean a bus error occurred during the transmission of the status word itself. A similar result occurs if a bus error happens during the transmission of a command word. The intended terminal will not respond to a garbled command, and the Master BCI will set the status error bit after an appropriate time out. But, in this later case, note that the status word contained in the remote device will be a good status left over from some previous message operation. This defines the need for the last command sequence.

It is to be recognized that the last command register is maintained in each TCU in an RT and is, therefore, bus-dependent. The mode code to transmit

the last command must be sent on the same bus as referenced in the bus instruction pair for that RT. Also, the Last Command register is updated by all commands other than Transmit Last Command. Therefore, if the last command may be required for error analysis it must be retrieved first.

It should also be noted that the technique of retrieving the last command register assumes the most recent successful operation was not an identical (subaddress, word count, transmit/receive) command. Should back-to-back identical commands be possible, (such as for a very limited use RT), the system designer is responsible for seeing that intervening commands are sent to permit the correct operation of the retry procedures. Generally, a transmit status mode command before each such message will be most reliable.

One of the expected sources of message errors is that a remote device will occasionally be busy when commanded to transmit or receive data. A busy transmitter sends no data and a busy receiver accepts no data, so duplicated messages are not a problem and no confirmation of an error is required. A simple retry of the message is performed, and a count of repeated busy responses on a single message is maintained to prevent an infinite loop.

When a message error persists after the specified number of retries, the error is logged via the configuration management function and the message is switched to the redundant multiplex bus. Note if the message succeeds on the alternate bus, the bus operations simply continue and the switched message remains on the alternate bus. Thus, if one bus side of a terminal is operating in a marginal fashion, messages will migrate to the good bus side as errors are encountered.

An auxiliary function imposed on the message retry procedure is that of NO-OPing messages after a terminal has been declared failed. When a terminal is failed (a decision made by the configuration management function) no attempt is made to remove the instructions addressing the terminal from all bus lists. That would require an extensive search operation or some pointer lists to locate all the affected bus instructions. Performing the NO-OP in the retry procedure distributes the impact of removing a terminal's messages at the cost of an interrupt per message. It is also worth noting that the NO-OPing procedure is operationally almost identical to switching a message to the redundant bus.

4.1.5 Configuration Management

The repeated failure of a bus message is reported to the Configuration Management function for tabulation and analysis. A message which succeeds on immediate retry is not reported. (In fact, the processor is not even aware of the error if an automatic retry succeeds.) Reported errors are tabulated against a terminal address with separate counters accumulating errors on the two redundant communications paths.

The overall interaction of the assorted system functions involved in declaring or responding to a device failure is shown in simplified form in Figure 21. Configuration Management is notified only after repeated errors and then a failure is declared only after a critical threshold is reached or self tests reveal a hard failure.

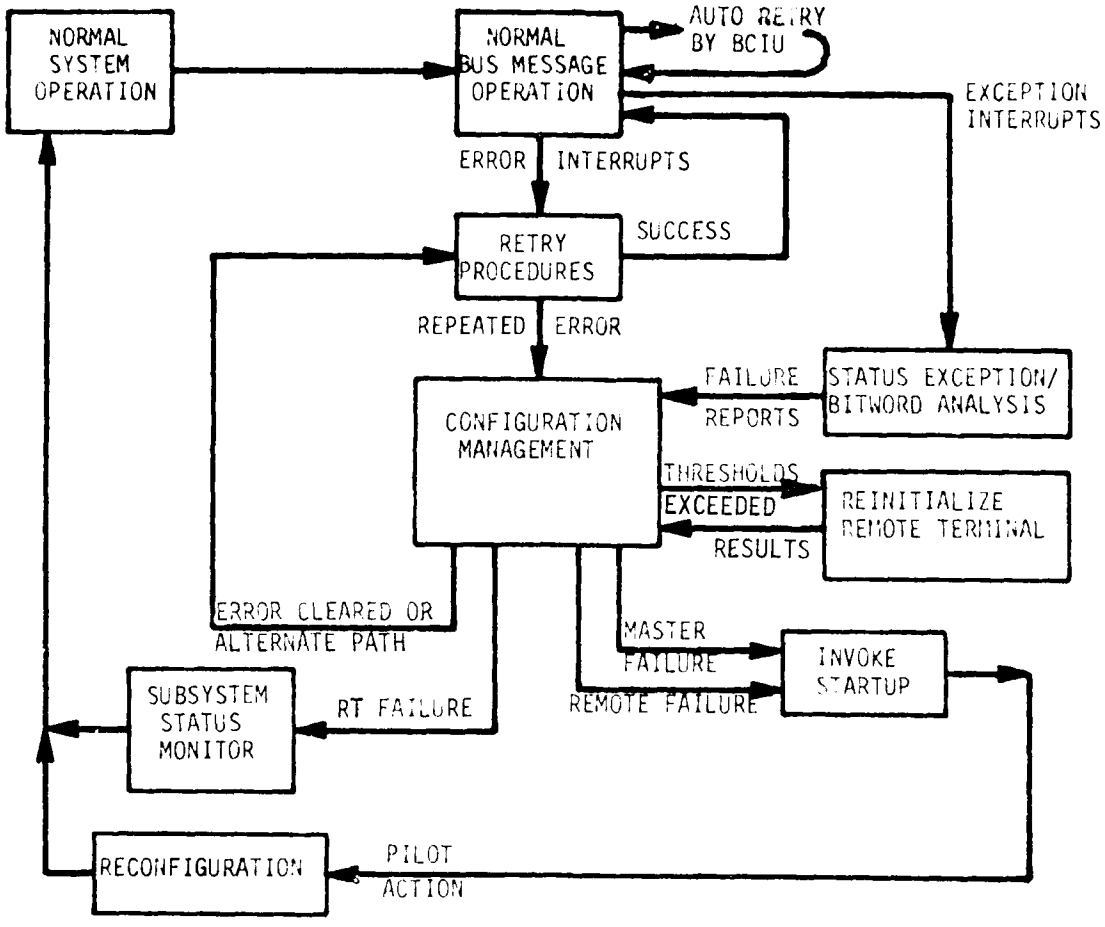


FIGURE 21. CONFIGURATION MANAGEMENT INTERACTIONS

The first task of Configuration Management is to decide what terminal to charge with the error. While determination that an error occurred is primarily based on the receiver's status word (or lack of it), the error is normally attributed to the transmitting terminal. The receiver is charged with the error only in the case that he reports an error when neither the Master processor/BCM nor the transmitting terminal detect any anomaly.

Error counters are maintained by Configuration Management for each bus side of each terminal address. Each time an error is logged against the terminal and bus side the counter is incremented and compared to a user-defined threshold.

When the error count reaches this threshold, the bus side is flagged as "suspect" for this terminal address. Once a bus side is flagged as suspect, the message retry procedures will no longer switch messages to this bus for this terminal. If the threshold is reached for both sides of a terminal, the terminal is reinitialized in an attempt to restore normal operating characteristics.

The reinitialization procedure is represented in Figure 22 and consists of a series of mode commands to reinitialize, perform self test and transmit the BIT word. The BIT word format for an RT is shown in Figure 23. As may be seen, the terminal is capable of detecting a variety of internal problems. The sequence of initialize and perform self tests primarily exercises the TCU and MTU functions. (The individual IM's and channels are tested in background of normal operations after being accessed for data.)

The reinitialize function will analyze the BIT word retrieved and flag each bus as good or failed and report back to configuration management. Should any errors occur in these mode code operations, this will be considered an indication of failure of the bus side of that terminal.

When a good bus indication is returned to configuration management, the terminal will be considered operable on that bus. The appropriate error counter and "suspect" flag will be cleared and normal communication with the terminal will be resumed. Should the error counters again reach the thresholds, the whole sequence will be repeated. To protect itself from excessive overhead, the system will keep count of the number of times a terminal has been reinitialized. If this count reaches a user-defined limit, the terminal will be flagged as failed.

DAIS has not implemented any recovery procedures for failed RTs. Once a terminal is flagged as failed, messages to that terminal are NO-OPed as errors are encountered and all communication with the RT ceases. Message traffic is restored only by a system reload and cold initialization.

Message errors which are attributed to a remote processor will be counted against each bus side and the count compared to a threshold as described above. One bus side may be flagged as "suspect" and message traffic will migrate to the other bus. However, if a remote processor becomes suspect on both buses, it is failed immediately and the startup/restart function is invoked.

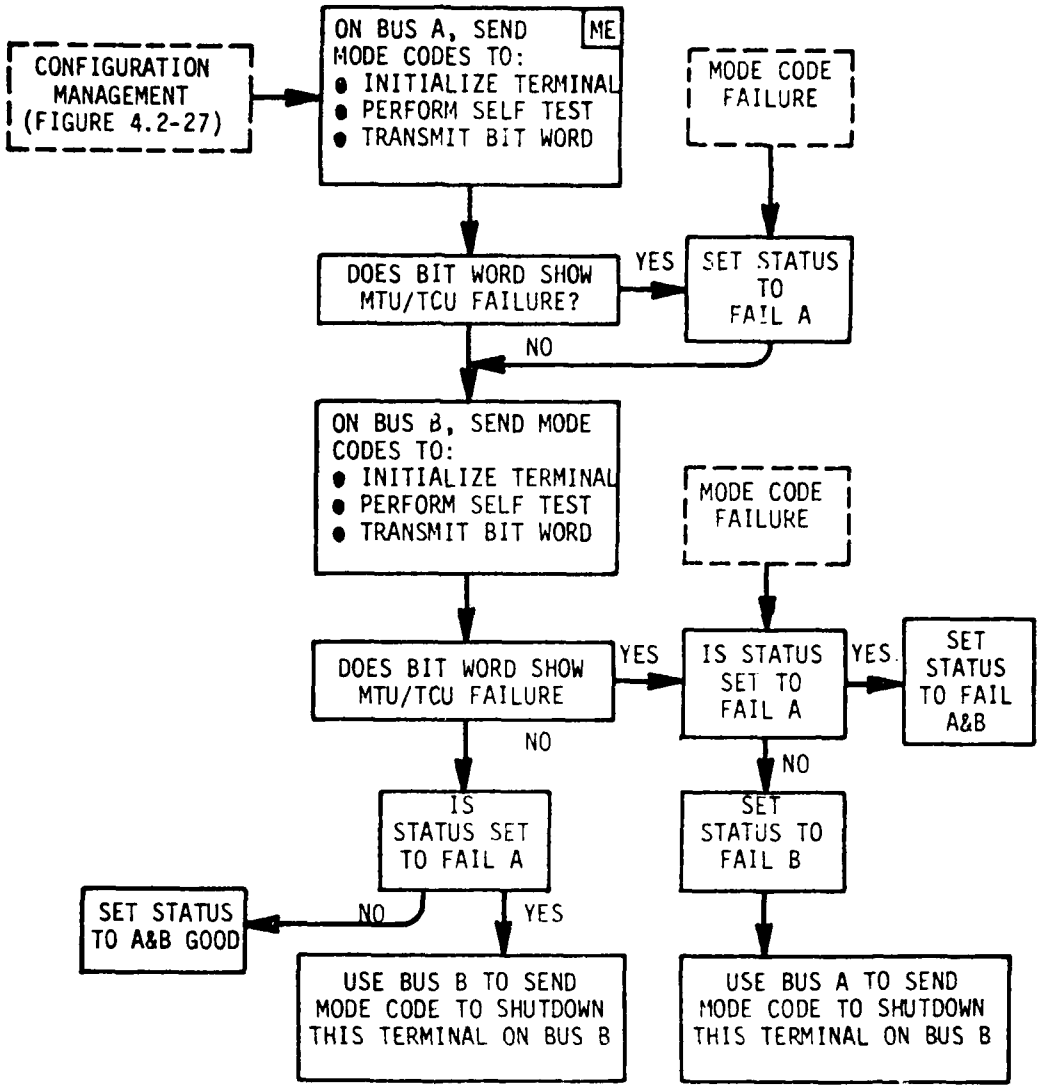
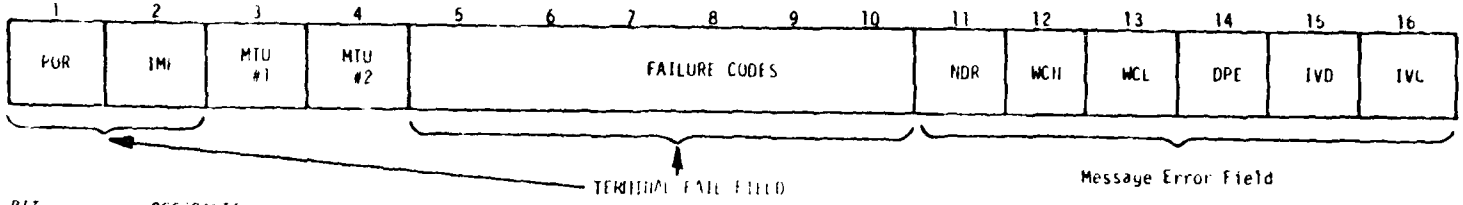


FIGURE 22. REINITIALIZE RT



BIT	DESCRIPTION
1	1 = Power-On-Reset (POR)
2	1 = Interface Module Failure
3	1 = MTU #1 Power Out
4	1 = MTU #2 Power Out
5-10	Failure Codes

BITS 5 6 7 8 9 10 (IMF=0)*

0 0 0 0 0 0	No Failure
0 0 0 0 0 1	MTU Failure
0 0 0 0 1 0	Not Used
0 0 0 0 1 1	Not Used
0 0 0 1 0 0	TCU Data Flow Error
0 0 0 1 0 1	TCU DROM Error
0 0 0 1 1 0	TCU Sequencer Error
0 0 0 1 1 1	TCU A/D
0 0 1 0 0 0	Converter Error
0 0 1 0 0 1	NOT USED
0 0 1 0 1 0	
0 0 1 0 1 1	
0 0 1 1 0 0	
0 0 1 1 0 1	
0 0 1 1 1 0	
0 0 1 1 1 1	

11	1 = No Data Received (NDR)
12	1 = Word Count High (WCH)
13	1 = Word Count Low (WCL)
14	1 = Data Parity Error (DPE)
15	1 = Invalid Data (IVD)
16	1 = Invalid Mode Code (IVC)

*If IMF=1, then failure code field indicates failed IM/Channel, such that bits 5-6 is channel and bits 7-10 is IM.

FIGURE 23. BUILT-IN-TEST (BIT) WORD - RT

The DAIS executive software architecture (a virtual memory approach) does not permit continued operation in the event of a processor failure. A reinitialization sequence is not done for processors since the startup/restart function provides a more comprehensive verification of both hardware and software and, in the event of a solid failure, gives the pilot better visibility into what equipment is operable, thereby facilitating reconfiguration.

In addition to error reports from message retry, the configuration management function may receive failure reports from the status exception/BIT word analysis function. This function is invoked when a status word returned during a message operation contains an exception condition other than M/E, SR, or busy. The exception is normally indicated by the T/F bit (devices with other exception indicators are known as "strangers") and more detailed information is contained in the BIT word.

The BIT word format for a BCM is shown in Figure 24. Since self tests are not performed for BCMs during normal operation, the power-on-reset is the only condition that will cause a T/F bit to be set for a remote processor. This is treated as a fatal condition and the processor failed immediately. In practice, this will not occur, since the loss of power by a remote will result in accumulated message errors and the recovery of power will cause the processor to cycle through the startup/loader which clears the BIT register prior to enabling for remote communications.

The flow of the status exception/BIT word analysis is shown in Figure 25. When the exception is from an RT, the BIT word is retrieved for analysis. Either a bus side (MTU or TCU) failure is reported or an IM/channel failure is reported to configuration management. The IM/channel failures are separately counted for information purposes, but otherwise either exception is cleared by reinitializing the terminal. Again, an excessive number of initializations will result in terminal failure.

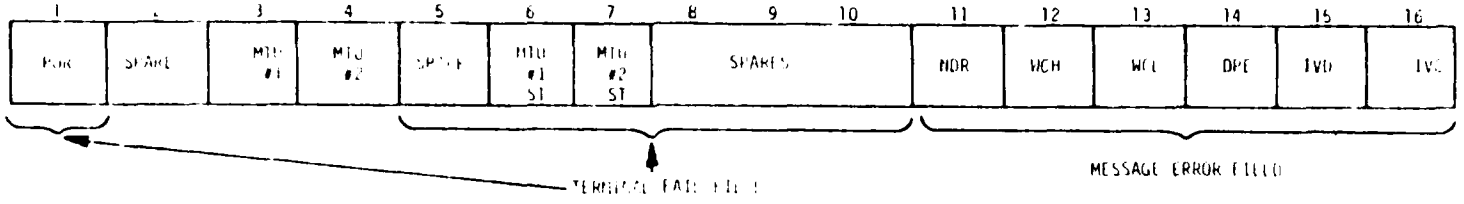
When an entire device is flagged as failed, Configuration Management will perform one of several actions depending on the type of device. If a remote terminal is failed, Mission Application Software will be notified by way of the Subsystem Status Monitor, so that appropriate action may be initiated.

When any processor fails, configuration management invokes the system startup procedures and all processors perform hardware and software tests to determine operability. Outputs to the PCP/ACP lights advise the pilot of the hardware/software status.

Note that DAIS has not implemented a Monitor function. Should the Master fail in such a way that the above functions cannot be performed, the system will freeze and manual intervention by the pilot is required.

4.2 DAIS Architecture Features

The DAIS architecture possesses specific characteristics which facilitate the adapability of the system to various applications and support effective incremental integration and test of the DAIS core elements. These key characteristics are discussed in the following sections.



BIT	DESCRIPTION	
1	1 = Power-On Reset (POR)	
2	SPARE	
3	1 = MTU#1 Power Out	
4	1 = MTU #2 Power Out	
5	Spare	
6	BCM/MTU1 Self Test	
7	BCM/MTU2 Self Test	
8	Spare	
9	Spare	
10	Spare	
11	1 = No Data Received (NDR)	REMOTE MODE ONLY (RECEIVING DATA WORDS)
12	1 = Word Count High (WCH)	
13	1 = Word Count Low (WCL)	
14	1 = Data Parity Error (DPE)	
15	1 = Invalid Data (IVD)	
16	1 = Invalid Mode Code (IVC)	

FIGURE 24. BUILT-IN-TEST (BIT) WORD - BCM

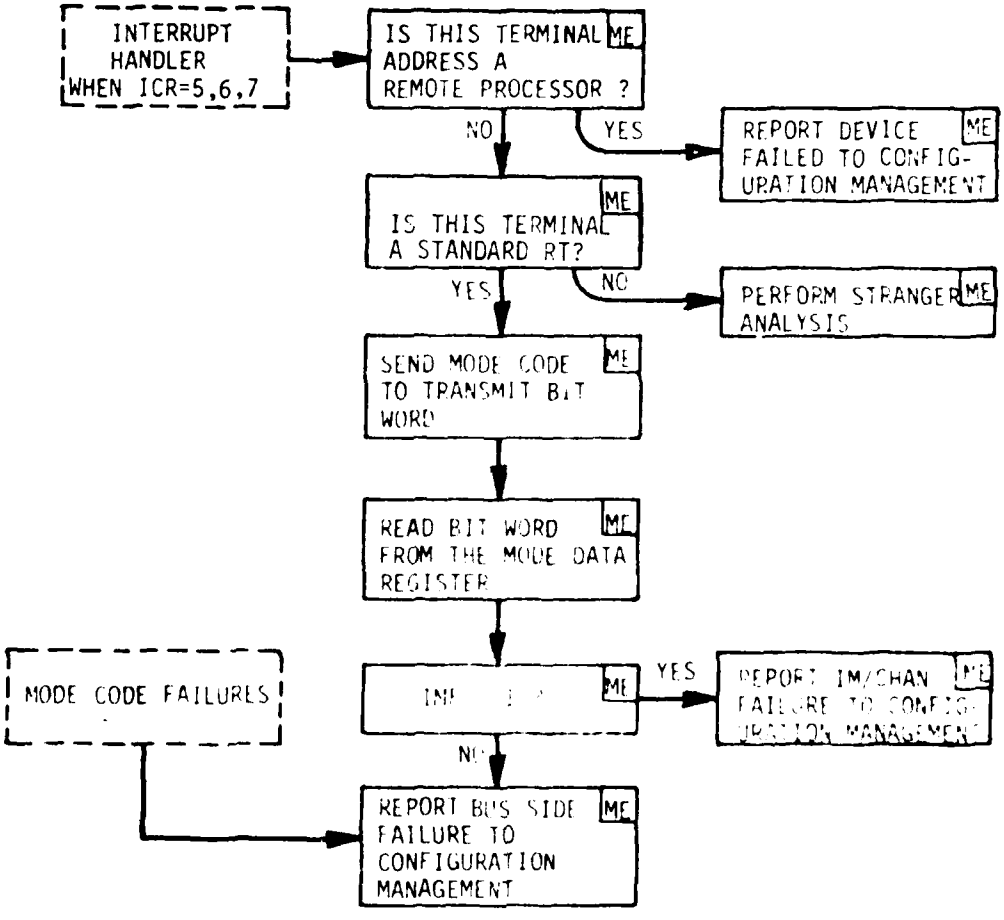


FIGURE 25. STATUS EXCEPTION/BIT WORD ANALYSIS

Specific features in the DAIS system which allow adaptation of the core elements to a specific avionics system configuration are presented below.

4.2.1 Bus Devices

The basic message structure, data format, and command-response protocol required for the multiplex system is described in Section 4.1.2. The message protocol is defined as that sequence of bus messages required to perform normal multiplex synchronous and asynchronous operations and to support the message error detection and recovery process. The MIL-STD-1553B definition of the command format allows up to 32 different addressed devices to be implemented on the multiplex system. DAIS has restricted this definition to allow up to four processors/BCMs and 28 remote terminal devices.

The system designer defines the specific configuration (number of processors/BCMs, number of RTs to interface with the avionic sensors and subsystems using the standard Interface Modules (IMs) or unique IMs if required). The system designer then defines the sensor inputs/outputs as data bus messages and inputs this information to PALEFAC. PALEFAC generates the Executive bus instruction list tables accordingly. This directs the messages to the proper processor containing the application software (e.g. EQUIPS module) which processes the sensor/subsystem data.

The DAIS architecture is readily adaptable to various numbers of devices on the bus. It provides the tools to automatically generate and direct the messages to the proper application software in one of the processors.

4.2.2 Processor/Bus Controllers

The architecture provides the capability to expand or reduce the number of processors/BCIs for a specific application. The system designer considers trade-offs in determining the number of processors/BCIs required. Mission requirements, reliability and availability must be considered.

The system designer defines the Mission Software application modules based upon the mission requirements and avionic suite. The application programmer develops the application modules using the J73 language and mission software standards. Based upon the application execution times, memory size, data access, comsubs, and the bus message traffic, the system designer then partitions the application tasks among the processors using PALEFAC. PALEFAC produces the tables required for run time execution of the application software in the specific configuration of processors/bus controllers. These tables are used by the master executive and the local executive.

4.2.3 Remote Terminal (RT)/Interface Modules (IMs)

The RT has a modular and programmable design. This allows flexible partitioning of the data messages to the appropriate sensors/subsystems and signal conditioning required by different numbers and mixes of subsystem signals. This is accomplished as follows:

The Timing and Control Unit (TCU) in the RT performs all of the timing, control, buffering, decoding, and checking required to receive or transmit information from the data bus and transfer that information as outputs or inputs from the RT via the Interface Modules (IM). The TCU contains a programmable device which controls the mapping of each data word in the RT. The interface between the TCU and IMs is standardized and contains the signals required to allow TCU to select the individual modules. Therefore, all IM slots can accept all IM types.

Each RT will interface with different numbers and mixes of sensor/subsystem signals for each specific system configuration. This is accomplished by inserting the proper number and type of interface modules into the IM slots in the RT housing and by programming the ROM in the TCU. If required, a special interface module can be designed to interface with an existing subsystem having a unique interface.

The functions of the RT can also be embedded in a sensor or subsystem so that the interface of the sensor or subsystem is directly with the multiplex data bus. Functionally, the embedded RT responds to commands received on either data bus in the same way as an RT.

4.2.4 Application Software

The DAIS System Architecture is designed to allow modular implementation of specific systems by using the required elements of the DAIS system, both hardware and software. First, the set of multiplex equipment for interfacing to the external world (sensors and Controls and Displays equipment) must be chosen. Then, the design of the processing system can begin. The Application Software is initially designed as if it will exist in one large virtual processor. This virtual processor has as much memory space and execution time available as the sum of the flight processors to actually be used in the system. After the data interfaces have been defined to the outside world (multiplex bus messages) and the functional performance required of the application software has been identified, the layout of Compool Blocks and Tasks can begin.

Compool Blocks are the data communication paths between the application software and the external world and among the application software tasks. Tasks are the processing elements in the application software which collectively perform the avionics processing function. Tasks access the Compool Blocks through calls upon Executive services (such as READ and WRITE) and operate upon the contents of the Compool Blocks for processing purposes.

After the functions of the application system have been designed in terms of Tasks and Compool Blocks, and the Application Software has been tested on the host processor, partitioning of the application system onto the flight processors begins. This partitioning is illustrated in Figures 26 and 27 for a two and three processor system configuration, respectively.

The application software can be easily partitioned across a set of flight processors by using the DAIS Executive and PALEFAC. The DAIS Executive provides general I/O control for all communications on the multiplex bus (between processors and to the external world). PALEFAC builds the data base which is then used by the Executive to control system communications.

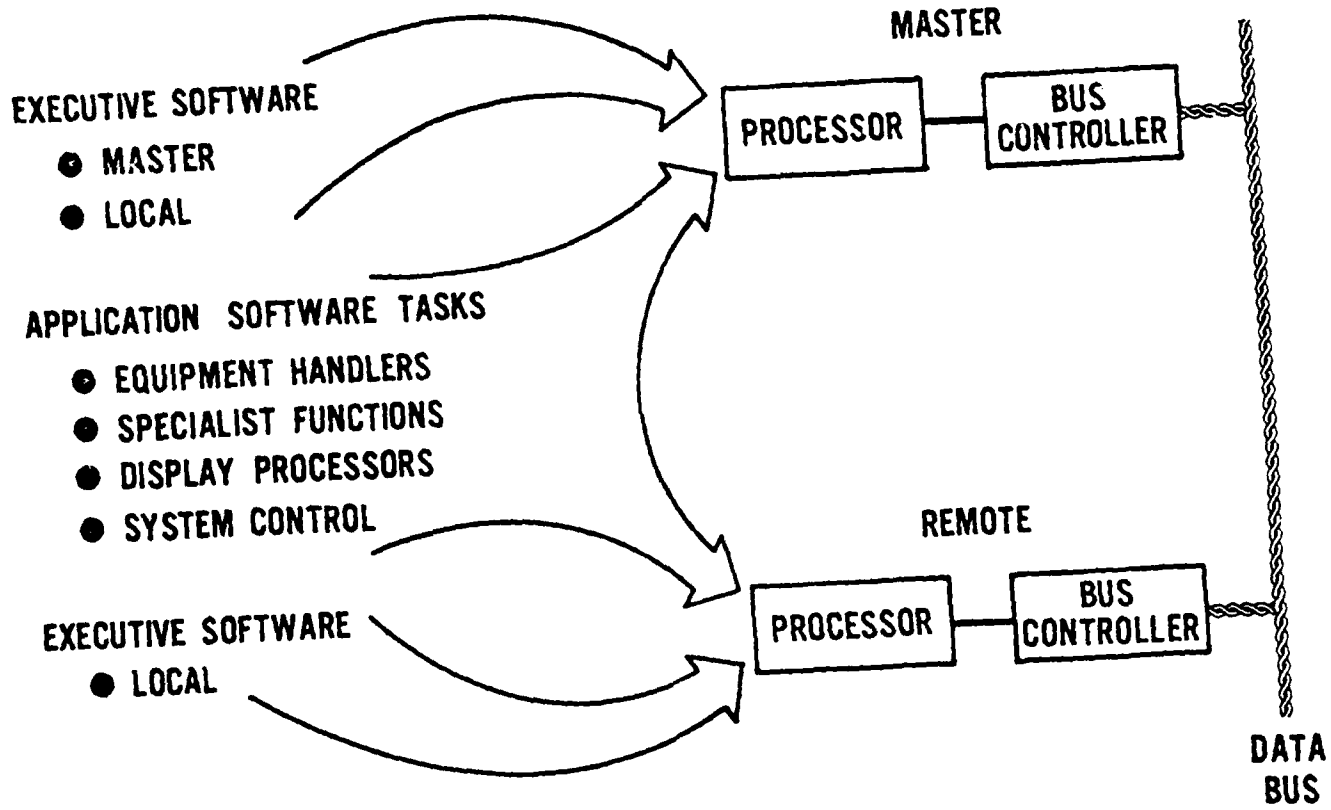


FIGURE 26. MISSION SOFTWARE PARTITIONING - TWO PROCESSORS

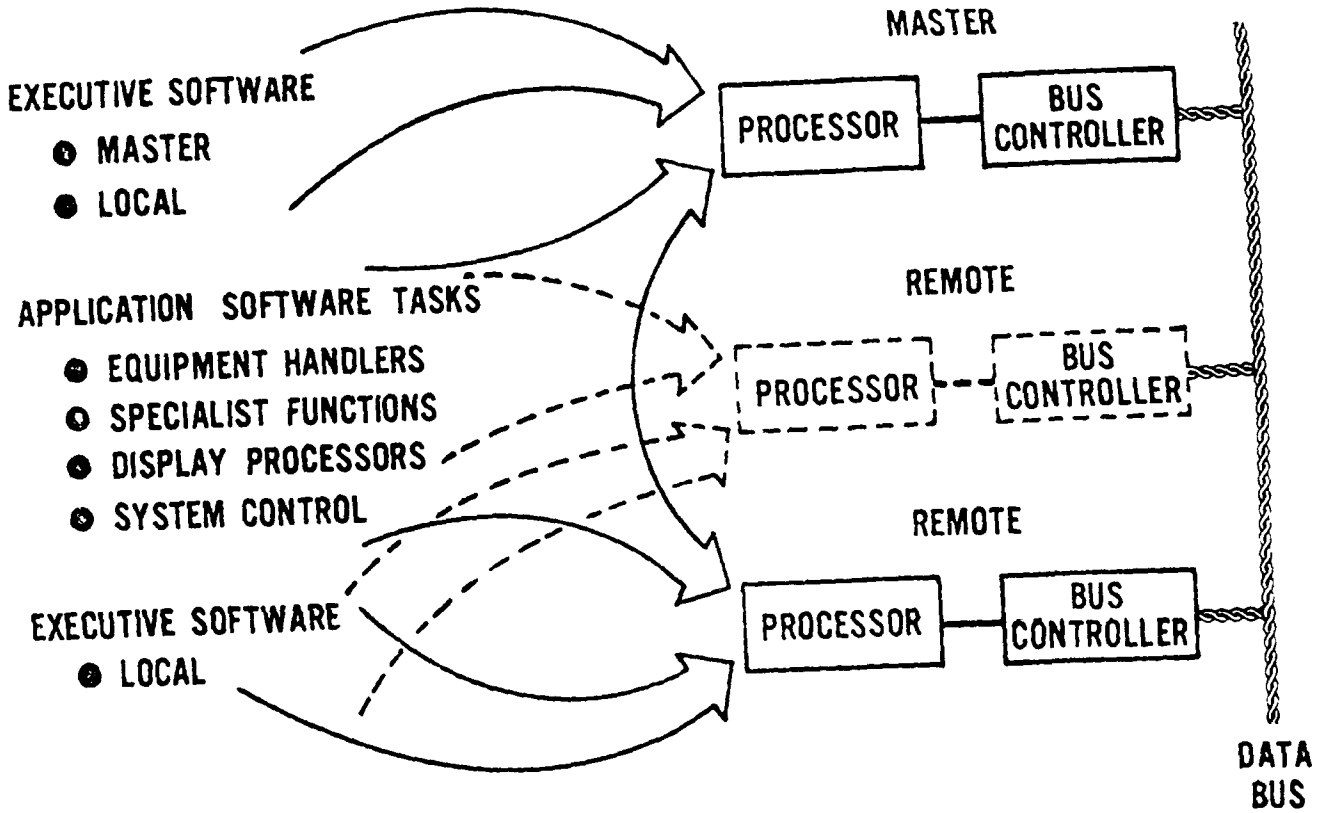


FIGURE 27. MISSION SOFTWARE PARTITIONING - THREE PROCESSORS

The use of the Executive and PALEFAC results in an automatic partitioning of the Compool Blocks onto processors based upon the partitioning of Tasks and upon which Tasks use which Compool Blocks. If a Compool Block is used for communications between tasks and if the tasks are split into two or more processors, the system will automatically generate a copy of the Compool Block at each processor; it will also generate the multiplex bus message to update all other copies when one copy is updated.

The system will also automatically generate the multiplex bus messages required to connect the Compool Blocks to the devices on the multiplex bus. All multiplex bus messages may be declared as synchronous with a period and phase. If not so specified, the system will automatically generate asynchronous updates of the bus messages when a copy is updated.

4.2.5 Interface Standards

The DAIS was converted from MIL-STD-1553A to MIL-STD-1553B.

4.2.5.1 DAIS/1553B Message Protocol

As discussed previously, MIL-STD-1553B defines the basic message structure and command-response protocol. In order to properly handle detected message errors and other DAIS system features, DAIS developed a set of procedures that define bus operations and message error and terminal failure detection and recovery processes. If a sensor/subsystem device's bus interface conforms to this composite DAIS/1553B protocol, it would be fully compatible with the DAIS System Architecture.

4.2.6 Portability

The DAIS Mission Software has, from the very beginning, been conceived as an easily retargetable system. The single most important feature that allows this capability is the development of more than 97% of the DAIS Executive Software in JOVIAL J73 (The only routines written in assembly language are processor-dependent I/O operations, register manipulation operations and lowest level interrupt handlers). All of the DAIS Application Software is implemented in JOVIAL J73. A striking demonstration of the portability of the DAIS Mission Software is the fact that it executes both on the DAIS processor and on the DECsystem-10, the host processor facility. In fact, stand-alone module testing and initial subsystem integration testing of Application Software is performed on the host processor. This is possible since the JOVIAL J73 Compiler has a code generator for both the DECsystem-10 and the DAIS processor, which points out the ease of retargetability of the DAIS Software System.

4.2.7 Redundancy

DAIS provides redundancy in the system architecture. This can be employed to provide backup and recovery to complete mission functions in spite of hardware failure. The areas which can be used for redundancy are:

a. Dual redundant data bus includes redundant bus interface modules in the BCIU and RT.

b. System can employ dual redundant RTs for a specific subsystem.

c. Multifunction keyboard and associated Data Entry Keyboard can be used as a backup to the Integrated Multifunction Keyboard and associated Data Entry Keyboard.

d. Raster displays can serve as backup to each other.

5.0 SUPPORT FACILITY

The purpose of the AVSAIL support facility is to test and evaluate the DAIS core element architecture and a representative set of sensors/subsystems. The support facility provides a real-time simulation of a military aircraft performing an operational mission. The simulation generates the interface signals between the simulated aircraft sensor suite and the DAIS system. The avionic equipment is thus subjected to a data signal environment which is nearly identical to actual flight. A simulated cockpit is included as part of the simulation for realistic evaluation of the avionic system.

The support facility for DAIS consists of a Software Test Stand (STS) and an Integrated Test Bed (ITB). Initially, the Software Test Stand provided a capability to test the mission software resident in the DAIS flight processors (AN/AYK-15). The Integrated Test Bed provided the capability to test the DAIS core elements which included the mission software, the DAIS processors (AN/AYK-15), the multiplex bus system (MIL-STD-1553A) and the cockpit controls and displays. Currently, both the STS and ITB use a MIL-STD-1553B multiplex bus system. The flight processors have been changed to the AN/AYK-15As on the STS only. Therefore, the STS provides the capability to test the DAIS core elements which include new mission software, the DAIS AN/AYK-15A processors, the MIL-STD-1553B multiplex bus system and the cockpit controls and displays. The ITB's flight processors are still the AN/AYK-15s, so the ITB is used for testing hardware and running simulations. Both the STS and ITB utilize a complex of digital computers consisting of a Digital Equipment Corporation (DEC) DECsystem-10 and a number of DEC PDP 11 series computers.

Simulation software resident in the DECsystem-10 was developed to provide real-time simulation of a military aircraft in an operational environment. The aircraft dynamics, the aircraft sensors and weapon targets are simulated. The simulation is driven from the cockpit by an operator acting as a pilot. The simulated cockpit is equipped with Controls and Displays so that the various modes of a mission may be flown by the pilot with an out-the-window background scene.

The support hardware and software provides interfaces between the DAIS elements under test and the host simulation software. The support facility provides the capability to set up a test, conduct a simulation, and record data from both the DAIS elements and the simulation. During a simulation, sufficient test data is displayed in real-time to indicate that the system is operating satisfactorily. It provides presentation of test results for observing systems performance.

The support facility is composed of:

1. STS Support Hardware
2. STS Support Software and PDP-11 Processors
3. ITB Support Hardware
4. ITB Support Software and PDP-11 Processors
5. Host Simulation Processor, DECsystem-10
6. Simulation Software
7. System Test Software
8. Picture System
9. Picture System Software

5.1 System Configuration

The initial support facility functional diagram is shown in Figure 28. It consists of STS & ITB hardware and software both sharing the Host Simulation Processor and the DAIS cockpit. The current support facility functional diagram is shown in Figure 29. Support software is resident in the DECsystem-10 and in the PDP-11 Processors.

5.1.1 Integrated Test Bed (ITB)

The Integrated Test Bed (ITB) block diagram is shown in Figure [illegible]. It is composed of the support facility and DAIS core elements. The support facility simulates all the equipment and environment external to the DAIS processor using the resident mission software, DAIS multiplex system, and DAIS controls and displays. The ITB support facility is composed of support hardware, support software resident in the PDP-11 processor, the simulation software resident in the host simulation processor, the DAIS simulated cockpit, and an out-the-window picture system.

The support hardware provides the interfaces between the simulation and the DAIS core elements. The interface includes: 1) the multiplex bus messages which drive the mission software, 2) the performance monitoring of the multiplex bus traffic and the internal operation of the mission software in the DAIS processors, 3) the cockpit controls and backup instruments, and 4) the out-the-window picture.

The support software controls and manages the support hardware and provides the means to link the DECsystem-10 simulation software with the DAIS elements. The support software also provides the performance monitor and control functions which are related to test set-up, control, and data collection for post-test analysis.

The host simulation processor is a commercial DEC system-10 complex. The simulation software generates the real world environment external to the DAIS processors in real time. The simulation is driven by the DAIS cockpit flight controls so that an operator can fly the simulation.

An out-the-window picture system provides a symbolic background scene outside of the cockpit so that the pilot operator will experience the visual orientation of flight with respect to IP's, targets and runways. The picture system also provides the option for displaying the HUD symbology overlayed on the background scene.

A pictorial representation of ITB is shown in Figure 31.

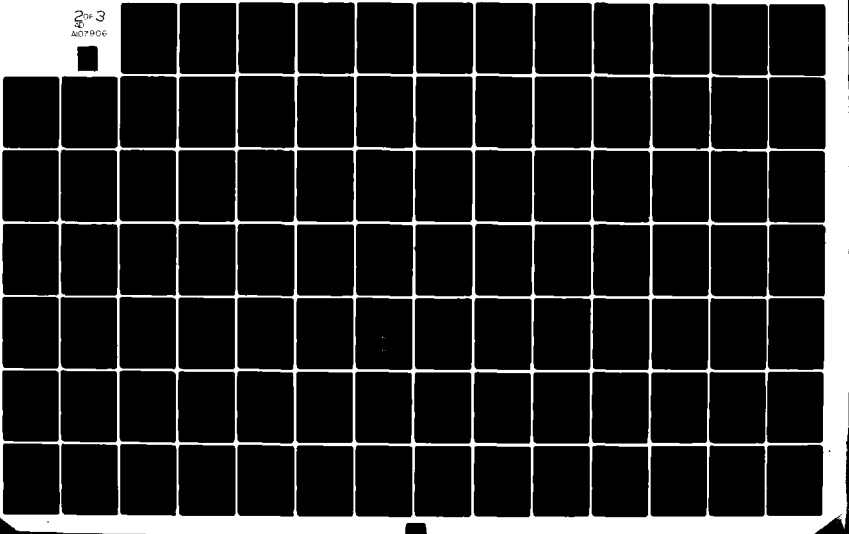
A set of system test software is resident in the host simulation processors, the ITB PDP-11 processors, and the DAIS processors. It performs the system readiness test prior to performing operational runs and diagnosing faults.

5.1.2 Software Test Stand (STS)

A block diagram of the initial STS is shown in Figure 32. The STS system was similar in structure to the ITB. It shared utilization of the DAIS cockpit and controls and displays, and real-time simulation software on the DECsystem-10 complex. Common support hardware and software were [illegible]

AD-A107 906 TRW DEFENSE AND SPACE SYSTEMS GROUP REDONDO BEACH CA F/G 9/2
DIGITAL AVIONICS INFORMATION SYSTEM (DAIS): DEVELOPMENT AND DEM--ETC(U)
SEP 81 M J COOK, R C MASON, J L STAUTBERG F33615-78-C-1502
UNCLASSIFIED AFWAL-TR-81-1165 NL

2 OF 3
AD7906



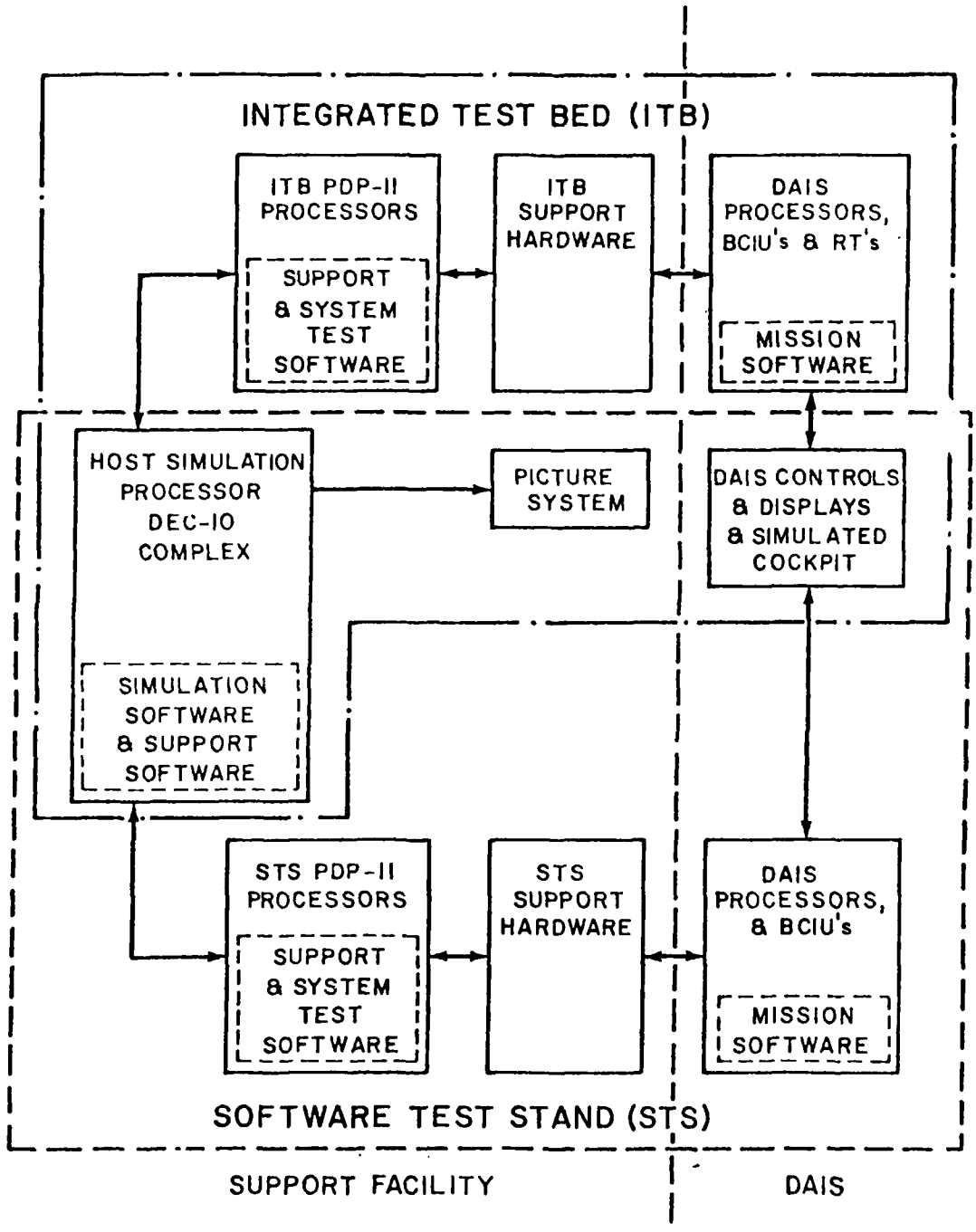


FIGURE 28. INITIAL SUPPORT FACILITY FUNCTIONAL DIAGRAM

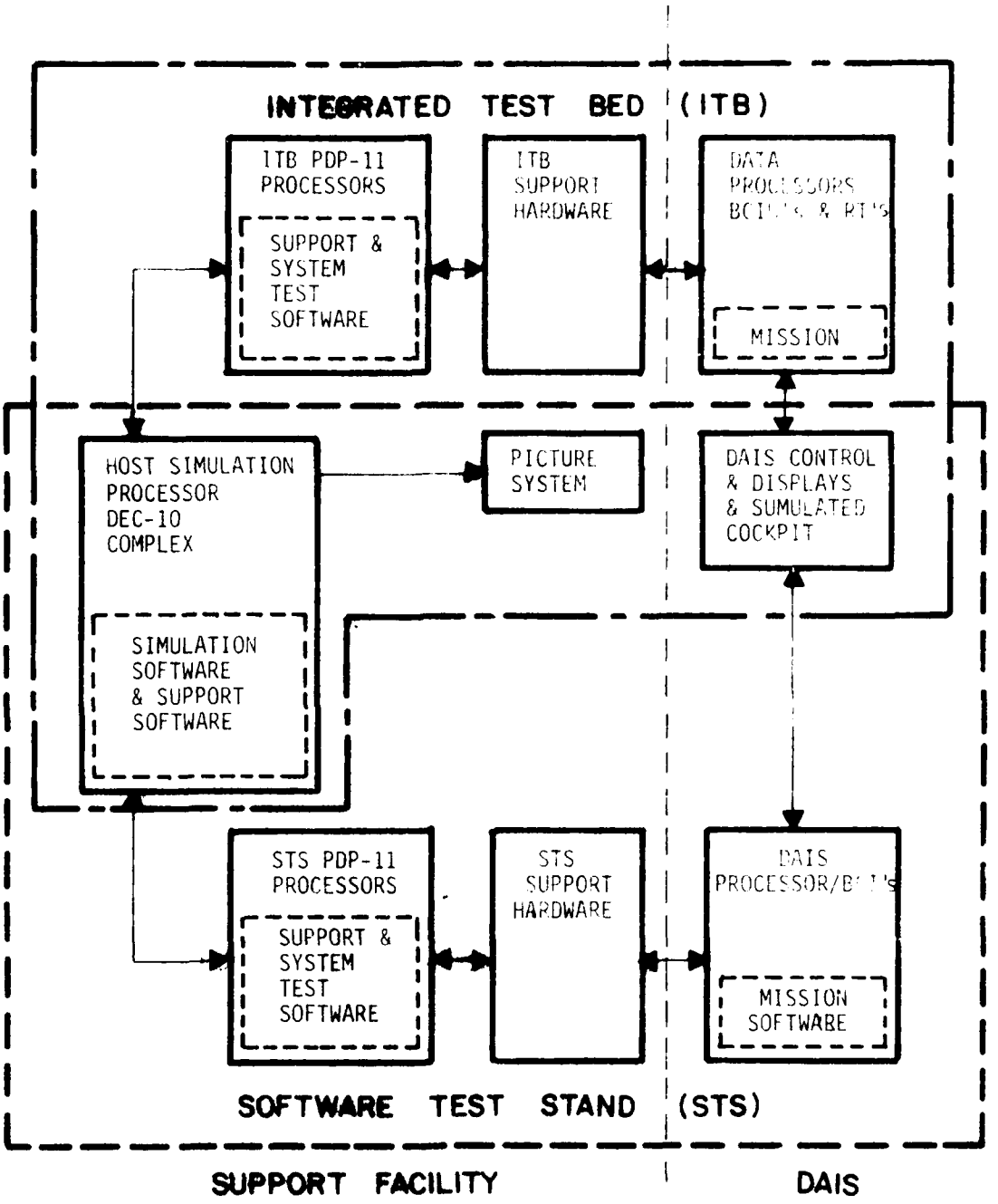


FIGURE 29. CURRENT SUPPORT FACILITY FUNCTIONAL DIAGRAM

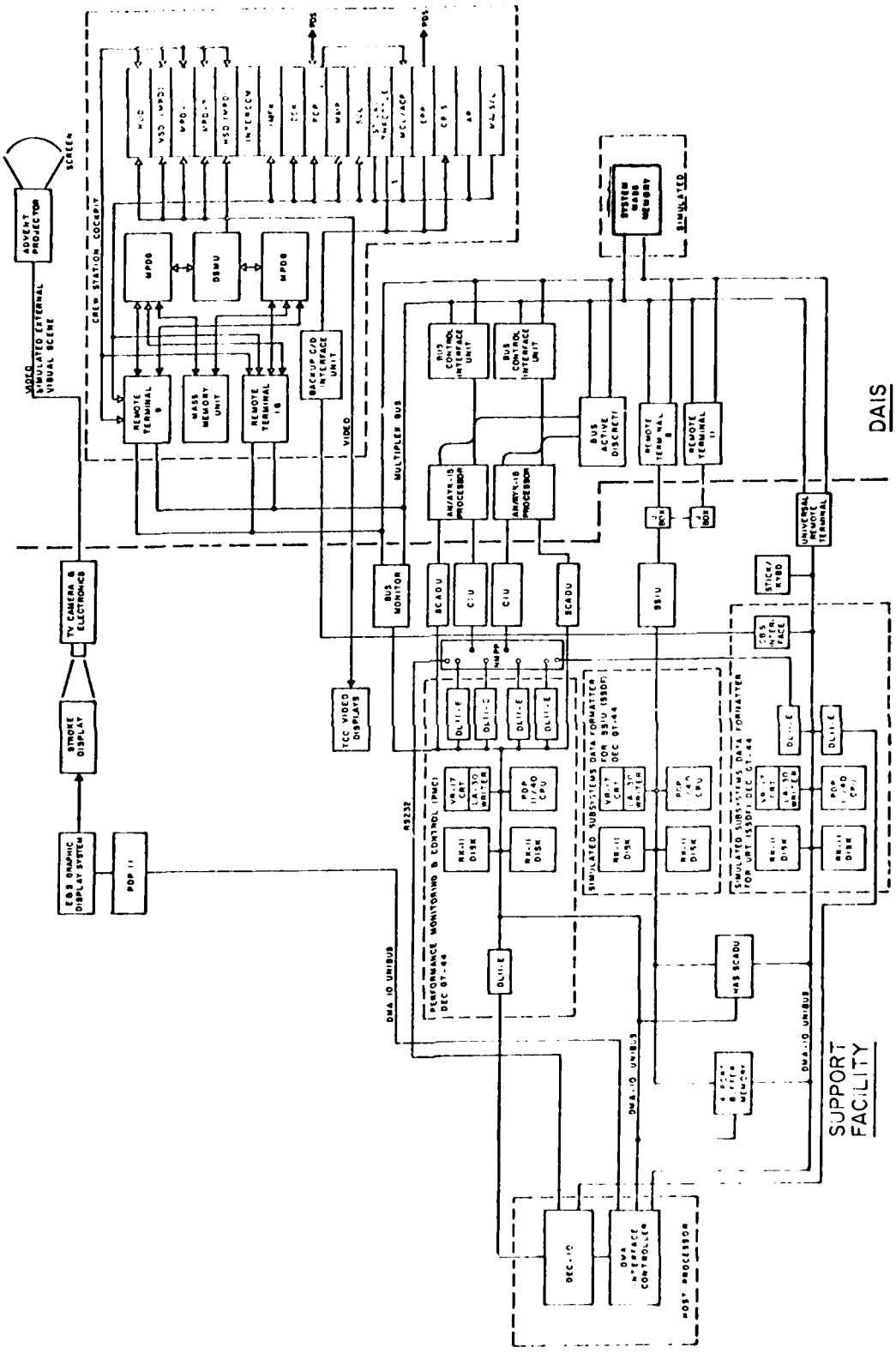


FIGURE 30. INTEGRATED TEST BED (ITB) FUNCTIONAL BLOCK DIAGRAM

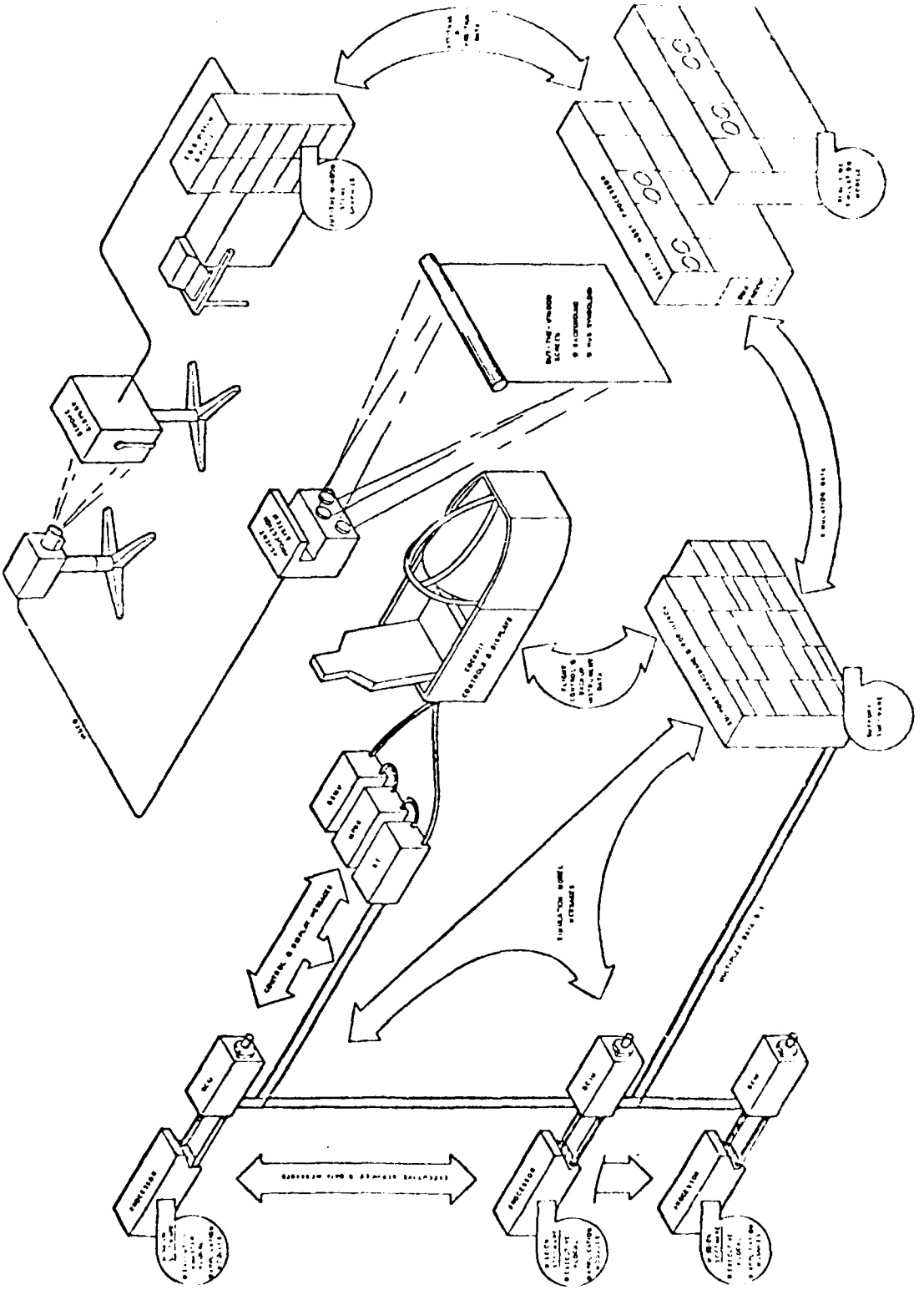


FIGURE 31. INTEGRATED TEST BED PICTORIAL DIAGRAM

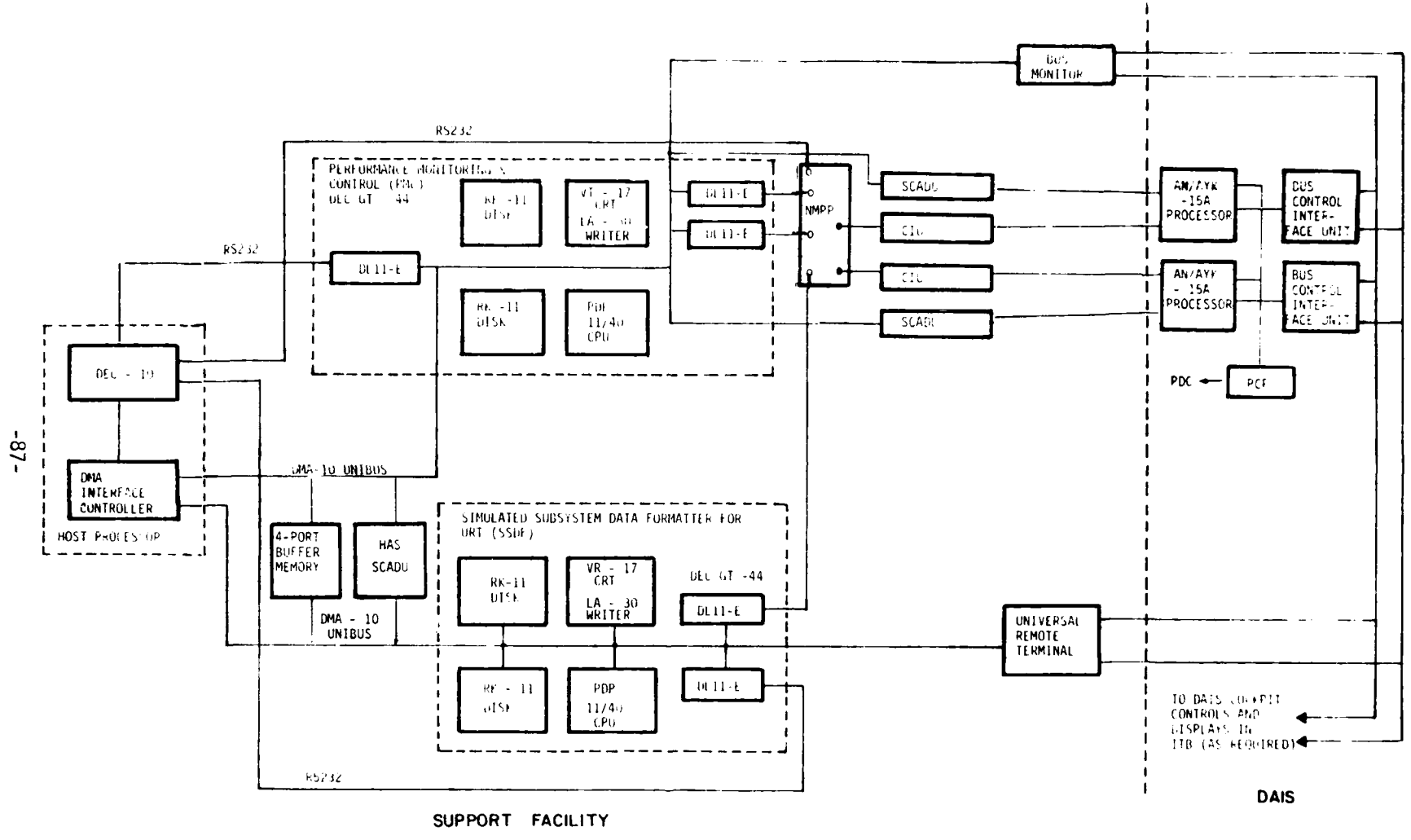


FIGURE 32. INITIAL SOFTWARE TEST STAND (STS) FUNCTIONAL BLOCK DIAGRAM

both ITB and STS. The initial STS was generally utilized to test Mission Software and the support software when operation with the real-time simulation models and DAIS cockpit was not required.

A block diagram of the current STS is shown in Figure 33. The current STS and the initial STS have the following differences:

a. The multiplex bus system was changed from MIL-STD-1553A to MIL-STD-1553B.

b. The flight processors were changed from AN/AYK-15s to AN/AYK-15As.

c. The SCADUs were replaced by the PMIUs.

d. The CIUs were eliminated.

5.1.3 Physical Configuration

The support facility is configured as laboratory equipment. The floor layout and rack/console configuration for STS and ITB are shown as follows:

a. Initial STS/ITB Floor Layout - Figure 34

b. Current STS/ITB Floor Layout - Figure 35

c. Initial STS Test Control Center - Figure 36

d. Current STS Test Control Center - Figure 37

e. Initial STS Equipment Racks - Figure 38

f. Current STS Equipment Racks - Figure 39

g. ITB Test Control Center - Figure 40

h. Initial ITB Equipment Racks - Figure 41

i. Current ITB Equipment Racks - Figure 42

5.2 ITB Support Hardware

The ITB hardware consists of the following subsystems:

5.2.1 Universal Remote Terminal

The Universal Remote Terminal (URT) simulates the operation of a set of up to 32 remote terminals. The URT receives or transmits data between the DAIS multiplex data bus and the PDP 11/40 processor. The URT provides the same interface to the DAIS multiplex as a Remote Terminal, except the URT simulates up to 32 RTs.

The URT is set up and controlled by the SSDF software in the PDP 11/40 by loading the URT registers and RAMs. In real-time operation, the URT performs the following:

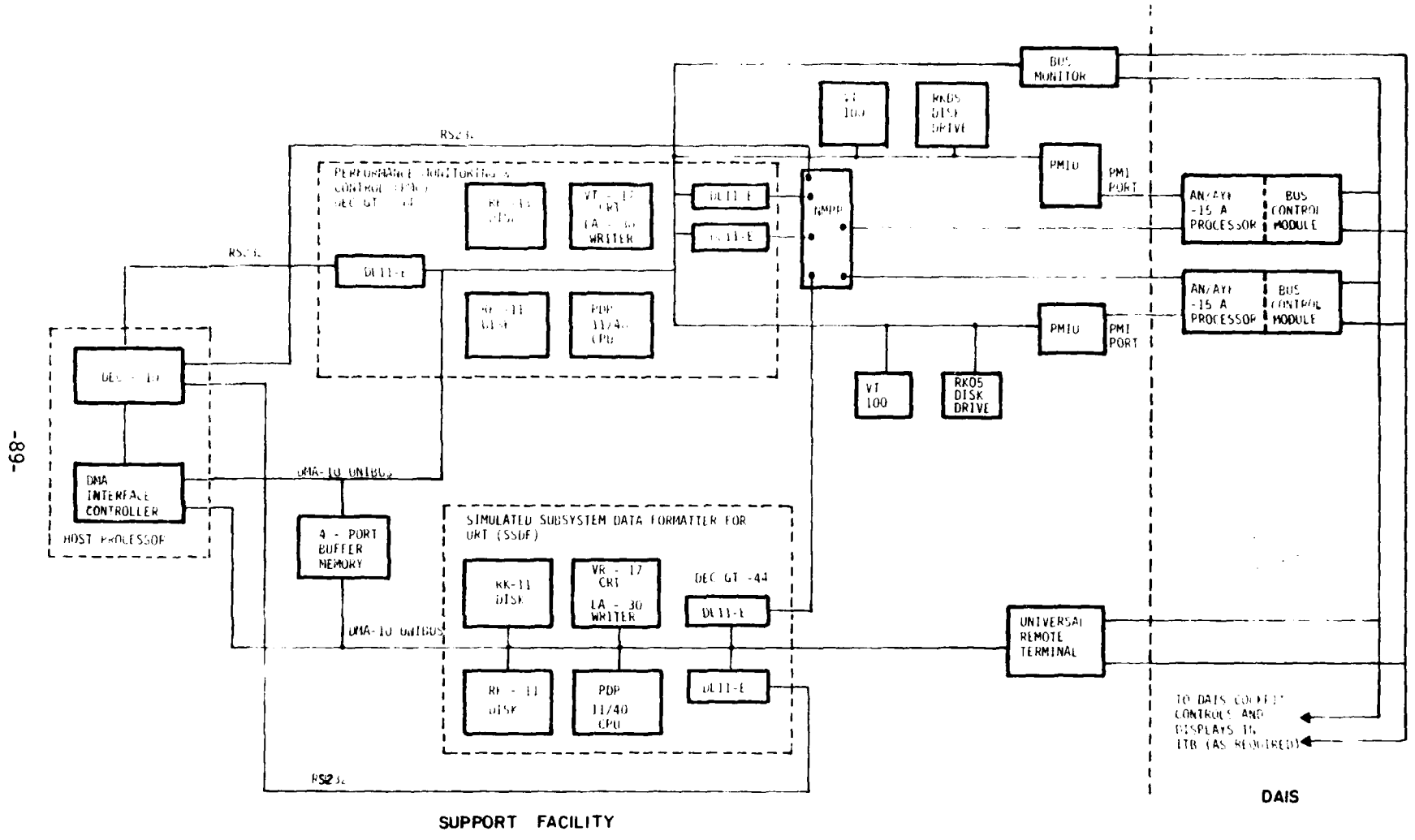


FIGURE 33. CURRENT SOFTWARE TEST STAND (STS) FUNCTIONAL BLOCK DIAGRAM

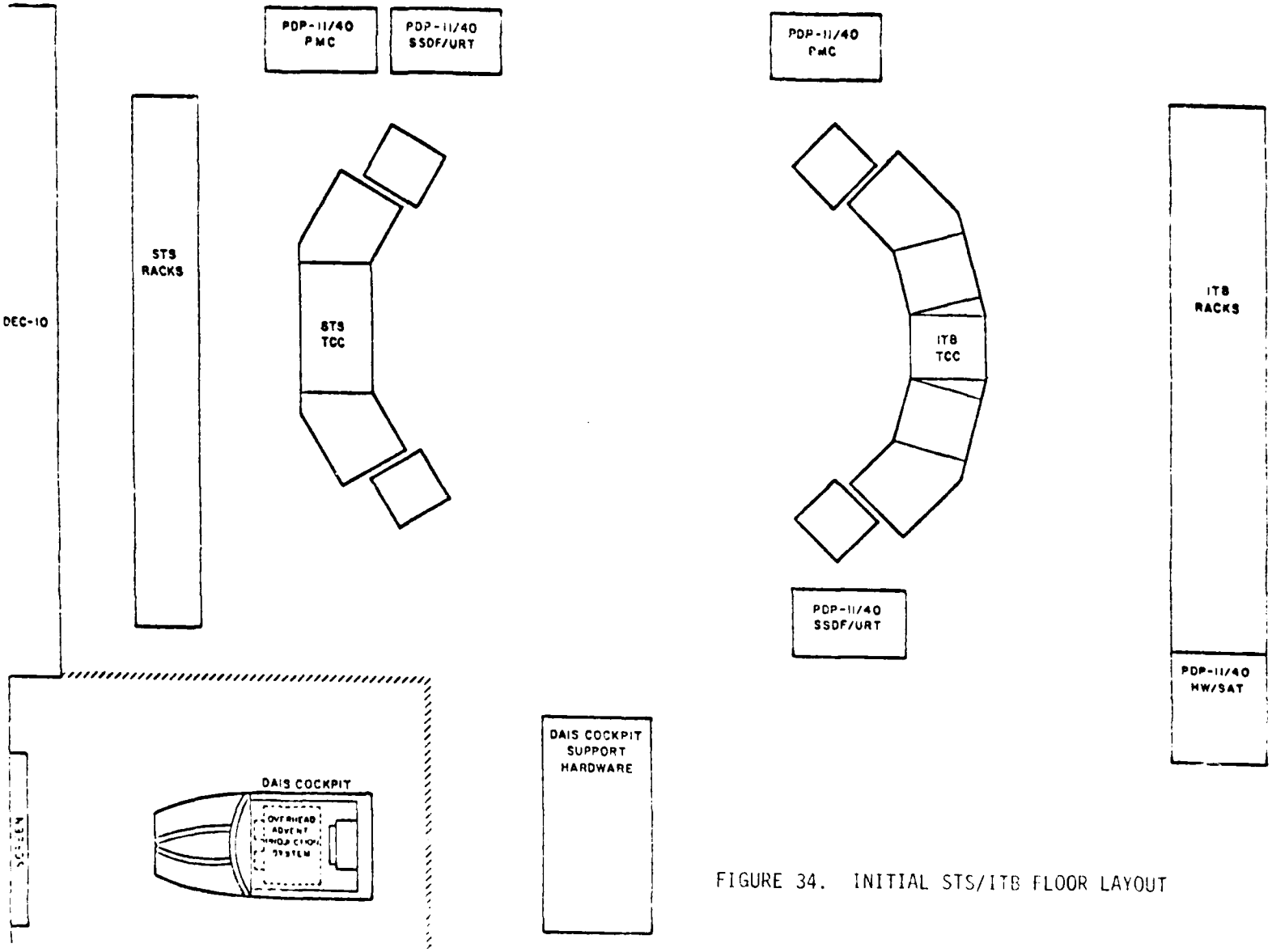


FIGURE 34. INITIAL STS/ITB FLOOR LAYOUT

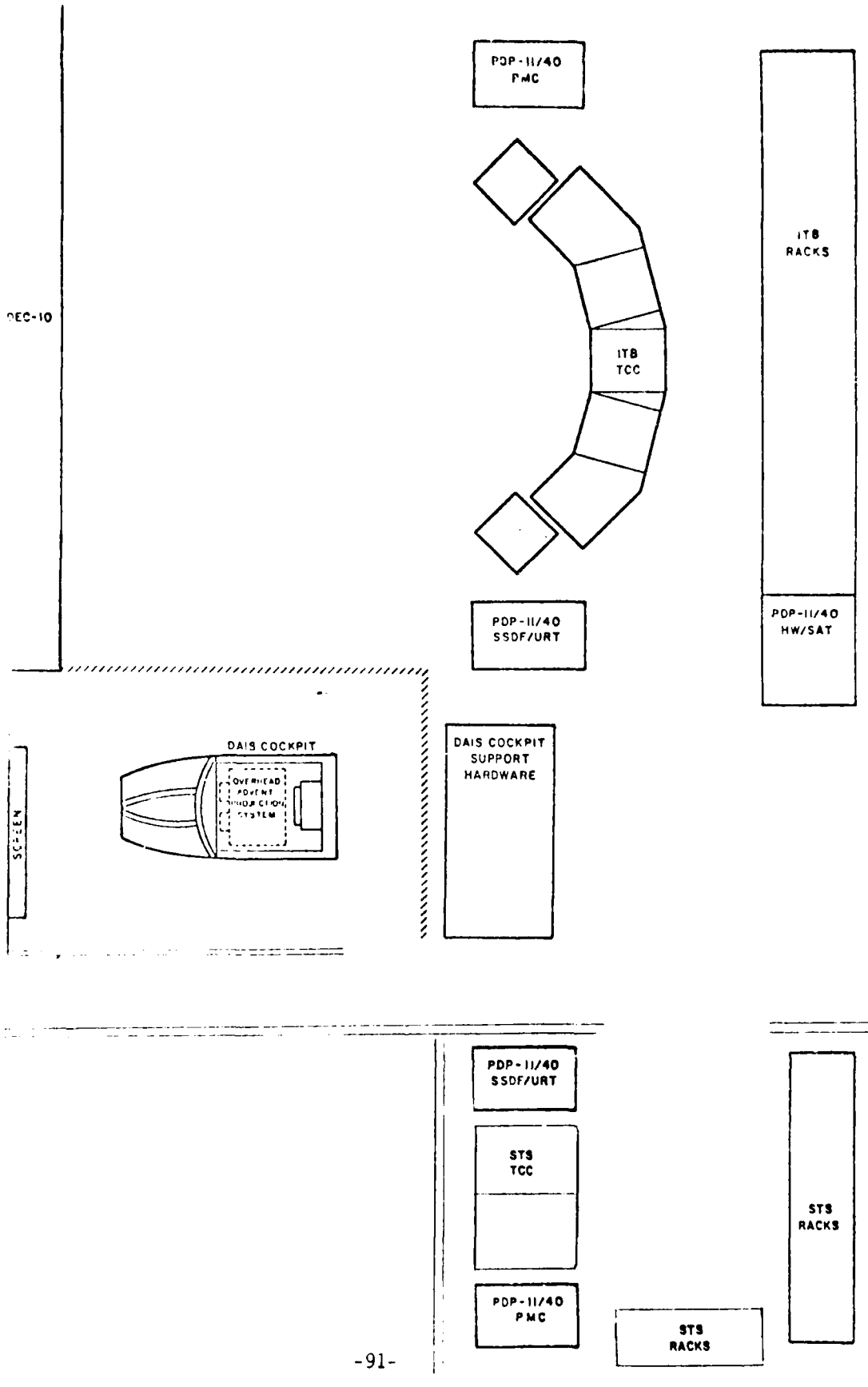


FIGURE 35. CURRENT STS/ITB FLOOR LAYOUT

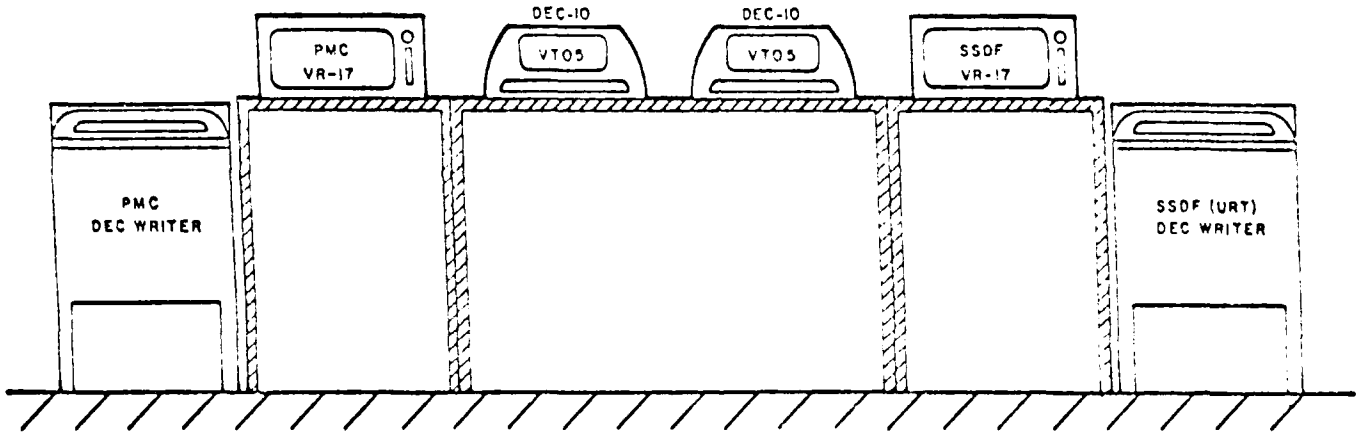


FIGURE 36. INITIAL SOFTWARE TEST STAND TEST CONTROL CENTER

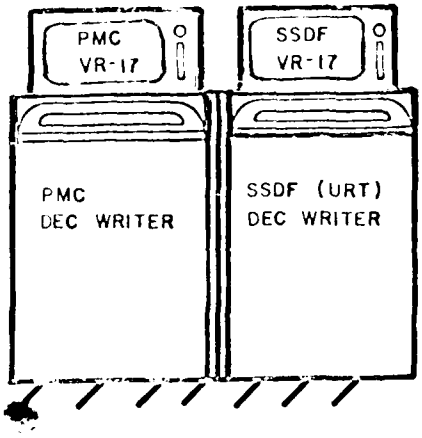


FIGURE 37. CURRENT SOFTWARE TEST STAND (STS) TEST CONTROL CENTER

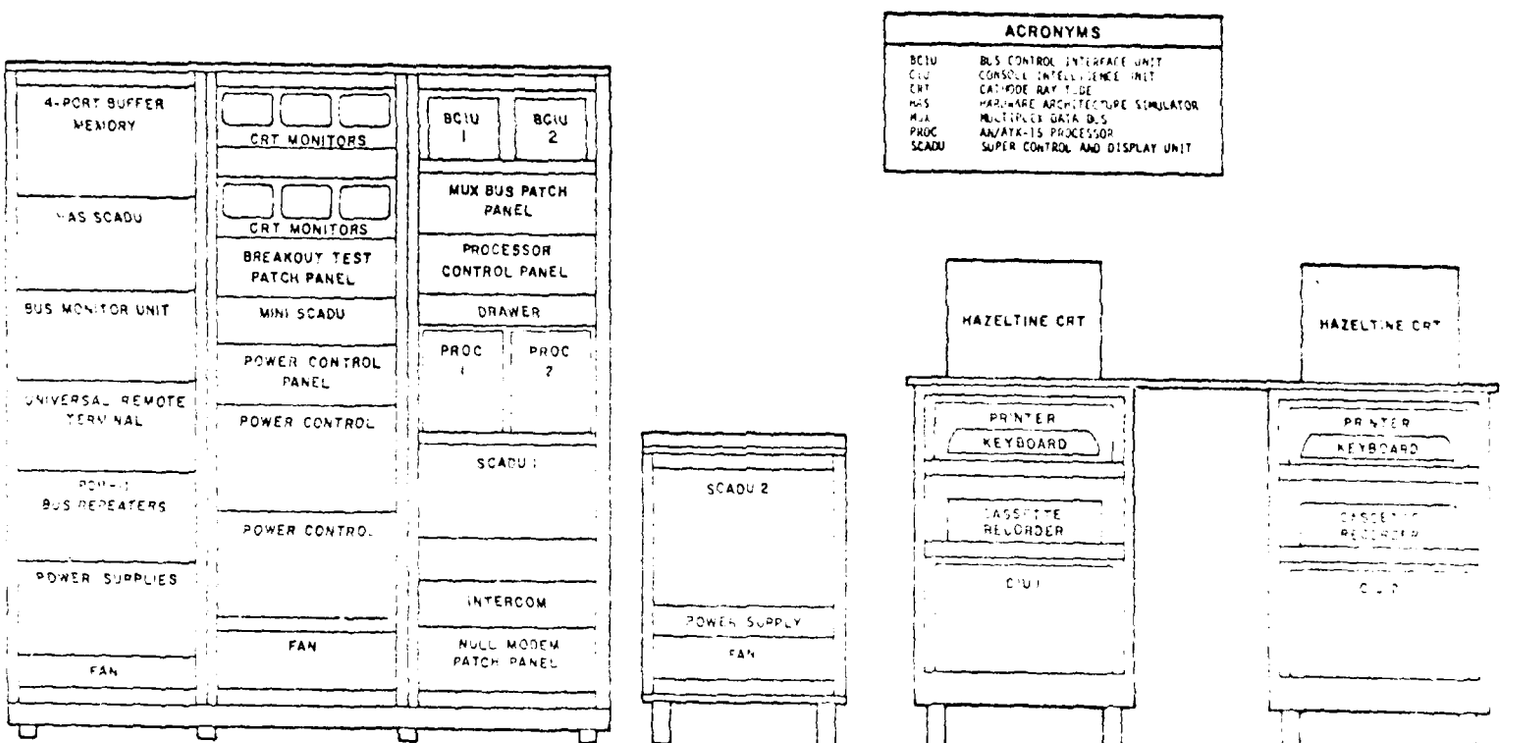


FIGURE 38. INITIAL SOFTWARE TEST STAND (STS) EQUIPMENT RACKS

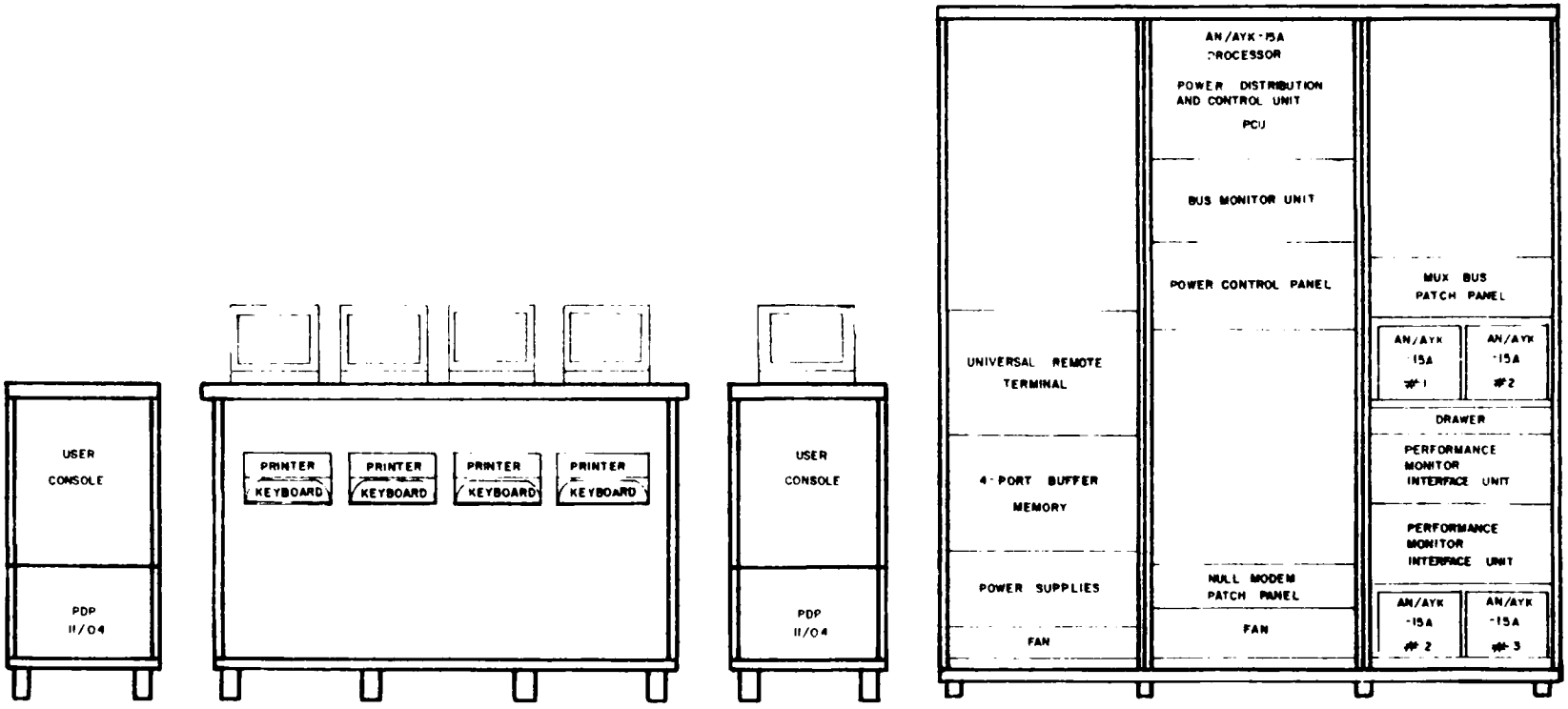


FIGURE 39. CURRENT SOFTWARE TEST STAND (STS) EQUIPMENT RACKS

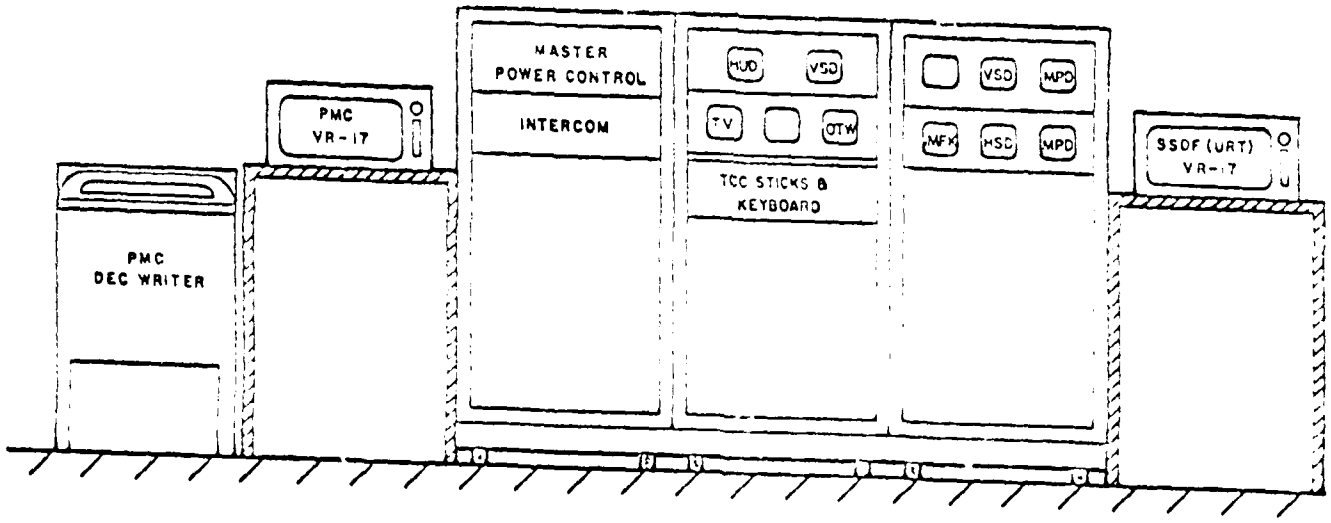


FIGURE 40. INTEGRATED TEST BED (ITB) TEST CONTROL CENTER

ACRONYMS	
BCIU	BUS CONTROL INTERFACE UNIT
BMU	BUS MONITOR UNIT
BR	BUS REPEATER
CIU	CONSOLE INTELLIGENCE UNIT
HAS	HARDWARE ARCHITECTURE SIMULATOR
MUX	MULTIPLEX DATA BUS
P/B/S	PROCESSOR/BCIU/S.ADU
R/F ASS'Y	RELAY/FILTER ASSEMBLY
SCADU	SUPER CONTROL & DISPLAY UNIT
URT	UNIVERSAL REMOTE TERMINAL

	CONSOLE INTELLIGENCE UNIT 4	CONSOLE INTELLIGENCE UNIT 2	FAN	FAN	MASTER POWER PNL	POWER PANEL 1	POWER PANEL 2
REMOTE TERMINAL R/F ASSEMBLY			BCIU's 1 & 2	BCIU's 3 & 4	EMERG. STOP	POWER PANEL 3	
HAZELTINE TERMINAL 4 CRT	HAZELTINE 2000 TERMINAL 3 CRT	HAZELTINE 2000 TERMINAL 2 CRT	MUX PATCH PANEL	MUX PATCH PANEL	PROCESSOR CONTROL PNL	SCOPE	LOGIC DRAWER
		HAZELTINE 2000 TERMINAL 1 CRT	STORAGE DWR		BUS MONITOR UNIT	UNIVERSAL REMOTE TERMINAL 1	POWER SUPPLY
KEYBOARD & PRINTER 4	KEYBOARD & PRINTER 3	KEYBOARD & PRINTER 2	PROCESSORS 1 & 2	PROCESSORS 3 & 4	SCADU 1	HAS SCADU	HAS 8 4-PORT R/F ASS'Y
		KEYBOARD & PRINTER 1	P/B/S R/F ASSEMBLY 1	P/B/S R/F ASSEMBLY 3	SCADU 2	BUS REPEATER	URT 1 & 2 R/F ASS'Y
	REMOTE TERMINAL 1 & 4	TAPE CASSETTE 1	P/B/S R/F ASSEMBLY 2	P/B/S R/F ASSEMBLY 4			BMU & BR R/F ASS'Y
	MUX PATCH PANEL	CONSOLE INTELLICENCE UNIT 3		MAIN POWER CONTACTORS	POWER SUPPLY BMU	4-PORT BUFFER MEMORY	POWER SUPPLY URT 1
			NULL MODEM PATCH PANEL		POWER SUPPLY SC3 SC4		POWER SUPPLY 4-PORT
	FAN	CIU 3 & 4 R/F ASS'Y			FAN	FAN	FAN

NOTES:
(1) 8 Main Racks are scaled to Panel Dimensions (19 or 24 x 70)
(2) Unlabeled areas are Blank Panels

FIGURE 41. INITIAL INTEGRATED TEST BED EQUIPMENT RACKS

ACRONYMS	
BCIU	BUS CONTROL INTERFACE UNIT
BMU	BUS MONITOR UNIT
BR	BUS REPEATER
CIU	CONSOLE INTELLIGENCE UNIT
HADS	HARDWARE ARCHITECTURE SIMULATOR
MUX	MULTIPLEX DATA BUS
P/B/S	PROCESSOR/SCIU/S.ADU
R/F ASS'Y	RELAY/FILTER ASSEMBLY
SCADU	SIMPLEX CONTROL & DISPLAY UNIT
URT	UNIVERSAL REMOTE TERMINAL

	CONSOLE INTELLIGENCE UNIT 4	CONSOLE INTELLIGENCE UNIT 2	FAN	FAN	MASTER POWER PNL	POWER PANEL 1	POWER PANEL 2
REMOTE TERMINAL R/F ASSEMBLY			BCIU's 1 & 2	BCIU's 3	EMERG. STOP	POWER PANEL 3	
					PROCESSOR CONTROL PNL		LOGIC DRAWER
			MUX PATCH PANEL	MUX PATCH PANEL			POWER SUPPLY
HAZELTINE 2000 TERMINAL 3 CRT	HAZELTINE 2000 TERMINAL 2 CRT	HAZELTINE 2000 TERMINAL 1 CRT		STORAGE DWR	BUS MONITOR UNIT	UNIVERSAL REMOTE TERMINAL 1	4-PORT R/F ASS'Y
			PROCESSORS 1 & 2	PROCESSORS 3 & 4	SCADU 1		
KEYBOARD & PRINTER 3	KEYBOARD & PRINTER 2	KEYBOARD & PRINTER 1					URT 1 & 2 R/F ASS'Y
REMOTE TERMINAL 1 & 4	TAPE CASSETTE 1	TAPE CASSETTE 2	P/B/S R/F ASSEMBLY 1	P/B/S R/F ASSEMBLY 3	SCADU 2		
			P/B/S R/F ASSEMBLY 2	P/B/S R/F ASSEMBLY 4			BMU & BR R/F ASS'Y
MUX PATCH PANEL	CONSOLE INTELLIGENCE UNIT 3	CONSOLE INTELLIGENCE UNIT 1		MAIN POWER CONTACTORS	POWER SUPPLY BMU	4-PORT BUFFER MEMORY	POWER SUPPLY URT
			NULL MODEM PATCH PANEL		POWER SUPPLY SC3 SC4		POWER SUPPLY 4-PORT
FAN	CIU 3 & 4 R/F ASS'Y	CIU 1 & 2 R/F ASS'Y			FAN	FAN	FAN

NOTES:
(1) 8 Main Racks are scaled to Panel Dimensions (19 or 24 x 70)
(2) Unlabeled areas are Blank Panels

FIGURE 42. CURRENT INTEGRATED TEST BED EQUIPMENT RACKS

1. Transfers the DAIS multiplex data to/from the PDP 11 for each message operation (controller-to-terminal, terminal-to-terminal, and terminal-to-controller) based upon:

   - RT address/subaddress
   - Transmit/receive bit
   - Word count
   - PDP 11 memory address

2. Responds to mode commands received on the DAIS multiplex data bus. The URT has the capability to generate an interrupt to the PDP-11 upon receiving the following mode commands:

   - MIL-STD-1553A
     - MTU1 or 2 shutdown
     - MTU1 or 2 shutdown override
     - Reset BIT word
     - Initialize terminal
     - Initiate serial channel I/O
     - Minor cycle sync

   - MIL-STD-1553B
     - Synchronize (with and without data word)
     - Transmit status
     - Transmitter shutdown
     - Override transmitter shutdown
     - Reset remote terminal
     - Transmit vector word
     - Transmit last command
     - Transmit BIT word

3. Provides the capability to set/reset bits in the serial channel activity register, module error register, and the BIT register for each simulated RT.

4. Provides the capability to inhibit the status response to any receive command.

5.2.2 Bus Monitor Unit

The Bus Monitor Unit (BMU) receives, records, and breakpoints on selected messages received on the DAIS multiplex data bus. The BMU stores the received messages into the PDP-11/40. The BMU interfaces with the DAIS multiplex data bus and monitors for data messages (controller-to-terminal, terminal-to-controller, and terminal-to-terminal messages) and mode commands. The BMU operates in either the automatic mode or manual mode.

The BMU is set up and controlled by the PMC or the PMIU software in the PDP-11/40. The BMU has the capability to provide selective monitoring as follows:

1. Record the bus traffic beginning at a specified breakpoint and record for a specific number of words.

2. Record the bus traffic beginning at a specified breakpoint and record for a specified length of time.

3. Record the bus traffic beginning at a specified breakpoint and record until a second specified breakpoint is reached.

4. Record the bus traffic beginning at a specified breakpoint and record only a specified message.

5. Record all bus traffic, only the control words (command and status), or only the data words.

6. Transmit a message.

The BMU also has a manual control function which can record all bus traffic, only the control words (command and status), only the data words, or only the message gap time. The manual modes are controlled directly from the BMU front panel.

5.2.3 Console Intelligence Unit

The Console Intelligence Unit (CIU) operates in conjunction with a RS-232 interface with the PMC PDP-11/40, DECsystem-10, and the Hazeltine terminals. It provides the means to control and load/change DAIS processor registers and memory. The CIU is supplied by the DAIS processor vendor to support the debug and test of the mission software under non-real-time conditions. The functions the user is able to perform are:

1. Clear Processor
2. Halt Processor
3. Toggle Memory Protect
4. Read Instruction Counter
5. Load Instruction Counter
6. Read Register
7. Load Register
8. Read Memory
9. Load Memory
10. Clear Breakpoints
11. Set a Breakpoint
12. Step Processor
13. Execute Processor
14. Load Tape Image
15. Write Tape Image
16. Stutter Mode
17. DEC to Cassette
18. Cassette to DEC

In the local mode, console commands are issued from the Hazeltine keyboard. When under control of the DECsystem-10 or PMC PDP-11/40, the CIU is in the remote mode. The DECsystem-10 or PMC PDP-11/40 functionally interfaces with the CIU similarly to the Hazeltine terminals. This allows the Hazeltine terminal functions to be performed at the DECsystem-10 for loading, or PDP 11/40 under PMC software control.

5.2.4 Hazeltine Terminals

The Hazeltine terminals, in conjunction with the Console Intelligence Unit (CIU) provide interactive control of the DAIS processor registers and memory as specified above. The Hazeltine terminals include the following:

1. Hazeltine CRT Display
2. Hazeltine Keyboard
3. Hazeltine Thermal Printer
4. Hazeltine Cassette Tape Unit

5.2.5 Processor User Console (AN/AYK-15A)

The processor User Console enables the user to communicate with the AN/AYK-15A processors. The User Console configuration is shown in Figure 43. The User Console was supplied by the DAIS processor (AN/AYK-15A) vendors to support the debug and test the mission software under non-real-time conditions. The functions the user is able to perform are:

1. Clear Processor
2. Display/Modify Memory
3. Display/Modify CPU General Registers
4. Display/Modify Instruction Counter (IC)
5. Display/Modify Fault (FT) Register
6. Display/Modify Status Word (SW) Register
7. Display/Modify Interrupt Mask (IM) Register
8. Display/Modify BCI Control Registers
9. Display/Modify Timer A
10. Display/Modify Timer B
11. Dump Memory Block
12. Step Processor
13. Step BCI
14. Set/Remove Breakpoints
15. Execute Processor
16. Load from Disk
17. Write to Disk
18. Control Printer
19. Clear User Console
20. Read User Console Status
21. Allow Processor Functions
22. Inhibit Processor Functions

In the local mode, console commands are issued from the Hazeltine keyboard. When under control of the DECsystem-10 or PDP-11/40 via the PMIU, the User Console is in the remote mode.

5.2.6 Performance Monitor Interface Unit (PMIU)

The Performance Monitor Interface Unit operates in conjunction with the PMIU software. It provides the STS users with the capability to monitor and control the software running in the DAIS processors (AN/AYK-15A). The PMIU software sets up the PMIU for real-time operation by loading the PMIU control registers via the DEC Unibus. The PMIU then performs one or more of the following CPU monitor functions:

1. Monitor all instruction addresses
2. Monitor a specific instruction (OP-code)
3. Monitor CPU or BCI instructions within areas of memory
4. Monitor memory stores within specific memory locations
5. Monitor operand stores where operand value meets conditions and address is specified

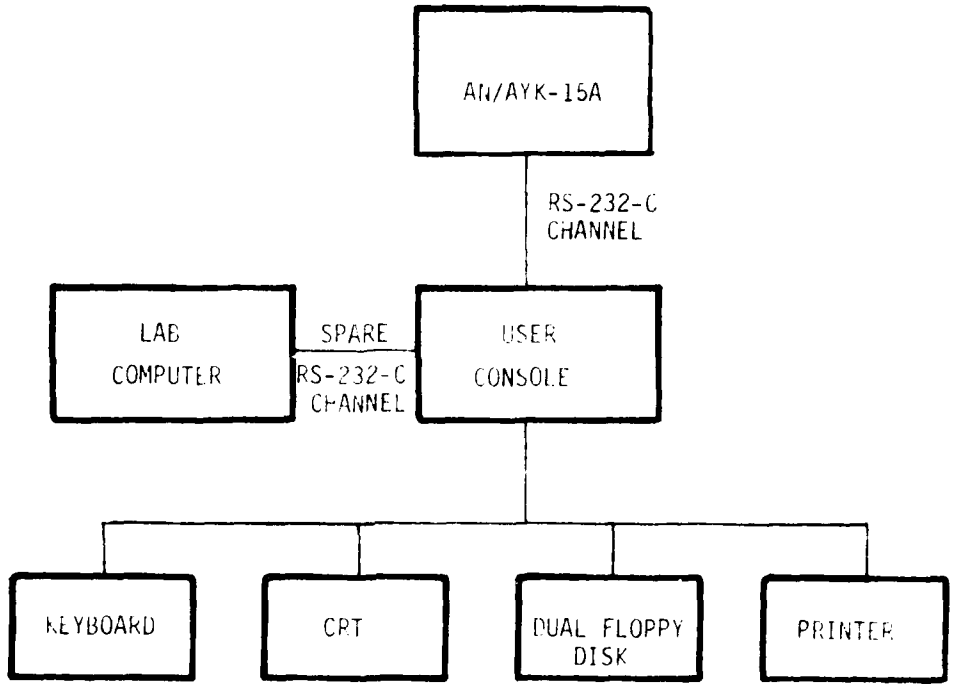


FIGURE 43. USER CONSOLE CONFIGURATION

6. Monitor memory reads within specific memory locations
7. Monitor all out of sequence branches
8. Monitor all interrupt occurrences
9. Monitor any DMA occurrence

The PMIU also performs one or more of the following control actions:

1. Halt AN/AYK-15A processors
2. Start execution AN/AYK-15A processors
3. Single Step AN/AYK-15A CPU or BCM instructions
4. Clear processor
5. Read AN/AYK-15A memory or registers
6. Write AN/AYK-15A memory or registers
7. Inhibit and enable PDP-11 interrupts

5.2.7 Simulated Subsystem Interface Unit (SSIU)

The Simulated Subsystem Interface Unit (SSIU) provides the interface, control, and data transfer functions required to interface a set of remote terminals (RTs) to a PDP-11 unibus. Each RT interfaces to the SSIU via a Facility Interface Module (FIM) which is specially designed for this purpose. Figure 44 is an interface block diagram illustrating the SSIU interconnection.

The SSIU provides the interface for the simulated sensor suite that in actual operation is interfaced by up to sixteen fully populated RTs with any complement of Interface Modules (IMs). A FIM occupies a single RT slot. It responds to any address which defines an IM slot and IM channel for which the data were actually intended. The SSIU uses this address as a pointer to store and retrieve data from unibus memory in preselected locations which are mapped to correspond to the RT configurations begin simulated.

The SSIU uses a PDP 11 unibus memory that can support a Direct Memory Access (DMA) data rate of at least 1 MHz for each 16-bit word and can operate in a burst mode (dedicated unibus) for up to 34 microseconds. Other features of the SSIU include:

1. At the end of each data block written into unibus memory, transferring a pointer by DMA to a preselected location defining the starting address of the data block

2. Providing an interrupt at the end of preselected messages

3. Providing an interrupt for several types of error conditions

Set up of the SSIU memory and control registers and monitor and control of the SSIU during operation are provided via unibus Programmed Input/Output (PIO).

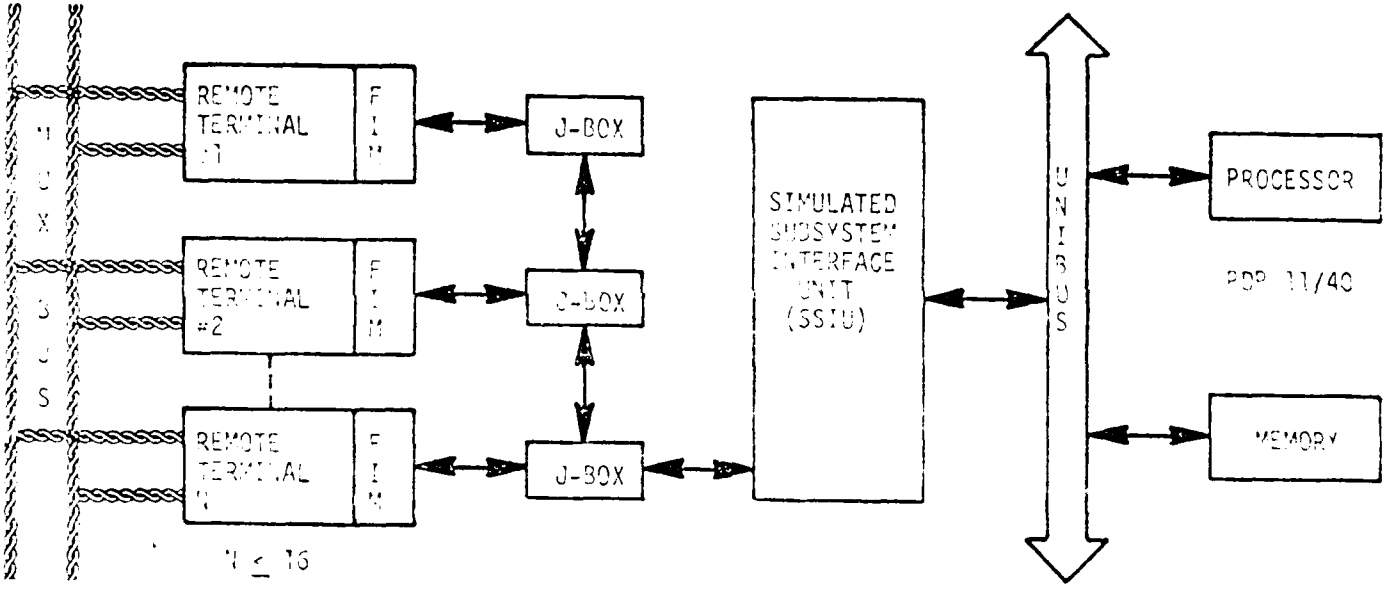


FIGURE 44. SSIU INTERFACE BLOCK DIAGRAM

5.2.8 Super Control and Display Unit (SCADU)

The Super Control and Display Unit (SCADU) operates in conjunction with the PMC software. It provides the ITB or STS users with the capability to monitor and control the software running in the DAIS processor (AN/AYK-15). It collects, stores and/or breakpoints on the data received from the DAIS processor (Table 8) and then halts the DAIS processors and/or BCIUs and interrupts the PMC PDP-11/40.

The DAIS processors make available to the SCADU information accessed or stored in the processor's memory. Based upon the processor's memory controller micro-code addresses, the SCADU determines which data (Table 8) is being received from the processor over the parallel multiplex data bus. The SCADU then selectively monitors and stores this data.

The PMC software sets up the SCADU for real-time operation by loading the SCADU control RAMs with specific micro-programmed monitor and control actions. The SCADU then performs one or more of the following monitor functions:

1. Monitor all instruction addresses
2. Monitor a specific instruction (Op-code).
3. Monitor all jump instructions which cause a branch.
4. Monitor all memory stores.
5. Monitor all memory accesses.
6. Monitor all interrupt occurrences.
7. Monitor all DMA occurrences.

The SCADU also performs one or more of the following control actions:

1. Breakpoint (halt) upon occurrence of one of the specific functions above.

2. Log the data in SCADU buffer (trace) with a time tag.

3. Compare the function with user's specific value (fixed point or floating point numbers) and break point.

4. Halt processors and BCIUs.

5. Interrupt PDP-11/40.

6. Start, stop, or reset the time tag timer.

After the appropriate control actions have occurred, the PMC software reads the data collected in the SCADU buffer, and reinitiates the same or a new operation.

TABLE 8. DATA AVAILABLE TO SCADU FROM AN/AYK-15 PROCESSOR

SCADU FUNCTION	SCADU DATA BUFFER CONTENTS		
	WORD A	WORD B	WORD C
IC (Instruction Address)	Instruction Address	Instruction	Operand Address
OPCODE (Instruction)	Instruction Address	Instruction	Operand Address
JUMPS	Operand Address		
STORES (1)	Operand Address	Most Significant Word (MSW) or Fixed Value	Least Significant Word (LSW) or Fixed Value
ACCESSES (1)	Operand Address	MSW or Fixed Value	LSW or Fixed Value
INTERRUPTS	Interrupt Address (20-3F)		
DMA's	Operand Address	Operand Value	

(1) NOTE: The contents of Word A, B, or C for the "STORES" and "ACCESSES" functions may vary depending on the instruction executed.

5.2.9 ITB Power Distribution and Control

The ITB Power Distribution and Control System (PDS) simulates aircraft electrical system power for the DAIS core hardware. The PDS simulates three power sources: battery, master AC generator, and Ram-Air Turbine (RAT). The output of the simulated power sources provides power to two sets (AC and DC) of three electrical bus systems: emergency, primary, and secondary. Each DAIS core hardware element is capable of simulated operation on any one of the electrical buses. Also, each individual hardware element has an on/off control. This permits simulated power control or power failure of any one of the power sources, simulated electrical buses, or individual equipments. The PDS includes provisions for software control of the power sources, electrical buses, or individual equipments. This software control is accomplished via a PDP-11/40.

The PDS system controls, distributes, and filters the power to each of the DAIS core hardware elements. The PDS system also controls and distribures power to the support hardware.

5.2.10 ITB Test Control Center

The ITB Test Control Center (TCC), shown in Figure 40, provides a centralized point for control of the entire ITB. The TCC contains the terminals which can control the PMC and SSDF PDP 11 processors and the DECsystem-10 simulation models via the PMC. Both system checkout and operation can be controlled from the TCC. The DAIS displays and the out-the-window scene are also displayed on CRT monitors at the TCC.

5.2.11 Controls and Backup Instruments System (CBIS)

The CBIS consists of the DAIS cockpit flight controls and displays that are not part of DAIS. The controls include the throttle, stick, rudder control and discretes. The displays are the basic flight instruments including the standby ADI, altimeter and warning lights. The CBIS interfaces to the ITB by means of the backup C/S interface unit which provides digital to analog conversions and a digital link to the SSDF (URT) PDP-11/40.

5.2.12 ITB Equipment Racks

The ITB equipment racks, as shown in Figure 42, include the ITB hardware as specified above as well as power supplies; special interface panels (multiplex data bus patch panel, Processor/BCM breakout test patch panel, and CIU/RS-232 patch panel); power control panels and processor control panel.

5.3 ITB Support Software and PDP-11 Processors

Support software is resident in the PDP-11 processors.

5.3.1 Performance Monitor and Control

The Performance Monitor and Control (PMC) software resides in a GT44 System (PDP-11/40) under RT 11. It is used to set up, control, and monitor the Mission Software resident in the DAIS processors (AN/AYK-15A). The PMC is capable of servicing from one to four DAIS processors at one time. It simultaneously monitors the multiplex bus traffic via the BMU and records selected DAIS system data for post-test analysis.

The PMC controls the SCADU's, CIU's, and BMU. The DAIS Processors are monitored and controlled through the SCADU's and the CIU's. The PMC starts and stops the set of DAIS Processors. The DAIS processor Mission Software is loaded from or dumped to the DECsystem-10 host simulation processor or the PDP 11 processors under PMC control.

The PMC sets up test cases which define the procedures and data collection for a simulation run. An operator is able to set up the test by means of an interactive interface and the creation of a test control file. The test control file is correlated with the DECsystem-10 simulation so that the collection of test data will proceed as defined by the file as the simulation progresses.

The PMC supports the system users in the debugging and evaluation of mission software. It allows selective real-time and non-real-time gathering of data from the DAIS processor and the multiplex data bus. The PMC provides a repertoire of commands to perform the non-real-time and real-time functions as follows:

1. Manipulate files on the DECsystem-10 and PDP-11 such as rename, delete, copy, etc.

2. Load or dump DAIS processor mission software from or to the PDP-11.

3. Perform the CIU functions as defined in paragraph 5.2.3 when the DAIS processors are halted.

4. Start and stop (halt) the DAIS processors and BCIUs.

5. Set up the SCADU to perform specified functions as defined in paragraph 5.2.2 during real-time operation.

6. Set up the bus monitor to record or breakpoint, as defined in paragraph 5.2.2, on specific multiplex data bus messages.

7. Display, print, or log the data collected from the bus monitor or DAIS processors via the CIU and SCADU. Data logged on a disk can be analyzed later by a Post Run Editor.

8. Provide the capability to examine or modify DAIS processor registers and memory locations either with absolute or symbolic addresses.

9. Provide interactive capability with the DECsystem-10 to set up and run the simulation models.

The PMC provides the capability to restart Mission Software, along with the simulation models, from a system snapshot point. This is accomplished by dumping Mission Software parameters at a specified PMC breakpoint, and reloading these parameters to restart the system. This snapshot and restart capability is used for repeated testing or demonstrations from a specific point in the mission profile.

5.3.2 Performance Monitor Interface Unit

The PMIU software provides the STS users with the capability to monitor and control the software running in the DAIS processors (AN/AYK-15A). The Performance Monitor Interface Unit (PMIU) is capable of controlling up to four PMIUs on one UNIBUS. The PMIU software is designed to run under the RT-11 operating system on the PDP-11. The PMIU software monitors and controls the DAIS processors (AN/AYK-15A) and monitors the multiplex bus traffic through the PMIU.

The PMIU software sets up test cases which define the procedures and data collection for a simulation run. An operator is able to set up the test by means of an interactive interface and the creation of a test control file. The test control file is correlated with the DECsystem-10 simulation so that the collection of test data will proceed as defined by the file as the simulation progresses.

The PMIU software supports the system users in the debugging and evaluation of mission software by allowing selective real-time and non-real-time gathering of data from the DAIS processor and the multiplex data bus. The PMIU provides a repertoire of commands to perform the non-real-time and real-time functions as follows:

1. Load or dump DAIS processor mission software from or to the PDP-11.

2. Set up the PMIU to perform specified functions as defined in paragraph 5.2.6 during real-time operation.

3. Display, print or log the data collected from the multiplex bus system or DAIS processors using the PMIU Post Run Editor.

5.3.3 Simulated Subsystem Data Formatter (SSDF/URT) Software

The main purpose of the SSDF/URT is to buffer data to and from the DAIS Models Simulation System (DMSS) and to provide timing signals to the DMSS. The SSDF/URT software sets up and controls the URT, double buffers the data in the four-port buffer memory, and provides a user-selectable display of the multiplex data bus messages. The software is designed to run on both the ITB and STS.

All URT data areas for transmission and reception of data are double buffered in the four port buffer memory (buffers A and B). When the models' output data is read, the SSDF/URT waits until the next minor cycle and switches the URT buffers. The DMSS output for CBIS is also read and processed if CBIS is on-line. The data received by the URT for the DMSS is passed to the DMSS as it is received. The CBIS/ICC inputs are passed to the DMSS 32 times per second.

The SSDF/URT and the DMSS use a handshaking flag to prevent the reading and writing of partially updated messages in the DMA window. Initially, the SSDF/URT sets the flag to a one. When the clock interrupt is generated to start a new DMSS cycle, the SSDF/URT completes writing any partial messages to the DMA window and then interrupts the DECsystem-1?. The DMSS sets the flag to a two, reads all inputs and then sets the flag to a three. While the DMSS is reading its inputs, the SSDF/URT reads the DMSS output and waits for the flag to be set to three. At this time, the SSDF/URT sets the flag back to one.

The minor cycle number (received with the synchronize mode command) is maintained by the SSDF/URT along with the major cycle number (number of times the minor cycle count transitions from 127 to zero). These two values are written into the DMA window as they change to allow tagging of any recorded data.

In non-real-time mode, the SSDF/URT performs the following:

1. Manages memory initialization.

2. Utilizes user-defined tables to initialize the URT RAMS and registers for the specific mission.

3. Initializes the four-port buffer memory, display processor, CBIS and keyboard.

4. Accepts user configuration commands to specify the following:

   a. Whether or not the SSDF/URT software should keep cycling the real-time models when mission software stops.

   b. Whether or not the URT should be set up to respond to minor cycle messages to bus address one (required for one-processor configuration to respond to a dummy minor cycle message).

5. Simulates a mass memory device using the PDP-11 disk unit to allow loading of STS and ITB processors over the DAIS multiplex data bus.

6. Initializes a real-time clock to provide interrupts 32 times per second to drive the real-time models.

The first minor cycle message received or a user command causes the SSDF/URT to enter real-time mode. In real-time mode, the SSDF/URT performs the following functions:

1. Maintains and displays minor and major cycle counts and passes them to the PMC software if required.

2. Generates interrupts to the DECsystem-10 to cycle the real-time models based on timer interrupts.

3. Moves data received from the URT and SSIU to the DMA window.

4. Controls the swapping of the URT double buffers.

5. Reads the models output data from the DMA to the four-port.

6. Uses the timer and DECsystem-10 interrupts to control access to the DMA window.

7. Inputs and outputs CBIS data (performing all necessary formatting).

8. Handles mode command interrupts from the URT as required to make the URT respond like a real RT.

9. Maintains displays of all pertinent information.

5.3.4 Evans and Sutherland Graphics System

The Evans and Sutherland graphics system is used to generate a cockpit out-the-window scene. The out-the-window scene is controlled by the simulation software so that the scene viewed from the cockpit has the correct dynamic orientation to synchronize with the simulated aircraft motion. The graphics interface software transforms aircraft attitude and position data obtained via the DECsystem-10 DMA window from the simulation software program into a form that can drive the Evans and Sutherland graphics system.

5.3.5 ITB PDP-11 Processors

The PDP-11 processors include:

1. Each PMC (PMIU) and SSDF (URT) processor has a complex consisting of one DEC GT-44 PDP-11/40 processor, and other peripherals.

2. The Evans and Sutherland Graphics interface processor consists of a PDP-11 processor.

5.4 STS Support Hardware

The initial STS hardware subsystems were identical to that described for the ITB:

1. One Universal Remote Terminal
2. One Bus Monitor Unit
3. Two Super Control and Display Units
4. Two Console Intelligence Units
5. Two Hazeltine Terminals
6. One Four-Port Buffer Memory
7. One HAS SCADU

The current STS hardware subsystems are identical to that described for the ITB or are described previously:

1. One Universal Remote Terminal
2. One Bus Monitor Unit
3. Two Performance Monitor Interface Units
4. Two User Consoles
5. One Four-Port Buffer Memory

The STS Power Distribution and Control (PDS) system controls, distributes and filters the power to each of these DAIS core elements and the STS support hardware.

The STS Test Control Center (the initial STS TCC is shown in Figure 36 and the current STS TCC is shown in Figure 37) provides a centralized point for control of the entire STS. The TCC contains the terminals which can control the PMC and SSDF PDP-11 processors and the DECsystem-10.

5.5 STS Support Software and PDP 11 Processor

The PMC software for STS is identical to the ITB PMC support software as described in paragraph 5.3.1.

The PMIU software for the STS is identical to the ITB PMIU support software as described in paragraph 5.3.2.

The SSDF (URT) software for STS is identical to the ITB SSDF (URT) software except interfaces for the SSIU and CBIS are not operated.

The PMC PDP GT-44 System and SSDF (URT) PDP GT-44 System are similar to the ITB PDP-11 processor.

5.6 DECsystem-10 Host Simulation Processor

The DECsystem-10 facility block diagram is shown in Figure 45. The KI DECsystem-10 CPU is used with the interfaces and peripherals shown in the block diagram. The DECsystem-10 facility is a conventional DEC configuration except for the Direct Memory Interface Controller which provides a DMA window access to the DECsystem-10 memory for data transfer between the DEC-system-10 and the PDP-11 processors.

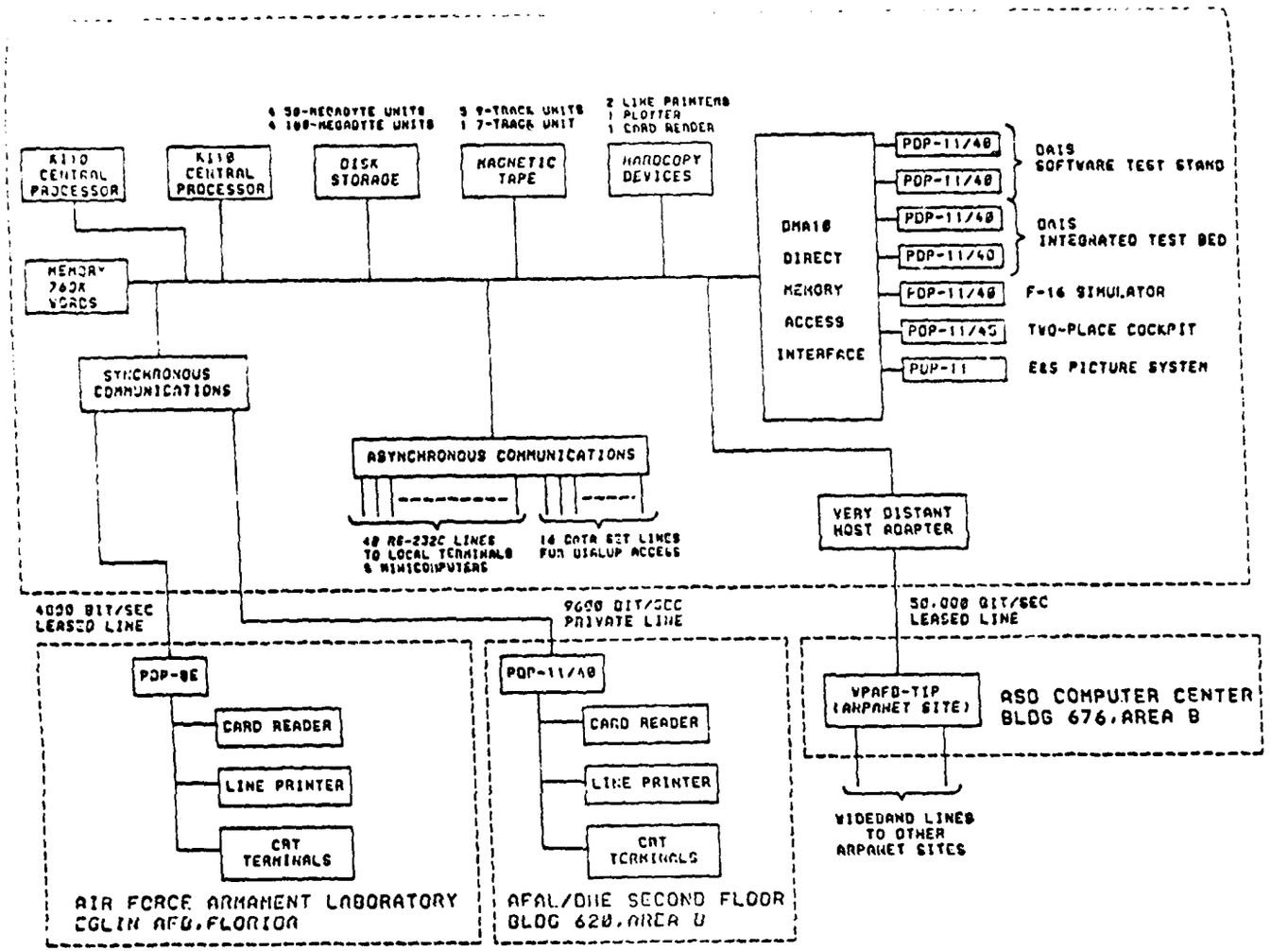


FIGURE 45. DECsystem-10 HOST SIMULATION PROCESSOR CONFIGURATION

5.7 Simulation Software

The Simulation Software, also known as the DAIS Models Simulation System (DMSS) is resident in the DECsystem-10 host simulation processor. The DMSS consists of seven functional types of software:

a. Executive software responsible for overall simulation control;

b. Sensor models, which simulate the on-board aircraft sensors;

c. SLU/STORES models, which simulate functional responses of weapon systems to the operational Stores Management software;

d. Aircraft/Environment models, which provide airframe dynamics, flight control dynamics, and propulsion data in response to pilot and external system actions;

e. Interface utilities, which provide access to the Crew Station cockpit controls and displays and to the Evans and Sutherland Picture System;

f. Checkpoint/Reset utilities, which allow for the capability to reset the scenario in both run mode and non-run mode system operation; and

g. Scenario Generator software, which can be used in place of the Aircraft/Environment software, and will simulate all inputs and outputs required to "fly" a predefined flight scenario.

The simulation software system as a whole operates with a major cycle (computational cycle) rate of 32 times per second. It should be noted, however, that not all models operate at this maximum rate. The models are broken into specific rate group subsets, based upon computational requirements (speed and accuracy) and operate at various speeds. Data is transferred 32 times per second, based on the highest rate group requirements. The data transfer information required by the simulation system from Mission Software is initiated when the SSDF/URT PDP-11 processor generates an interrupt to the DECsystem-10 at the start of each computational cycle. The simulation system then reads all inputs from the PDP-11 and then generates an interrupt back to the PDP-11. The simulation then performs its calculation and outputs its data to be provided to Mission Software to the DMA window. The SSDF/URT PDP-11 then accepts the data and transfers it to Mission Software.

5.8 Picture System

The picture system shown in Figure 46 consists of a simulation software-controlled ground display graphics generator and projector system. The graphics-generated display of the ground as seen by the pilot in the simulated cockpit, is projected on a motion picture screen in front of the cockpit.

An Evans and Sutherland graphics system generates an idealized pattern which represents ground terrain, targets, runway, etc. The Evans and Sutherland is controlled by the DECsystem-10 simulation software via the PDP 11 processor. The graphics are generated on a stroke CRT display and monitored by a closed circuit television camera. The TV image is transmitted to the Advent projection subsystem.

5.9 Data Reduction and Analysis Software

The Data Reduction and Analysis Software (DRAS) processes, formats, and outputs data from the Integrated Test Bed (ITB) or the Software Test Stand (STS). The DRAS is composed of two computer programs which perform the following functions:

- Read ITB/STS data from tape or disk
- Convert data from binary to decimal
- Rewrite selected data onto a disk
- Read the data from the disk file
- Generate report files
- Generate a disk file suitable for plotting

The report files generated by the DRAS include a data report, an event and time report, and an error message report. The data report contains a time history of selected parameters recorded during ITB/STS system operation. The event and time report contains a block of STS/ITB data recorded at the time a discrete event changes. These events include changes in navigation, steering, or master modes, weapon release, thrust, INS alignment status, and weight-off-gear. The error message report contains a list of all errors which may have occurred during processing.

DRAS is used to support system testing and DMSS testing. It is also used to provide a permanent record documenting the successful completion of formal demonstrations.

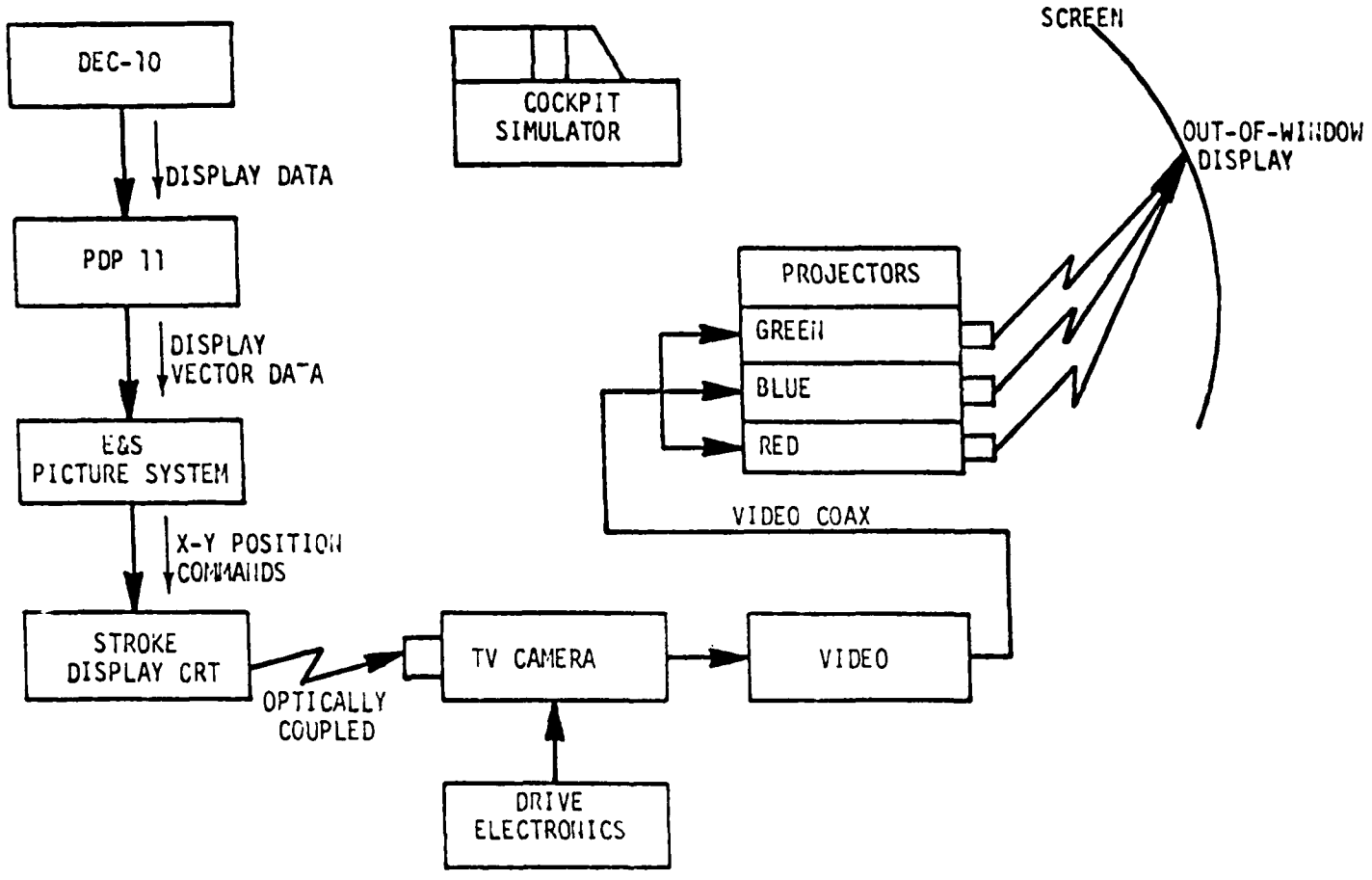


FIGURE 46. FUNCTIONAL CONFIGURATION - PICTURE SYSTEM

6.0 SOFTWARE CONVERSION

6.1 Mission Software Conversion

The Mission Software was converted from JOVIAL J73/I to JOVIAL J73. The conversion and retest effort took almost 15 months because corrections and changes were being made to the new JOVIAL J73 compiler simultaneously. The MSW conversion was accomplished in an orderly and organized manner.

6.1.1 Executive Conversions

The executive software underwent three stages of conversions. First, the executive was converted from MIL-STD-1553A to MIL-STD-1553B bus protocol. The second stage of conversion was the retargeting of the executive from the AN/AYK-15 processor to the MIL-STD-1750 (AN/AYK-15A). The last stage of conversion was the translation of the source code from J73/I to MIL-STD-1589B (JOVIAL J73). At the completion of each of the three conversions, the executive was fully tested.

The conversion to MIL-STD-1553B bus protocol resulted in simplification of the bus control functions of the executive. A small amount of redesign of the executive was required to incorporate the new bus protocol efficiently.

The conversion to AN/AYK-15A required modifications primarily to the assembly language routines of the executive. These routines contain the machine-dependent interface such as interrupt handlers and Bus Control Interfaces. The new processor also had a different repertoire of machine instructions which also affected these routines. Also, the use of the JOVIAL AN/AYK-15A code generator necessitated the inclusion of a stack management system into the executive.

The language conversion to JOVIAL involved primarily syntax changes. However, several semantic changes were required because of strong type checking and addition/deletions of data types.

6.1.2 Application Software Conversion

The application software was converted to JOVIAL J73 and retested. First, the Compool Blocks and the Comsubs were converted because these are used by the other tasks. Then, the application tasks were converted. The acceptance test programs (ATP's) were converted along with each task so that testing could be kept up-to-date.

Some of the changes required to convert the tasks were trivial; square brackets were changed to parentheses and the syntax of certain JOVIAL statements was modified. Many of the data items had to be converted so that the data types matched. The structure of the data tables in the common data area had to be redefined.

6.2 Core Element Test Software Conversion

The Controls and Displays (C&D) Acceptance Test Program (ATP) was converted from JOVIAL J73/I to JOVIAL J73. The ATP handles the interface between the C&D devices and the test software; it also controls testing by

prompting the user, and it performs the actual acceptance test. The only problems encountered during the conversion involved redefining specified data tables; other required changes were simple to make.

The Remote Terminal (RT)/Interface Module (IM) Diagnostics exercises RT and IM hardware through a series of tests and allows the isolation of errors. The RT/IM Diagnostics has two versions:

1. The original is loaded in an AN/AYK-15 processor and uses the MIL-STD-1553A multiplex bus system for testing.

2. The second is loaded in an AN/AYK-15 processor, as the original, and uses the MIL-STD-1553B multiplex bus system for testing. This version was created by converting the original diagnostics to meet MIL-STD-1553B.

The RT/IM Diagnostics tests are described in the hardware integration and test section.

6.3 Multiplex Diagnostics Software Conversion

The Master Multiplex System Diagnostics are a set of up to 32 tests that test the multiplex bus system and its devices. These devices include:

1. Master processor and BCIU or Master processor/BCM
2. Remote processor and BCIU or Remote processor/BCM
3. Remote Terminal
4. Universal Remote Terminal
5. System Mass Memory

There are three versions of the Multiplex Diagnostics:

1. The original is loaded in an AN/AYK-15 processor and uses the MIL-STD-1553A multiplex bus system for testing.

2. The second version is loaded in an AN/AYK-15 processor, as the original, and uses the MIL-STD-1553B multiplex bus system. This version was created by converting the original diagnostics to meet MIL-STD-1553B and adding new tests.

3. The last version is loaded in an AN/AYK-15A processor and uses the MIL-STD-1553B multiplex bus system. The version was created by converting the second diagnostics to AN/AYK-15A instruction set.

Table 9 contains a list of the tests for all the versions. A list of devices used in the test by each version is found in Tables 10 through 12.

The Command Generation Software (CMDGEN) is a general purpose program designed to give a user the following capabilities:

TABLE 9. LIST OF ALL THE MUX DIAGNOSTICS TESTS

TEST #	AN/AYK-15 MIL-STD-1553A	AN/AYK-15 MIL-STD-1553B	AN/AYK-15A MIL-STD-1553B
Ø	Initial Polling	Initial Polling	Initial Polling
1	BCIU Initialization	BCIU Initialization	Spare
2	BCIU Undefined Mode Codes	BCIU Undefined Mode Codes	BCI Undefined Mode Codes
3	BCIU MTU Shutdown Mode Codes	BCIU MTU Shutdown Mode Codes	BCI MTU Shutdown Mode Codes
4	BCIU Mode Code with Interrupts	BCIU Mode Code with Interrupts	BCI Mode Code with Interrupts
5	RT Undefined Mode Codes	RT Undefined Mode Codes	RT Undefined Mode Code
6	RT Initialize Terminal Mode Codes	RT Initialize Terminal Mode Codes	RT Initialize Terminal Mode Codes
7	RT MTU Shutdown Mode Codes	RT MTU Shutdown Mode Codes	RT MTU Shutdown Mode Codes
8	URT Mode Codes	URT Mode Codes	URT Mode Codes
9	BCIU Asynchronous Data Test A	BCIU Asynchronous Data Transfer Test A	BCI Asynchronous Data Transfer Test A
10	BCIU Asynchronous Data Test B	BCIU Asynchronous Data Transfer Test B	BCI Asynchronous Data Transfer Test B
11	Spare	Spare	BCI State Transition
12	Spare	BCM Internal Operations Test	BCI Internal Operation
13	Spare	Remote Transmission Lockout Flag	Remote Transmission Lockout Flag

TABLE 3. LIST OF ALL MUX DIAGNOSTICS TESTS (CON'T)

TEST #	AN/AYK-15 MIL-STD-1553A	AN/AYK-15 MIL-STD-1553B	AN/AYK-15A MIL-STD-1553B
14	Spare	Remote Reception Lockout Flag	Remote Reception Lockout Flag
15	Spare	Spare	Lockout Flag Set/Reset
16	Synchronous Data Test A	Synchronous Data Transfer Test A	Synchronous Data Transfers A
17	Synchronous Data Test B	Synchronous Data Transfer Test B	Synchronous Data Transfers B
18	Synchronous Data Test C	Synchronous Data Transfer Test C	Synchronous Data Transfers C
19	Synchronous Data Test D	Synchronous Data Transfer Test D	Synchronous Data Transfers D
20	Synchronous Data Test E	Synchronous Data Transfer Test E	Synchronous Data Transfers E
21	Mass Memory Data Transfer	Mass Memory Data Transfer	Mass Memory Data Transfer
22	Mass Memory Data Transfer	Mass Memory Data Transfer	Mass Memory Data Transfer
23	Mass Memory Integrity	Mass Memory Integrity	Mass Memory Integrity
24	Mass Memory Integrity	Mass Memory Integrity	Mass Memory Integrity
25	Spare	Spare	Remote BCI Self Test
26	Spare	Spare	Master BCI Self Test
27	PI/O Wrap	PI/O Wrap	Lockout Flag Duration

TABLE 9. LIST OF ALL MUX DIAGNOSTICS TESTS (CON'T)

TEST #	AN/AYK-15 MIL-STD-1553A	AN/AYK-15 MIL-STD-1553B	AN/AYK-15A MIL-STD-1553B
28	Interrupt Wrap Test	Interrupt Wrap	Lockout Flag during Receive
29	DMA Wrap Test	DMA Wrap	Lockout Flag during Transmit
30	BIM Wrap Test	BIM Wrap	MTU Shutdown
31	Spare	Spare	Bus Active Bit
32	Bus List	Bus List	Bus List

TABLE 10. DEVICES USED BY EACH TEST IN MUX DIAGNOSTICS AN/AYK-15, 1553A

TEST NUMBER	DEVICES DESIGNED FOR: MASTER BCIU	REMOTE BCIU	RT-FIM	RT-IMS	URT	MM
0	X	X	X	X	X	X
1	X					
2	X	X				
3	X	X				
4	X	X				
5	X		X	X		
6	X		X	X		
7	X		X	X		
8	X				X	
9	X	X				
10	X	X				
11	**********	**********	NOT DEFINED	**********	**********	
12	**********	**********	NOT DEFINED	**********	**********	
13	**********	**********	NOT DEFINED	**********	**********	
14	**********	**********	NOT DEFINED	**********	**********	
15	**********	**********	NOT DEFINED	**********	**********	
16	X	X	X		X	
17	X	X	X		X	
18	X	X	X		X	
19	X	X	X		X	
20	X	X	X		X	
21	X					X
22	X	X				X
23	X					X
24	X					X
25	**********	**********	NOT DEFINED	**********	**********	
26	**********	**********	NOT DEFINED	**********	**********	
27	X					
28	X					
29	X					
30	X					
31	**********	**********	NOT DEFINED	**********	**********	
32	X	X	X	X	X	

TABLE 11. DEVICES USED BY EACH TEST IN MUX DIAGNOSTICS AN/AYK-15, 1553B

TEST NUMBER	DEVICES DESIGNED FOR: MASTER BCIU	REMOTE BCIU	RT-IMS	URT	MM
0	X	X	X	X	X
1	X				
2	X	X			
3	X	X			
4	X	X			
5	X		X		
6	X		X		
7	X		X		
8	X			X	
9	X	X			
10	X	X			
11	**********	NOT DEFINED	**********		
12	X				
13	X	X			
14	X	X			
15	**********	NOT DEFINED	**********		
16	X	X		X	
17	X	X		X	
18	X	X		X	
19	X	X		X	
20	X	X		X	
21	X				X
22	X				X
23	X				X
24	X				X
25	**********	NOT DEFINED	**********		
26	**********	NOT DEFINED	**********		
27	X				
28	X				
29	X				
30	X				
32	X	X	X	X	

TABLE 12. DEVICES USED BY EACH TEST IN MUX DIAGNOSTICS AN/AYK-15A, 1553B

	DEVICES DESIGNED FOR:				
TEST	MASTER BCIU	REMOTE BCIU	RT	URT	MM
0	X	X	X	X	X
1	********** NOT DEFINED **********				
2	X	X			
3	X	X			
4	X	X			
5	X		X		
6	X		X		
7	X		X		
8	X			X	
9	X	X			
10	X	X			
11					
12					
13					
14					
15					
16	X	X		X	

TABLE 12. DEVICES USED BY EACH TEST IN MUX DIAGNOSTICS AN/AYK-15A, 1553B (Con't)

	DEVICES DESIGNED FOR:				
TEST	MASTER BCIU	REMOTE BCIU	RT	URT	MM
17	X	X		X	
18	X	X		X	
19	X	X		X	
20	X	X		X	
21	X				X
22	X	X			X
23	X				X
24	X				X
25	X	X			
26	X				
27	X	X		X	
28	X	X		X	
29	X	X		X	
30	X	X		X	
31	X	X		X	
32	X	X	X	X	

1. Build and execute up to 100 bus instructions. These instructions cause commands and/or data to be sent on the multiplex bus by the BCM.

2. Specify the data to be sent on the multiplex bus by the BCM.

3. Load BCM registers.

4. Specify the bus address used by the BCM controlled by CMDGEN.

5. Cause the BCM to run in either master or remote mode.

There are three versions of CMDGEN:

1. AN/AYK-15, MIL-STD-1553A
2. AN/AYK-15, MIL-STD-1553B
3. AN/AYK-15A, MIL-STD-1553B

The development of these versions was similar to the development of the Master Multiplex Diagnostics.

6.4 1750A ATP

The AN/AYK-15A Acceptance Test Program (ATP) was converted to test the MIL-STD-1750A. The ATP consists of extensive tests for each instruction in the instruction set. The first step in converting the ATP was to define the differences in the two instruction sets (MIL-STD-1750 and MIL-STD-1750A). Any differences in the instruction sets resulted in differences in the two ATP's.

Differences include:

- Several instruction set changes were made (RFR and CFR were combined to make RCFR, for example).
- New interrupts were defined.
- New options were defined (such as expanded memory).
- Stack was made to increment in opposite direction.

The general instruction modules required minor changes because of instruction set differences. The executive module needed to be modified to include new interrupt handlers, stack differences, and some expanded memory processing.

Several new tests were written to test the expanded memory option and the various options associated with expanded memory, such as access lock/key, E/W, bit, and page register operations. These modules were written as stand-alone tests. New tests were also written to test new instructions such as BEX (branch to executive) and VIO (vectored I/O). These were designed to be run with the executive module.

7.0 HARDWARE CONVERSION

The hardware conversion of the DAIS ITB consisted of upgrading the BCIU's and RT's to be 1553B-compatible and subsequently replacing the BCIU's and associated AN/AYK-15 processors with the integrated AN/AYK-15A processor/BCM. The integration and test of the AN/AYK-15A is described elsewhere in this report. This section will concentrate on the multiplex conversion.

7.1 Multiplex System Conversion

Shortly after MIL-STD-1553B was formally approved and published, the DAIS Problem Report Board (PRB) appointed a working group composed of AFWAL and contractor personnel to perform an impact assessment of converting DAIS to the new standard. The working group identified issues, developed technical approaches to resolving them, and drafted costs and schedule plans for accomplishing the conversion. Based on this plan DAIS management elected to perform an in-house conversion of the BCIU's and let a sole source contract for the upgrade of the RTs.

The impact assessment report identified twelve technical tasks as follows:

- Delete Message Error, Busy, and Terminal Fail Bit suppression for Transmit BIT and Transmit Last Command.
- Update Last Command Register for Transmit Status and Transmit BIT (previously not updated).
- Design/implement new system control procedure for RT Asyrchronous protocol which does not require the reset mode command.
- Design/implement new system control procedure for interprocessor asynchronous protocol which does not use status embedded ARV or require reset status code.
- Define/implement a Reset Built-In-Test (BIT) without a Reset BIT Mode Command.
- Define/implement a new bus shutdown procedure using 1553B Mode Commands.
- Implement RT/Subsystem Retry at the RT/IM Level.
- Change/delete mode command assignments for DAIS/1553B equivalent command.
- Do not suppress the busy state in the status response for transmit status.
- Assign a new asynchronous subaddress.

- Include subaddress 31 as an additional mode command indicator in both Master and Remote BCIU.

- Re-assign all status bit discretes.

These tasks reflect the fact that whereas 1553A had left mode code definitions optional, 1553B spelled them out in detail and did not include all the mode codes DAIS had defined. The more significant differences were in the areas of asynchronous operations, RT/IM error handling and the use of subaddress 31 for mode codes.

The asynchronous differences resulted from the fact that DAIS had embedded an Asynchronous Request Vector (ARV) in the status word and used a mode code to clear the ARV after it had been read. MIL-STD-1553B allowed only the service request bit in the status word, moved the ARV to the vector word and provided a mode code to read the vector but none to clear it.

For the RT/IM error handling DAIS had defined a status word bit and a mode code, Interrogate Module Error Register, to identify the IM and channel in error. MIL-STD-1553B included neither the status word bit nor the mode code.

Subaddress 31 had been used in DAIS to interrupt a remote processor (the only subaddress on which a message would cause an interrupt) as part of the asynchronous protocol. The use of this for a mode code required DAIS to use another subaddress for asynchronous messages in addition to the modifications necessary to recognize 31 as signaling a mode code.

It is interesting that recognizing subaddress 31 for mode codes is the only modification of the BCIU actually requiring a hardware change. All other modifications were accomplished in BCIU microcode. The RT modifications included an automatic retry of the IM/channel in error and this was part of the motivation for not attempting the RT upgrade in house.

The conversion of DAIS to 1553B was accomplished in a three-phase effort of system engineering, documentation, and development. More than half the total effort was devoted to documentation, resulting in a complete set of core element specifications fully 1553B-compatible. The final testing was accomplished in two steps, the first using a PROM simulator (RAM) to debug the microcode changes. The second step involved burning the real PROMS. This testing was completely successful and accomplished ahead of schedule. An additional two units were converted and provided to the developers of the AN/AYK-15A to aid in their testing.

8.0 INTEGRATION AND TEST

The modular structure and well-defined interfaces of both the hardware and software elements in the DAIS System Architecture greatly facilitate the integration and test. A step-by-step approach was taken to test and integrate both the DAIS core elements and the support hardware and software. Initially, stand-along tests were performed on individual elements. Then, these elements were incrementally integrated and tested until the system was completed. The following sections highlight several of the key techniques which contributed to the successful and rapid integration and test activities.

8.1 Stand-alone Tests

8.1.1 Core Element Hardware

As each core element subsystem was received from the contractor, stand-along acceptance tests were performed. First, the contractor supplied test support equipment and software. These tests were then augmented with FWAL and SITC tests. The tests were developed to verify all the electrical and functional interfaces of each subsystem. In several cases, the contractor-supplied tests were not very comprehensive, and the test support equipment did not allow complete dynamic testing of all the functional interfaces. As a result, specifically for the multiplex equipment, complete acceptance testing of these core elements was not completed until the subsystem was integrated and tested with other system elements as discussed in the hardware integration and test section below.

The AN/AYK-15A processor has an Acceptance Test Program that runs the AN/AYK-15A in a stand-alone state. The ATP is an integral part of the DAIS processor qualification testing package. The ATP consists of four functional types: (1) Executive software, responsible for control and execution of specific processor tests; (2) Processor test modules, responsible for the testing of AN/AYK-15A processor functions, e.g. instruction tests; (3) Error processing, responsible for the processing of errors detected by processor test modules; and (4) Interrupt handler, responsible for the processing of interrupts.

The ATP was designed so that the minimum number of 64K loads are required. An attempt was made to group similar test modules within a particular load, for example memory tests and memory write protect test. Those tests that do not lend themselves to executive control were placed in separate loads. Table 13 contains a list of special test names and descriptions. The special tests and CPU instruction tests are located in the LOAD modules. A list and description of the ATP load modules is contained in Table 14.

8.1.2 Executive Software Testing

The DAIS Startup Loader was also tested. A set of application tasks were developed which send messages to the C&D for display. The displays (on the IMFK) were sufficient to inform the test operator of the state of the software in each of the processors after inputs to the Processor Control Panel. Several problems were uncovered in the Startup Loader, Executive's interface to the Startup Loader, processor, and the BCM. After these problems were corrected, the Startup Loader performed as required.

TABLE 13. SPECIAL ATP TEST NAMES AND DESCRIPTIONS

NAME	DESCRIPTION
HANGT	Randomly generates and executes a CPU instruction list.
GENREG	Verifies capability to read/write the sixteen general registers.
STATREG	Verifies capability to read/write the status word.
ILLIO	Verifies proper interrupt and fault register setting upon execution of illegal I/O instructions.
ILLINT	Verifies proper interrupt and fault register setting upon execution of illegal CPU instructions.
WRPRAM	Verifies capability to read/write the memory protect RAM.
BNCH	Computes processor throughput.
TRGTST	Outputs trigger GO discrete.
DISCRT	Verifies proper operation of external discrete lines.
DISABL	Verifies that no interrupts are handled immediately following a disable instruction.
ROMTST	Verifies integrity of the bootstrap ROM.
PIOTST	Verifies proper operation of PIO lines.
TIMER	Verifies interval timer values with and without interrupts.
INTERT	Verifies capabilities of handling, masking, and clearing externally generated interrupts.
INMASK	Verifies capability to read/write the interrupt mask.
BCMGEN	Verifies capability to read/write the sixteen BCM general registers.
BCMCN1	Verifies capability to read/write the sixteen BCM control registers.
BCMCN2	Verifies capability to multiple store, compare and perform bit manipulation on the sixteen BCM control registers.

TABLE 14. ATP LOAD MODULE NAMES AND DESCRIPTIONS

NAME	DESCRIPTION
BOUND.OBJ	Boundary test
HANG.OBJ	Hang test
ICTEST.OBJ	Instruction counter test
INDEX.OBJ	Index test
MEMADR.OBJ	Illegal memory address accessing test
MEMTIM.OBJ	Memory access timing test
MEMTST.OBJ	Memory access and protection test
PWRDWN.OBJ	Power up/down test
LOAD1.ABS	Executive-controlled tests 000-063
LOAD2.ABS	Executive-controlled tests 064-097
LOAD3.ABS	Executive-controlled tests 098-0D0
LOAD4.ABS	Executive-controlled tests 0D1-111

8.1.3 Application Software Unit Test

Unit testing of the Application Software was performed on the host computer, a DECsystem-10. At this level of testing, the logical operation of each single task was tested. Acceptance Test Programs (ATP's) and other support programs were written to implement the testing.

The support programs required for the testing include the following. 1) a version of the DAIS Executive that runs on the DECsystem-10 computer, and 2) a data area similar in function to the data base created by PALEFAC.

Each ATP generates data to be used as input to the program being tested. Using the executive services it places the input data in the data area. The ATP then invokes the MSW program being tested. The MSW program accesses its input, executes, and places its output in the common data area. Finally, the ATP accesses and then prints the results of the test. The testing process was automated so that retesting can be easily accomplished.

8.1.4 Environmental Model Tests

Acceptance testing was performed on the DAIS Models Simulation System (DMSS). The acceptance test was performed at three levels:

1. Stand-alone test - a verification of responses by the DMSS to simulated inputs.

2. Interface and Functional test - a verification of internal DMSS communications and outputs to the DMA-10 window.

3. System test - a validation of the entire DAIS system.

Stand-alone testing was performed for each individual model. The test load module consisted of a tailored test driver and the model being tested. These tests were run on the DECsystem-10 computer.

The interface and functional test was performed by inserting the sensor model being tested into the baseline model set for all other models. Then, using the mission scenario generator (MSG), the response of the new model was recorded and then compared with the response of the baseline model set. This process was repeated for each model until all sensor models were individually integrated and tested.

The acceptance tests revealed several problems that required resolution.

8.2 Hardware Integration and Test

Incremental integration and testing was performed, as each subsystem became available, until the system was fully integrated and tested. The key to the successful integration and test was the development of extensive test software. Four major test software programs were developed: Multiplex System Diagnostics Software, Remote Terminal (RT)/Interface Module (IM) Diagnostics Software, Controls and Displays Acceptance Test Program (ATP), and Performance Monitor Interface Unit (PMIU) ATP.

The Multiplex System Diagnostics were previously described in the Multiplex Test Software Conversion section. Sample Hardware Configurations for the AN/AYK-15A Multiplex Diagnostics are shown in Figure 47.

The Remote Terminal (RT)/Interface Module (IM) diagnostics exercise RT and IM hardware through a series of tests and allow the isolation of errors. Each test is designed for an RT or a specific type of IM. The diagnostics run in an AN/AYK-15 processor as shown in Figure 48. The device(s) to be tested is attached to the MIL-STD-1553B bus. The user specifies the system configuration and the tests to be run. The diagnostic's executive then sequentially calls each test selected. Each test that is integrated with this software is required to do the following:

1. Perform its own error checking and reporting and maintain error counters.

2. Perform all BCIU control including interrupt handling.

3. Return the following items in the same state as when the test was called:

   a. BCIU registers
   b. All interrupt vectors
   c. The BCIU subaddress pointers

The handling of all errors by a test are controlled by user-specified parameters. The error reporting for each test must interface with these parameters to allow the following options:

1. Stop on errors and allow the user to retry the failing sequence or go on to the next sequence.

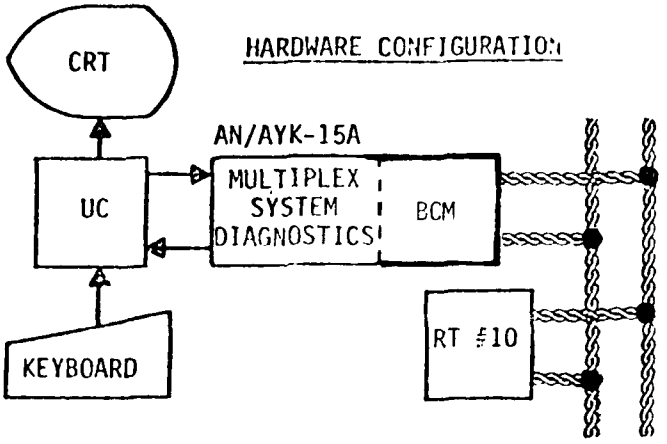
2. Do not display errors but maintain counts of the different types of errors.

A list of RT/IM diagnostic tests in found in Table 15.

The Multiplex System Diagnostics and the RT/IM Diagnostics were developed in a structured and modular fashion so additional tests could be readily added. Several of the key features of the above software test programs were:

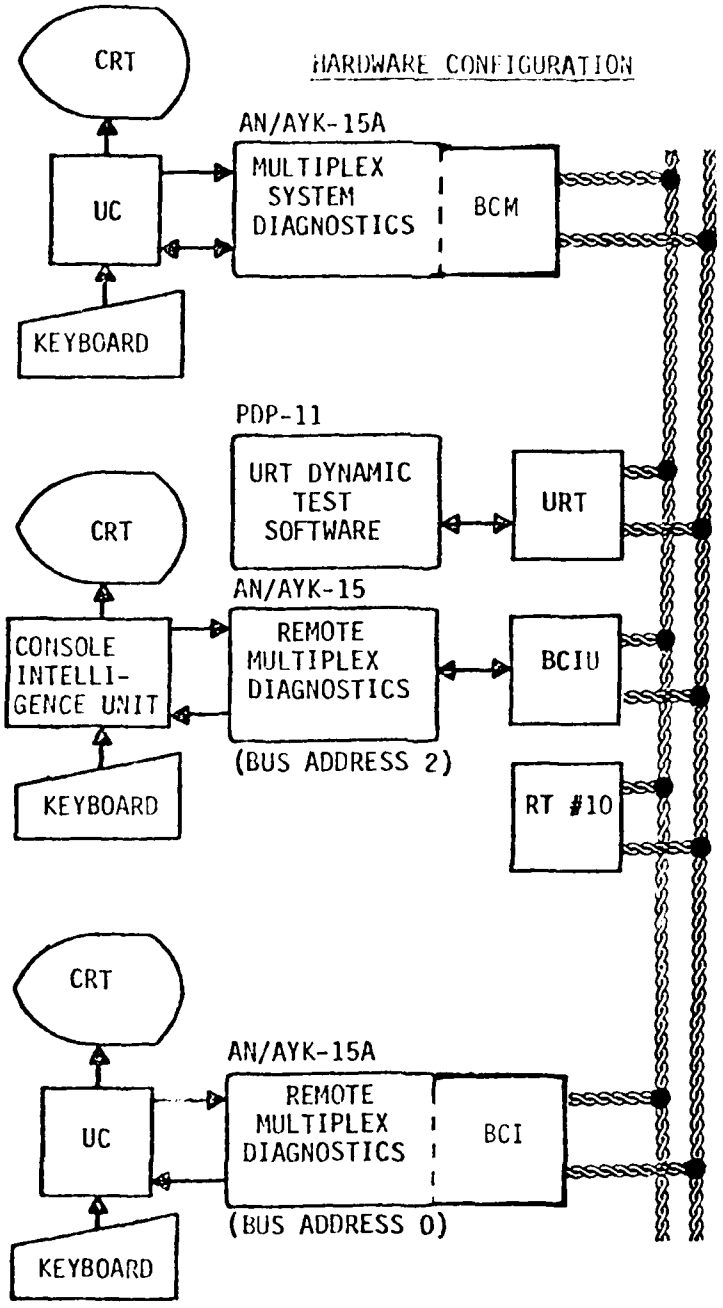
- Testing of all functional areas of the core elements under test.

- Testing of functional areas with extensive data patterns with error checking under test software control. Many failure modes were bit pattern sensitive and/or intermittent requiring a large number of data patterns at the system operating speed in order to detect and isolate.



(This runs the RT undefined Mode Codes Test, RT Initialize Terminal Mode Codes Test, and RT MTU Shutdown Mode Codes Test.)

FIGURE 47. SAMPLE HARDWARE CONFIGURATIONS



(This runs all defined tests except Remote Transmission Lockout Flag Test and Remote Reception Lockout Flag Test which require only 15A processors on the bus.)

FIGURE 47. SAMPLE HARDWARE CONFIGURATIONS (Con't)

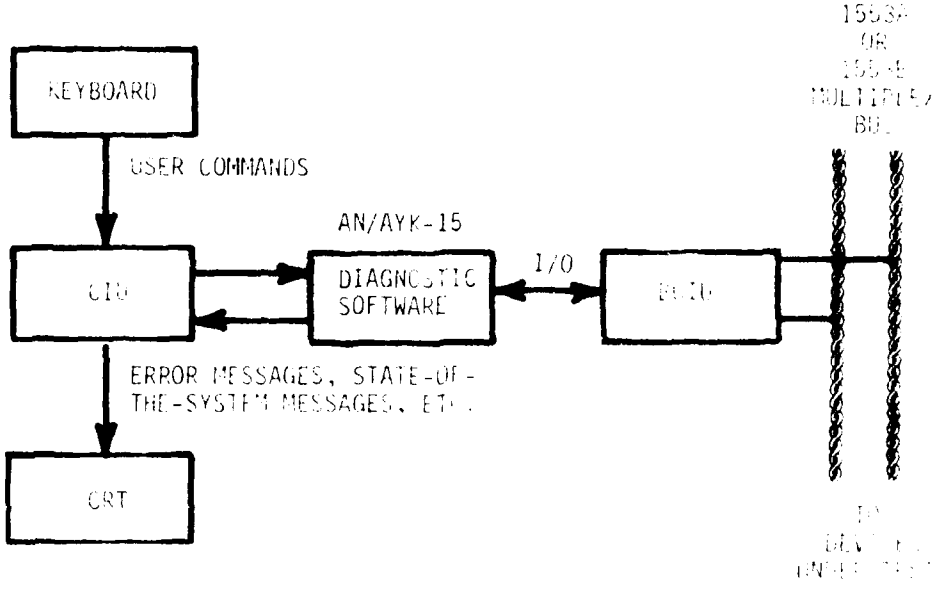


FIGURE 48. BLOCK DIAGRAM OF HARDWARE USED BY THE RT/IM DIAGNOSTICS

TABLE 15. RT/IM DIAGNOSTICS TESTS

TEST NUMBER	TEST NAME
0	Initial polling
1	Spare
2	Spare
3	Spare
4	Spare
5	RT undefined mode codes
6	RT initialize terminal mode codes
7	RT MTU shutdown mode codes
8	Spare
9	Spare
10	Spare
11	Single-ended discretes in/out
12	Differential discretes in/out
13	Switch closure in/out
14	DC Analog in/out
15	AC Analog in/out
16	Spare
17	Spare
18	Spare
19	Momentary in
20	Synchro in/out
21	Serial in/Serial out (normal operation)
22	Serial in/Serial out (parity error operation)
24	Remote terminal to Remote
25	Spare
26	Spare
27	Spare
28	Spare
29	Spare
30	Spare
31	Spare
32	Spare

- Segmenting test software to functional areas of the hardware under test. This facilitated problem isolation by allowing the test operator the option of running test software on the failing segment only.

- Providing stop on error and loop on section, subsection, and failing data pattern. This facilitated problem isolation and provided conditions under which a problem (even intermittent) could be 'scoped' or traced with a logic analyzer.

- Generating data patterns at execution rates at least comparable to the normal operation rate for the equipment under test. Many failure modes were induced only at or near maximum data rates because of noise, signal line "ringing" or "cross talk" and other high speed phenomena.

- Sequencing through all phases of the test repetitively (looping) with error checking. Capability allowed equipment testing for thermal, noise, and other intermittent errors, and provided a high degree of confidence when equipment operated successfully for extended periods of time.

- Supporting system verification prior to loading and running mission software, and supporting problem detection and isolation during mission software checkout and predemonstration phases. System integrity was established before loading mission software so mission software debugging would not be plagued with hardware problems.

- Providing structured and modular test software to facilitate development in stages and to allow inclusion of test software from various sources.

The C&D ATP performs two major functions in testing the C&D. First, it handles the interface between the C&D devices and the test software. These interface programs are called EQUIP's. Secondly, it controls execution by prompting the user and performs the actual tests. Table 16 briefly describes the programs comprising this ATP. Figure 49 presents the hardware configurations used in performing the tests.

The AN/AYK-15A Performance Monitor Interface Unit (PMIU) Acceptance Test Program is an integral part of the PMIU qualification package. The PMIU ATP consists of three functional types: (1) Interface test software, responsible for the testing of the PMIU/PDP-11 interface; (2) Control and monitor test, responsible for the testing of PMIU control and monitor functions, (3) Post run editor, responsible for the formatting and processing of data gathered during PMIU testing.

The ATP was designed so as to allow user directives via interactive or disk file inputs. This will facilitate the use of the PMIU ATP for AYK-15A software development. Figure 50 shows the PMIU system and interface requirements. Prior to execution, the ATP software must be loaded into the PDP-11 with all appropriate devices as shown in Figure 50.

TABLE 16. CONTROLS AND DISPLAYS ATP

Task	Description
EQUIPS:	
QP01IMFK'IN	IMFK input equip for RT 16.
QP02IMFK'IN	IMFK input equip for RT 9.
QP03IMFK'ARBIT	IMFK input arbitrator task.
QP04DEK1'IN	DEK-1 input equip for RT 16.
QP05DEK1'ARBIT	DEK-1 input equip for RT 9.
QP06DEK1'ARBIT	DEK-1 input arbitrator task.
QP07MMP'IN	MMP input equip for RT 16.
QP08MMP'IN	MMP input equip for RT 9.
QP09MMP'ARBIT	MMP input arbitrator task.
QP10DEK2'IN	DEK-2 input equip for RT 16.
QP11DEK2'IN	DEK-2 input equip for RT 9.
QP12DEK2'ARBIT	DEK-2 input arbitrator task.
QP13MPDG1'IN	MPDG1 status input equip for RT 16.
QP14MPDG1'IN	MPDG1 status input equip for RT 9.
QP15MPDG1'ARBIT	MPDG1 status input arbitrator task.
QP16MPDG2'IN	MPDG2 status input equip for RT 16.
QP17MPDG2'IN	MPDG2 status input equip for RT 9.
QP18MPDG2'ARBIT	MPDG2 status input arbitrator task.
QP19MFK'IN	MFK input equip for RT 16.
QP20MKF'IN	MFK input equip for RT 9.
QP21MKF'ARBIT	MFK input arbitrator task.
QP22SA8'IN	SA8 input equip for RT 16 and RT 9.
QP24SA9'IN	SA9 input equip for RT 16 and RT 9.
QP26SA11'IN	Determine the state of the PCP.
QP50IMFK'OUT	Displays data/pages on the IMFK.
QP51MMP'OUT	Outputs light commands to the MMP.
QP52SA6'OUT	SA6 output equip for RT 16 and RT 9.
TEST SOFTWARE:	
CP00INIT	Initializes data and schedules equips, calls CP01 to begin execution.
CP01TEST'SELECT	Allows the user to select major test categories and then schedule the selection. Currently only CP02 is available (C & D test).
CP02CD'TEST	Allows the user to select which C & D device he wishes to test and then schedule the selection.

TABLE 16. CONTROLS AND DISPLAYS ATTRIBUTES

Task	Description
TEST SOFTWARE Cont.	
CP05MMP'TEST	Turns on/off MMP lights when respective key hits are made.
CP10IMFK'TEST	Turns on/off IMFK lights when respective key hits are made. Displays all applicable characters on CRT.
CP15MFK'TEST	Turns on/off MFK lights when respective key hits are made. Activates all back light displays.
CP20DEK1'TEST	Echoes back on IMFK the value entered on the DEK1.
CP25DEK2'TEST	Echoes back on IMFK the value entered on the DEK2.
CP30SCU'TEST	Allows the user to select which SCU device he wishes to test and then schedule the selection.
CP31SCU'KEY'TEST	Turns on/off SCU lights when respective key hits are made.
CP32SCU'TRIGGER'TEST	Displays trigger level on IMFK.
CP33SCU'ROCKER'TEST	Displays rocker position in IMFK.
CP34SCU'CONTROLLER'TEST	Displays the numeric value of the SCU controller and terminates upon hitting the controller switch.
CP35AP'TEST	Allows the user to select which armament panel device he wishes to test and then schedules the selection.
CP36ARM'PANEL'TEST	Displays the states of AP switches on the IMFK.
CP37MASTER'ARM'TEST	Displays the state of the master arm switch on the IMFK.
CP38SALVO'SWITCH'TEST	Displays the state of the salvo switch on the IMFK.
CP40HU	Displays the state of the HUD switches of the MMP on the IMFK.
CP45MPDG'KEY'TEST	Allows the user to select which MPDG display he wishes to test.
CP46VSD'KEY'TEST	Turns on/off VSD lightswhen respective key hits are made and displays range switch state on IMFK.

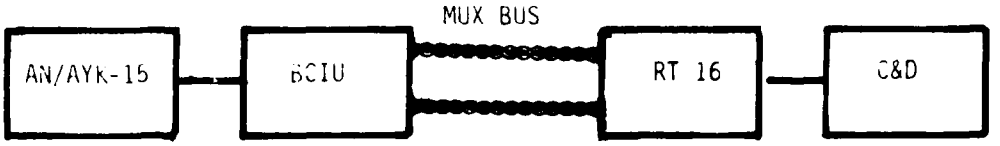
TABLE 16. CONTROLS AND DISPLAYS ATP (Con't)

Task	Description
TEST SOFTWARE Cont.	
CP47HSD'KEY'TEST	Turns on/off HSD lights when respective key hits are made and displays range switch state on IMFK.
CP48MPD1'KEY'TEST	Turns on/off MPD1 lights when respective key hits are made and displays range switch state on IMFK.
CP49MPD2'KEY'TEST	Turns on/off MPD2 lights when respective key hits are made and displays range switch state on IMFK.
CP50PCP'TEST	Allows the user to test either the input or output function of the PCP.
CP51PCP'INPUT'TEST	Displays the status of the PCP switches.
CP52PCP'OUTPUT'TEST	Turns on/off PCP lights in a pre-defined order. Also, forces the processor to issue PCP latch commands, which may be tested by the user.
CP55STICK'TEST	Displays the status of the stick switches and trigger on the IMFK.
CP60THROTTLE'TEST	Displays the status of the throttle switches on the IMFK.
CP65RT'CONTROL'TEST	Allows the user to select which RT is primary and whether the IMFK or MFK/MPD1 display is to be used.
CP70MP	Allows the user to select which function of the MPDG is to be tested.
CP71BOOT'MPDG	Allows the user to boot, load and execute an MPDG.
CP72MPDG'STATUS	Displays the status of the MPDG.
CP73WAYPOINT	Allows the user to select the various options concerning the activation and control of the map driver.
CP74TABLE'CONTRL	Allows the user to read, modify and display text tables.
CP75TASK'ASSIGNMENT	Assigns tasks to MPDG displays as a function of user input.
CP76MASTER'MODE	Allows the user to select various data sets that are used in MPDG displays. Also allows symbology to be turned on/off.

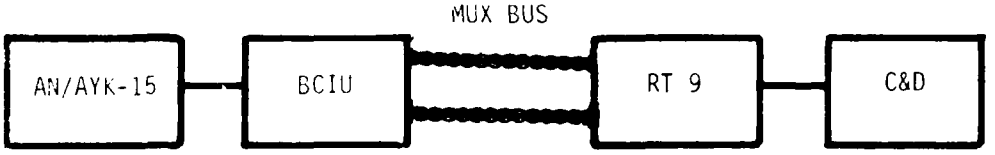
TABLE 16. CONTROLS AND DISPLAYS ATP (Con't)

Task	Description
TEST SOFTWARE Cont.	
CP90NAV'UPDATE	Calculates aircraft position based on its velocity vector and current position.
CP95WAYPT'DISPLAY'TEST	Allows the user to test the various functions associated with the waypoint display.
CP96DECLUTTER'TEST	Allows the user to select various declutter modes and observe the result on the MPDG display.
DP51MAPUPDATE	Determines which map chips need to be loaded and sends appropriate data to map driver.

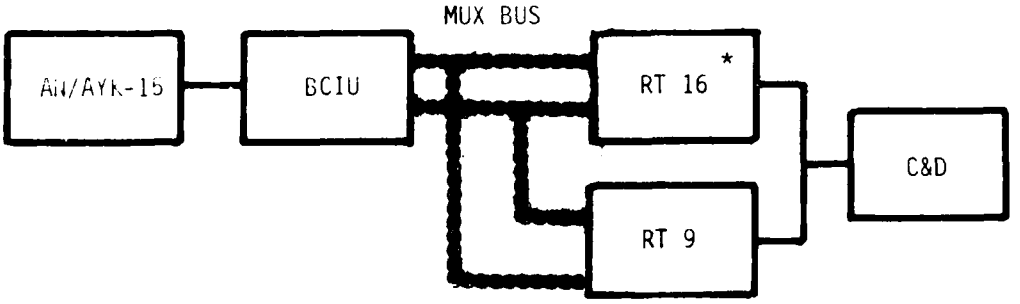
CONFIGURATION 1:



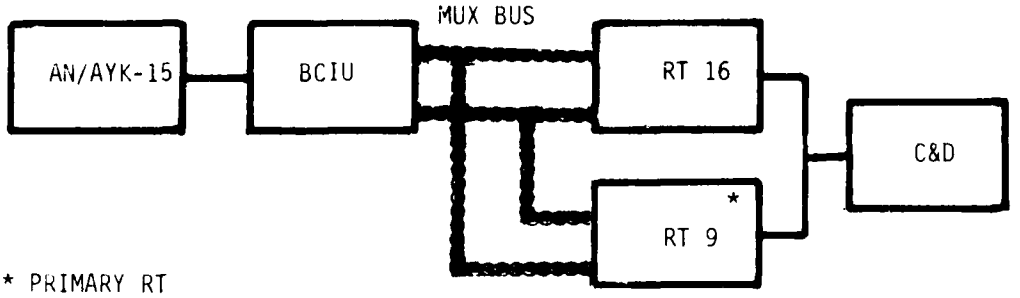
CONFIGURATION 2:



CONFIGURATION 3:



CONFIGURATION 4:



* PRIMARY RT

FIGURE 49. HARDWARE CONFIGURATIONS FOR TESTING THE C&D

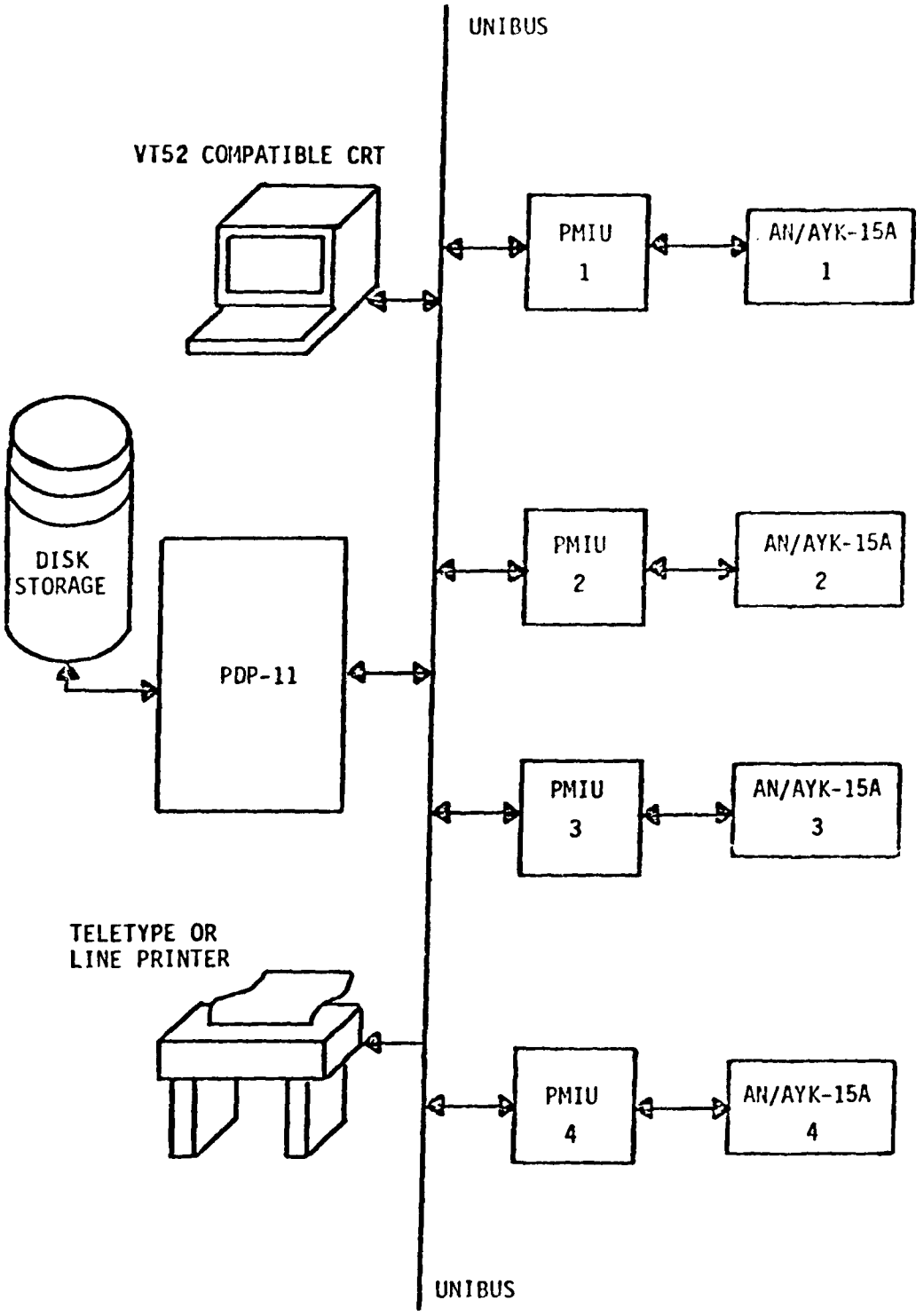


FIGURE 50. PMIU INTERFACE DIAGRAM

8.3 System Readiness Test

The System Readiness Test (SRT) is designed to quickly test the operating condition of the DAIS Software Test Stand (STS) or the Integrated Test Bed (ITB). The following systems are tested:

1. PMC and SSDF PDP-11/40s
2. Console Intelligence Units (CIU)
3. Super Control and Display Units (SCADU)
4. Remote Terminals (RTs)
5. Universal Remote Terminals (URT)
6. AN/AYK-15 processors and BCIUs or AN/AYK-15A processor/BCIs
7. Bus Monitors
8. Performance Monitor Interface Units
9. Evans and Sutherland System
10. DAIS Controls and Displays

The following group of programs forms the SRT:

1. Control file that runs with the PMC software or the PMIU software

2. Assembly language programs that run on the AN/AYK-15 or AN/AYK-15A

3. A FORTRAN program that runs on the DECsystem-10

4. Post run edit files that run on the PMC PDP-11/40

The programs work together to perform a quick check on the above hardware in less than 15 minutes; therefore, the DAIS users could run this software every morning and before demonstrations to insure the DAIS systems are in working order.

8.4 Results of Testing AN/AYK-15As from Two Contractors

This section describes the tests that were performed on the Westinghouse and Sperry Univac AN/AYK-15As during 1980 and early 1981. A summary of the test results for each vendor and a summary of the compatibility between the two implementations of the AN/AYK-15A are included.

8.4.1 Inventory of Tests

The tests performed on each AN/AYK-15A include:

- Execution of the Bootstrap Loader
- 3-Processor Executive Acceptance Test Program
- Processor Acceptance Test Program
- Multiplex System Diagnostics
- Bus Control Module (BCM) Error Response Tests
- Input/Output and Internal Tests
- Miscellaneous Tests

All of the tests were performed with both AN/AYK-15/BCIUs and other AN/AYK-15As from both vendors.

8.4.1.1 Bootstrap Loader

The Bootstrap Loader was used to load all of the 15A test software programs from the PDP-11/URT to the 15A processor memory over the 1553b multiplex bus as illustrated in Figure 51. The software program in the PDP-11 was used to simulate a mass memory device on the multiplex bus. Software resident in the upper core of the 15A memory was then used to initiate the BCM as a bus controller and transfer application and test software and data from the mass memory device to the 15A processor memory. 64K words can be transferred in approximately 15 seconds using this technique.

8.4.1.2 3-Processor Executive Acceptance Test Program (ATP)

The 3-Processor Executive ATP with the AN/AYK-15A as the unit under test (UUR) was executed using the test configuration shown in Figure 51.

8.4.1.3 Processor Acceptance Test Program (ATP)

The Processor ATP (15A, 1750) was executed using the test configuration shown in Figure 52. This set of software programs was used to test the entire instruction repertoire and functional capability of the 15A processor.

8.4.1.4 Multiplex System Diagnostics

The Multiplex System Diagnostic tests were executed using the test configuration shown in Figure 53. The software programs were used to test the Master and Remote mode functional requirements of the 15A BCM and the BCM/processor interface. The software in the PDP-11 was used to simulate multiple RTs via the URT.

8.4.1.5 BCM Error Response Tests

BCM Error Response tests were conducted to test the performance of the 15A BCM when errors occur on the multiplex bus and were accomplished using the test configuration shown in Figure 54.

8.4.1.6 Input/Output (I/O) and Internal Tests

I/O and Internal tests were conducted to test the input and output requirements of the AN/AYK-15A (e.g., programmed input/output, direct memory access, interrupts, etc.) and to verify the resolution of unique problems encountered during the debugging stage of the 15As. The test configuration for the I/O tests is shown in Figure 55.

8.4.1.7 Miscellaneous Tests

Investigations and measurements were made to obtain additional information on the performance of the AN/AYK-15As. Of particular interest are the measurements of memory access times with CPU, DMA and BCM memory contention on the Sperry Univac AN/AYK-15A and the measurements of the Master mode BCM start and stop times for both the Westinghouse and Sperry Univac AN/AYK-15As. A summary of these measurements is given in section 8.5.2.3.

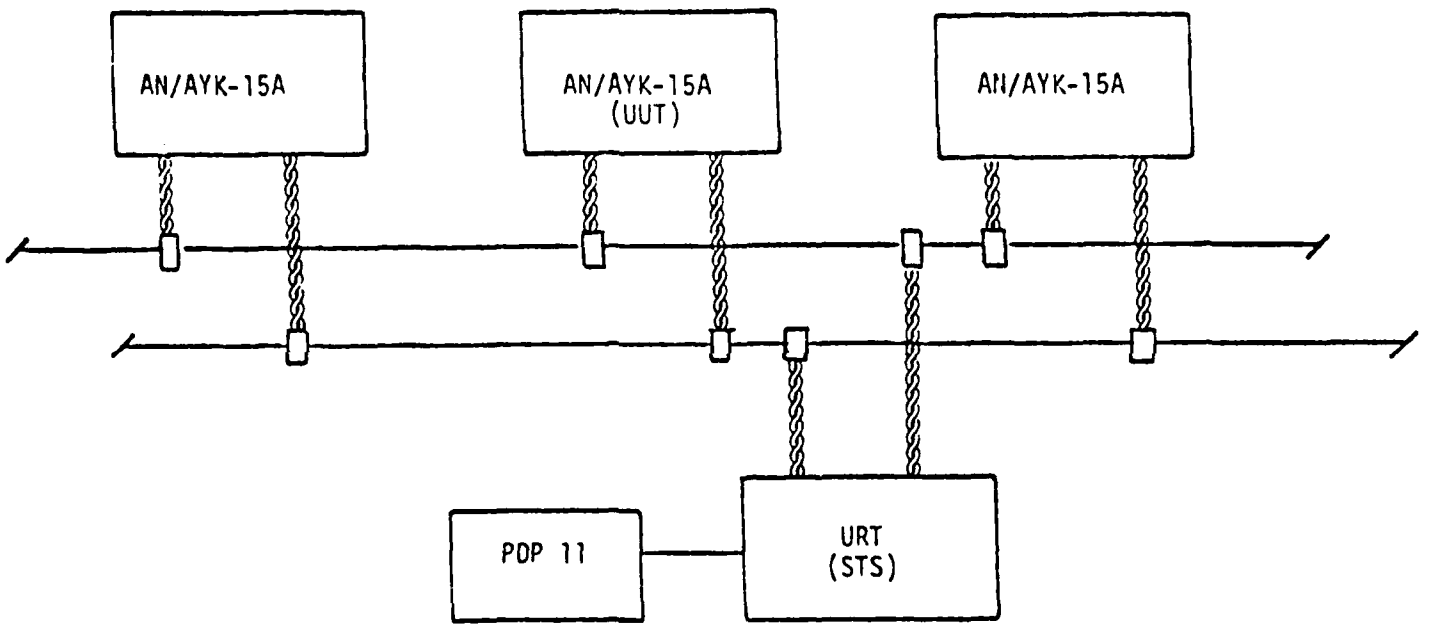


FIGURE 51. TEST CONFIGURATION FOR THE 3-PROCESSOR EXECUTIVE ATP

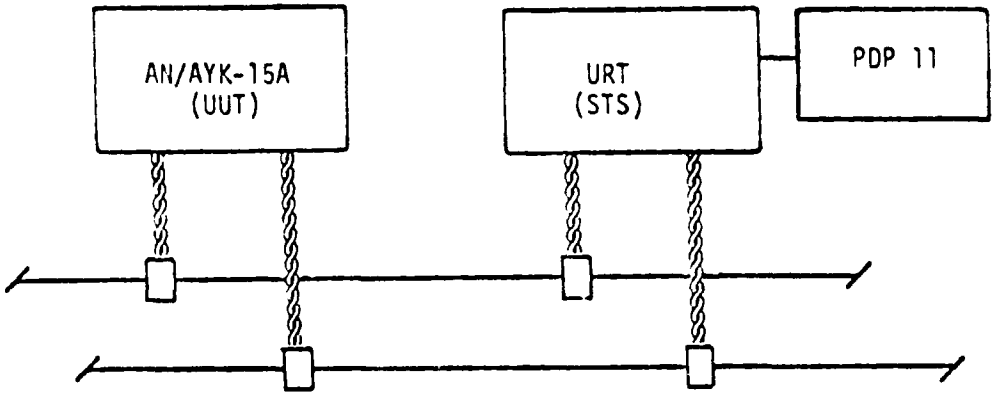


FIGURE 52 TEST CONFIGURATION FOR THE ONE-PROCESSOR AY

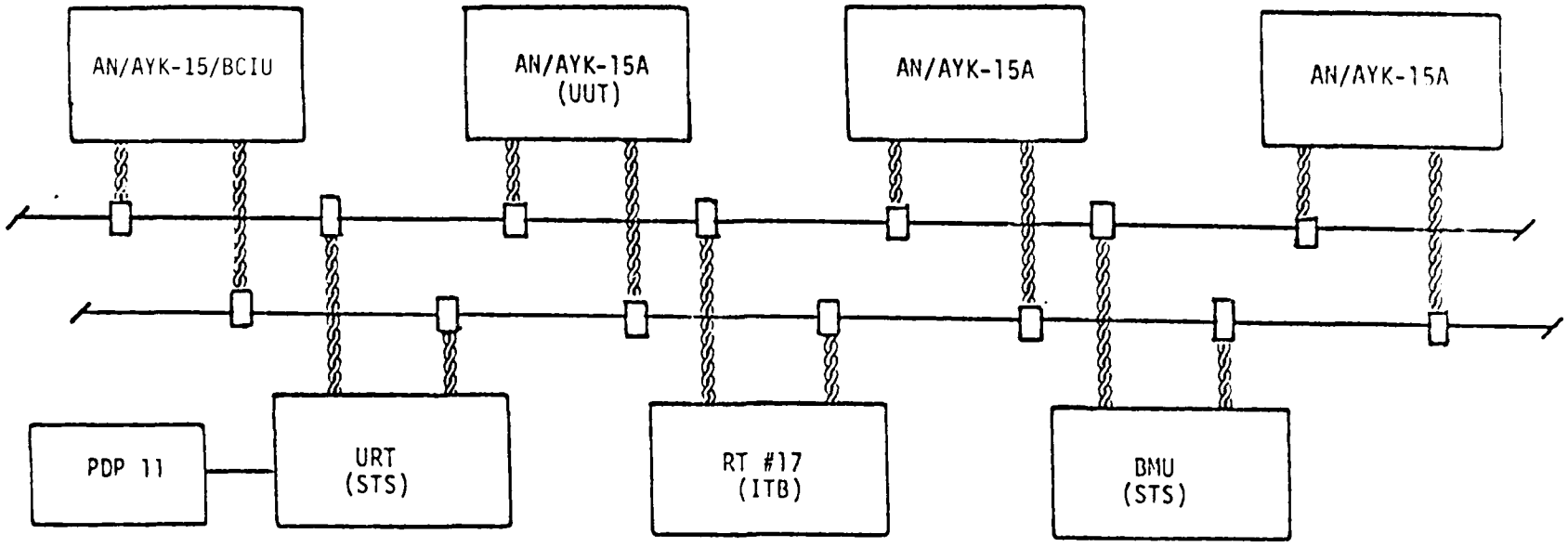


FIGURE 53. TEST CONFIGURATION FOR MULTIPLEX DIAGNOSTIC TESTS

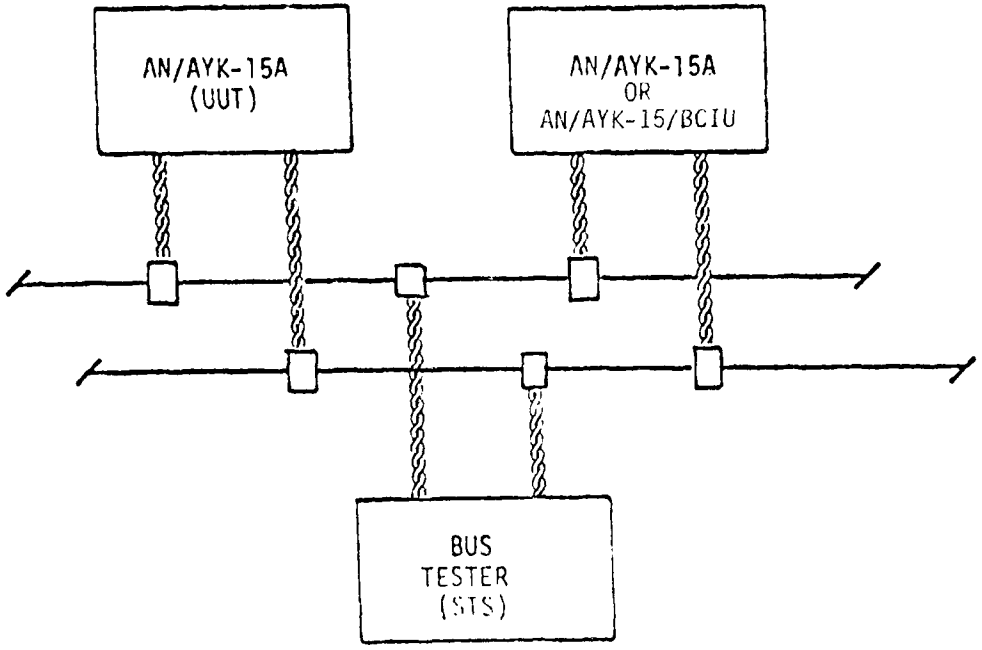


FIGURE 54. TEST CONFIGURATION FOR BCM ERROR RESPONSE TESTS

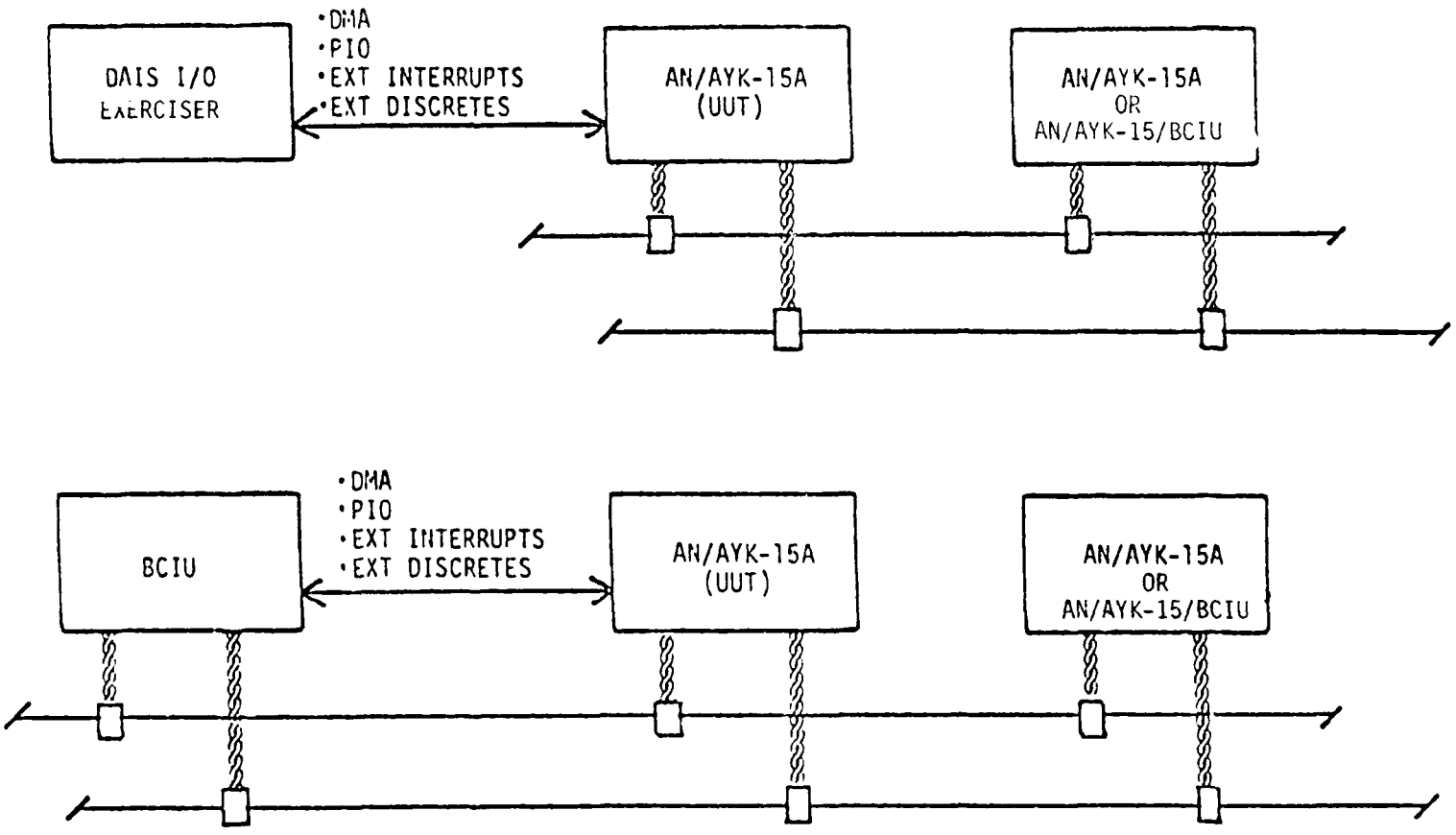


FIGURE 55. TEST CONFIGURATIONS FOR THE I/O TESTS

8.4.2 Results of Testing

This section summarizes the overall test results for the Westinghouse and Sperry Univac AN/AYK-15As. These results are based on the performance of the units after several iterations of problem isolation and repair. Specific test results associated with the individual units were documented in a series of informal test reports.

8.4.2.1 Test Results for Westinghouse AN/AYK-15A

The Bootstrap Loader and the 3-Processor Executive ATP were executed without incident. Tables 17 through 19 indicate the results of the Processor ATP, Multiplex System Diagnostics, and BCM Error Response tests, respectively, that failed on the Westinghouse AN/AYK-15A. Table 20 summarizes the results of the I/O tests.

8.4.2.2 Test Results for Sperry Univac AN/AYK-15A

The Bootstrap Loader and the 3-Processor Executive ATP were executed without incident. Tables 21 through 23 indicate the results of the Processor ATP, Multiplex System Diagnostics, and BCM Error Response tests, respectively, that failed on the Sperry Univac AN/AYK-15A. Table 24 summarizes the results of the I/O tests.

8.4.2.3 Results of Miscellaneous Tests

The results of memory access times are summarized in Table 25 and were measured on the Sperry Univac AN/AYK-15A prototype early in May of 1982. The results of measurements taken on the AN/AYK-15A from both vendors for the Master mode BCM start and stop times under various conditions are summarized in Table 26.

8.4.3 AN/AYK-15A Compatibility

One of the original goals cited for the dual source procurement of the AN/AYK-15A digital processor was to ascertain the feasibility of procuring compatible 15As from two vendors utilizing different implementations to meet the same requirements. To achieve this result, the development specification had necessarily be:

1) general enough to support rather than supplant the design and implementation of the AN/AYK-15A, and

2) specific enough to avoid multiple interpretations and unclearly specified requirements that lead to incompatible performance.

The addition of the specification changes was made, in part, to improve the specification and eliminate those areas of the specification where incompatibility issues had been observed between the two vendors.

A simple measure of compatibility between the Westinghouse and Sperry Univac AN/AYK-15As involves the substitution of either 15A into a system configuration without changes to system performance and without

TABLE 17. WESTINGHOUSE AN/AYK-15A SUMMARY OF FAILING PROCESSOR TESTS

TEST NAME	REASON FOR FAILURE
FSB FS FSR	Results not accurate to the least significant bit on floating point subtracts.
FDB FD FDR EFD EFDR	Results not accurate to the least significant bit on floating point divides.
BRX	Results not accurate to the least significant bit on floating point subtracts and floating point divides.
HANGT TEST	Execution of an illegal opcode sometimes causes CPU Memory Protect and Illegal Instruction bits to be set in the Fault Register--depending on the contents of the general registers.
BCMGEN TEST	Some of the BCM control and general registers are tied together.
Illegal I/O Test	Execution of an illegal I/O instruction (output opcode with an input operand) that deals with the BCI control register 0 or BCI general register 0 causes BCI to generate an interrupt when the GO bit in the PCR is zero.

TABLE 18. WESTINGHOUSE AN/AYK-15A SUMMARY OF MULTIPLEX DIAGNOSTIC TESTS

TEST NUMBER	TEST NAME	MASTER	REMOTE
INIT	Initialization	P	P
0	Initial Polling	P	P
2	BCI Undefined Mode Commands	P	P
3	BCI MTU Shutdown Mode Commands	P	P
4	BCI Mode Commands With Interrupts	F(Note 1)	P
5	RT Undefined Mode Commands	P	N/A
6	RT Initialize Terminal Mode Commands	P	N/A
7	RT MTU Shutdown Mode Commands	P	N/A
8	URT Mode Commands	P	N/A
9	Master Async to Remote Sync Transfers	P	P
10	Master Sync to Remote Async Transfers	P	P
11	BCM State Transitions	P	N/A
12	BCM Internal Operations	P	N/A
13	Remote Transmission Lockout Flag	P	P
14	Remote Reception Lockout Flag	P	P
15	Lockout Flag Set/Reset	P	P
16	Master to Remote Synchronous Transfers	P	P
17	Master to Remote Synchronous Transfers	P	P
18	Remote to Remote Synchronous Transfers	P	P
19	Master to Remote Synchronous Transfers	P	P
20	Remote to Remote Synchronous Transfers	P	P
21-24	Mass Memory Tests	-	-
25	Remote BCI Self-Test	P	P
26	Master BCI Self-Test	P	N/A
27	Lockout Flag Duration	P	P
28	Lockout Flag on Receive	P	P
29	Lockout Flag on Transmit	P	P
30	MTU Shutdown Command	F(Note 2)	P
31	Bus Active Bit	P	P
32	Bus List	P	P
SUBADDRESS FOR MODE CODES			
0		P	P
31		P	P
MULTIPLEX BUS			
A		P	P
B		P	P

P: Pass
F: Fail
-: Not Run
N/A: Not Applicable

Note 1: Specification Interpretation Problem (See section 8.5.3)

Note 2: Westinghouse 15A does not incorporate SCN2 changes to SA 421 205

TABLE 19. WESTINGHOUSE AN/AYK-15A BCM ERROR RESPONSE PROBLEMS

MODE	MESSAGE PROTOCOL	INDUCED ERROR	EXPECTED RESULT	ACTUAL RESULT
Master	All	Bit count high in status word from remote	Status error detected	No error detected
Master	All	Multiple status word exceptions	Retries performed in accordance with SCN 2 to SA 421 205	Retries not performed in accordance with SCN 2 to SA 421 205
Master	Terminal-to-Terminal	Bit count high in data word	Invalid data bit set in BITR	Data parity error set in BITR
Master	Terminal-to-Controller	Remote XMTR busy	XMTR busy and received data error bits set in ISR	XMTR busy bit set in ISR only
Master	Terminal-to-Controller	Word count high	Received data error bit set in ISR	Intermittently no error detected
Master	Terminal-to-Terminal	Remote XMTR busy	XMTR busy, received data error and receiver status error bits set in ISR	XMTR busy bit set in ISR only
Master	Terminal-to-Terminal	Busy sit set in remote XMTR status word & SR bit set in remote recv status word. Data is transmitted.	PCI, XMTR busy, and RCVR asynchronous request bits set in ISR.	Only XMTR busy bit set in ISR and data parity error bit set in BITR. Jams bus on retries.
Remote	Controller-to-Terminal	High bit count in command word	Command ignored	intermittently sets data parity error and invalid data bits in BITR and updates LCR.

TABLE 19. WESTINGHOUSE AN/AYK-15A BCM ERROR RESPONSE PROBLEMS (Con't)

MODE	MESSAGE PROTOCOL	INDUCED ERROR	EXPECTED RESULT	ACTUAL RESULT
Remote	Controller-to-Terminal	Bit count high in data word	Invalid data bit set in BITR	Data parity error bit set in BITR
Remote	Controller-to-Terminal	Word count high	Word count high bit set in BITR	Data parity error bit set in BITR. Status response jams bus.
Remote	Controller-to-Terminal	First data word sync inverted	Invalid data bit set in BITR	Word count low bit set in BITR

TABLE 20. WESTINGHOUSE AN/AYK-15A SUMMARY OF I/O AND SPECIAL TESTS

TEST NAME	PASS/FAIL	REASON FOR FAILURE
Timer A	P	
Timer B	P	
Trigger Go	P	
PIO	P	
No-Op	P	
Discrete Input/Output	P	
TBCR Instruction	P	
BCM "PIO/DMA"	P	
CPU Memory Protection	P	
BCIU Standalone	P	
BCM and BCIU Wrap	P	
External Interrupts	P	
Machine Error Interrupts	P	
DMA Read	P	
DMA Write	P	

P: Pass F: Fail

TABLE 21. UNIVAC AN/AYK-15A SUMMARY OF FAILING PROCESSOR TESTS

TEST NAME	REASON FOR FAILURE
BENCHMARK	Processor throughput measured at only 385 KOPS.
MEMTST	The CPU memory protect interrupt is posted after execution of the next instruction. This delay causes the software to assume that write protect did not occur and display an error message.
HANG HANGT	The delay in posting of the CPU memory protect interrupt (level 1 interrupt) causes problems when the next instruction also causes a level 1 interrupt which is posted immediately. A level 1 interrupt is therefore generated on top of another level 1 interrupt which causes the software to "hang-up" in the interrupt handler.
DB BRX (DBY) IM (DIM) D DR DD DDR	The CPU is not generating a fixed point interrupt for the smallest negative number (8000 or 80000000) divided by the largest negative number (FFFF or FFFFFFFF).

TABLE 22. UNIVAC AN/AYK-15A SUMMARY OF MULTIPLEX DIAGNOSTIC TESTS

TEST NUMBER	TEST NAME	MASTER	REMOTE
INIT	Initialization	P	P
0	Initial Polling	P	P
2	BCI Undefined Mode Commands	P	P
3	BCI MTU Shutdown Mode Commands	P	P
4	BCI Mode Commands With Interrupts	P	P
5	RT Undefined Mode Commands	P	N/A
6	RT Initialize Terminal Mode Commands	P	N/A
7	RT MTU Shutdown Mode Commands	P	N/A
8	URT Mode Commands	P	N/A
9	Master Async to Remote Sync Transfers	P	P
10	Master Sync to Remote Async Transfers	P	P
11	BCM State Transitions	F(Note 1)	N/A
12	BCM Internal Operations	P	N/A
13	Remote Transmission Lockout Flag	P	P
14	Remote Reception Lockout Flag	P	P
15	Lockout Flag Set/Reset	P	P
16	Master to Remote Synchronous Transfers	P	P
17	Master to Remote Synchronous Transfers	P	P
18	Master to Remote Sybchronous Transfers	P	P
19	Master to Remote Synchronous Transfers	P	P
20	Master to Remote Synchronous Transfers	P	P
21-24	Mass Memory Tests	-	-
25	Remote BCI Self-Test	P	P
26	Master BCI Self-Test	P	N/A
27	Lockout Flag Duration	P	P
28	Lockout Flag on Receive	P	P
29	Lockout Flag on Transmit	P	P
30	MTU Shutdown Command	P	P
31	Bus Active Bit	P	P
SUBADDRESS FOR MODE CODES			
0		P	P
31		P	P
MULTIPLEX BUS			
A		P	P
B		P	P

P: Pass
F: Fail
-: Not Run
N/A: Not Applicable

Note 1: BCM does not generate an interrupt to the processor for one internal operation (minimal impact)

TABLE 23. UNIVAC AN/AYK-15A BCM ERROR RESPONSE PROBLEMS

MODE	MESSAGE PROTOCOL	INDUCED ERROR	EXPECTED RESULT	ACTUAL RESULT
Master	All	Bit Count High In Status Word From Remote	Status Error Detected	No Error Detected
Master	All	Multiple Status Word Exceptions	Retries Performed in Accordance With SEN 2 to SA 421 205	Retries Not Performed in Accordance With SEN 2 to SA 421 205
Master	Terminal-To Terminal	Bit Count High In Data Word	Invalid Data Bit Set in BitR	Data Parity Error Bit Set in BitR
Master	Terminal-To Controller	Word Count Low	Retry Gap Time on the Order of 60 μsec.	Retry Gap Time of 180 μsec.
Remote	Controller-To Terminal	Bit Count High in Data Word	Invalid Data Bit Set in BitR	Data Parity Error Bit Set In BitR
[illegible]te	Controller-To Terminal	Data Word Sync Inverted	Invalid Data Bit Set in BitR	If first or last Data Word Sync is inverted, then Word Count Low Bit is set in BitR. *
Remote	Controller-To Terminal	High bit count on Command Word	No error detected	Command not received

*If any other data word sync is inverted, then tag word lockout flag remains set and BCM hangs up

TABLE 24. UNIVAC AN/AYK-15A SUMMARY OF I/O AND SPECIAL TESTS

TEST NAME	PASS/FAIL	REASON FOR FAILURE
Timer A	P	
Timer B	P	
Trigger Go	P	
PIO	F	
No-Op	P	
Discrete Input/Output	P	
TBCR Instruction	P	
BCM "PIO/DMA"	F	
CPU Memory Protection	P	
BCIU Stand-alone	F	The MSB of the PIO Command Word is not set for a PIO input. (See section 8.5.3)
BCM and BCIU Wrap	F	
External Interrupts	P	
Machine Error Interrupts	P	
DMA Read	F	Intermittent DMA Channel Parity Errors Occur When CPU is running.
DMA Write	P	

P: Pass F: Fail

TABLE 25. UNIVAC AN/AYK-15A MEMORY ACCESS TEST

CONDITION	NUMBER OF MEMORY ACCESSES PER MICROSECOND			
	CPU	BCI	DMA	TOTAL
CPU only	1.02	-	-	1.02
BCI only	NOT TESTED*			
DMA only	-	-	[illegible]	[illegible]
CPU & BCI	0.95	0.08	-	1.03
CPU & DMA	0.82	-	[illegible]	1.07
BCI & DMA	NOT TESTED*			
CPU, BCI & DMA	0.77	[illegible]	0.24	1.09

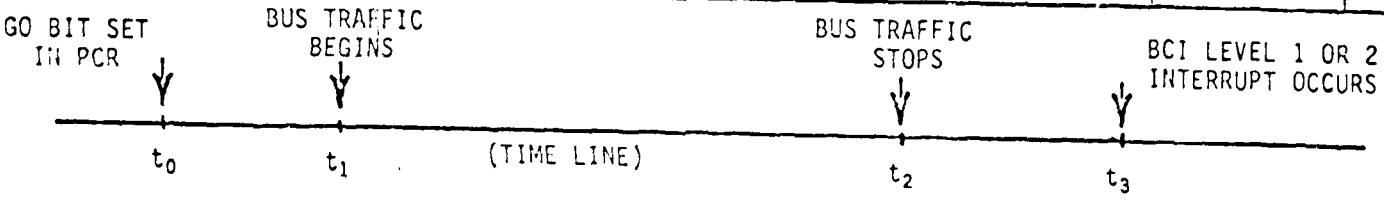
*BCI will not run unless CPU is running

Basis

- CPU performing a series of double load, double add, double store instructions.
- BCM performing a series of no-op instructions.
- DMA write operations performed repetitively using DAIS I/O exerciser.

TABLE 26. MASTER 15A BCM START AND STOP TIMES

FIRST MESSAGE	CONDITION	MEASURED	RESULTS (in μsec)	
			UNIVAC #4	WESTINGHOUSE #2
CONTROLLER-TO-TERMINAL WITH 32 WORDS	GO BIT SET IN PCR	$t_1 - t_0$	66	35
	GO BIT RESET DURING FIRST MESSAGE; NEXT INSTRUCTION IS EITHER A LINK OR A TERMINAL-TO-TERMINAL INSTRUCTION	$t_3 - t_2$	52	28
	GO BIT RESET DURING FIRST MESSAGE, NEXT INSTRUCTION IS EITHER A CONTROLLER-TO-TERMINAL OR A TERMINAL-TO-CONTROLLER INSTRUCTION	$t_3 - t_2$	63	30
	HALT INSTRUCTION FOLLOWS FIRST MESSAGE	$t_3 - t_2$	75	37
	SERVICE REQUEST BIT SET IN STATUS FROM REMOTE	$t_3 - t_2$	66	28
TERMINAL-TO-CONTROLLER WITH 32 WORDS	MASTER RECEIVES DATA ON SUBADDRESS 30 (ASYNCHRONOUS)	$t_3 - t_2$	68	28



hardware or software modifications. [illegible] ibility between the AN/AYK-15A acceptable--demonstrating the [illegible] based on a single development specification.

Nevertheless, weaknesses in the [illegible] have led to some minor incompatibilities. Neither the Westinghouse nor Sperry Univac incorporate all of the changes to the AN/AYK-15A design specification, other incompatibilities remain between the two implementations of the AN/AYK-15A. In most cases, these incompatibilities can be avoided by choosing the correct conventions for the 15A software programs and the other core hardware that interfaces to the 15A.

The general areas of incompatibility between the Westinghouse and Sperry Univac AN/AYK-15A are summarized as follows:

- User Console Interface - Functionally, the user consoles can be interchanged between the 15As. However, when the Westinghouse user console is connected to the Univac 15A, or vice versa, some of the user console functions, e.g. clear processor, do not result in identical performance.

- Throughput - Although the specification required a processor throughput of at least 400 KOPS and a Master CPM intermessage gap time of 15 seconds or less, the specification does not address other gap times--some of which have been measured as shown in Figure 6. [illegible] in these gap times between the Westinghouse and Sperry Univac 15A, coupled with the fact that the Univac 15A achieves a processor throughput of only 240 KOPS, causes a significant increase in the executive overhead time for DAIS Mission Data when the Univac 15A is the Master processor.

- Programmed Input/Output (PIO) - The original wording of the 15A development specification for this external interface was ambiguous with respect to assertion of the [illegible] of the PIO address for input from an external device. Since the DAIS BCU, the [illegible] PIO command is an input or output, it was determined that only the Westinghouse implementation would interface with the BCU. Although this problem has been rectified, failure to incorporate this change makes the two 15As incompatible. Note, however, that the external interface has sufficient control lines to avoid this incompatibility in the future.

- Bootstrap ROM - The specification is unclear as to whether the ROM should be disabled immediately after a Disable ROM instruction, or disabled after the instruction following a Disable ROM instruction. As a result, the Westinghouse and Univac 15As disable the ROM at different times. The ROM can be programmed to avoid this incompatibility.

- Exception Response - There is some ambiguity in the specification regarding the setting of the RDE bit in the ISR when a remote transmitter is busy. The Westinghouse 15A, upon receiving a busy status response from a remote transmitter, does not expect data and, therefore, sets only the XBSY bit in the ISR. The Univac 15A, on the other hand, continues to expect data and, therefore, sets both the XBSY and RDE bits in the ISR. Since the ICR codes for this type of response are identical in either case, the problem becomes trivial. However, multiple status word exceptions and/or errors on the bus (see Tables 19 and 23) can lead to incompatible status response handling in the master 15As.

- Power-On Initialization - Global memory protection of the 15A processor after a power-on initialization or a user console clear command is treated differently in the two implementations of the AN/AYK-15A. SCN 2 to SA 421 205 clarifies the proper requirements.

8.5 Fiber Optics Multiplex System Integration and Test

The AVSAIL fiber optics integration plan was initiated to investigate the feasibility of time-division multiplexed communication over a fiber optic bus. Because of its inherent qualities (i.e. noise immunity, security, electrical isolation, low loss and high bandwidth), fiber optics was chosen as a desirable transmission medium for MIL-STD-1553B.

The successful completion of the AVSAIL Fiber Optics integration plan would make possible the operation of real-time software simulation supported by an optically coupled network of up to 32 MIL-STD-1553B devices.

Interfacing between the MIL-STD-1553B Bus and an optical network requires several considerations. Among them is the conversion of the bi-state (one, zero) optical format to a tri-state (plus, minus and zero) Manchester format. Because of this state transition, another consideration is introduced: delay. In order to discern between an end of message or a command (status word followed by a data sync), the optical receiver must wait a finite amount of time before deciding whether the Manchester output should return to zero (end of message) or remain at a minus value (data sync detected).

Manchester-to-optical encoding is performed by modulating the optical transmitter with the positive half of the Manchester waveform (plus, zero converts to one, zero).

The hardware implementation of this plan has been broken down into several different tasks assigned to various contractors. Singer Kearfott was contracted to provide optical translator units capable of interfacing any present or future MIL-STD-1553B device to a fiber optic network. Westinghouse and IBM contracted separate efforts to produce embedded fiber optic interfaces for AN/AYK-15A processors. Under a separate contract, IBM developed a star coupler that effectively ties all available optical

Sperry Univac provided two prototype optical interface modules to AFWAL for evaluation. These modules were [illegible] tested for optical output power and bit error rate (BER) versus optical [illegible].

Although the Univac modules had acceptable optical output power, a discontinuity was noted in the bit error rate measurement.

With both modules, the BER was acceptable in both the upper and lower optically received power ranges, but a total loss of communication was observed between them.

After verifying the existence of the problem with the AFWAL, the modules were returned to Univac for analysis. The modules were still unavailable for retesting at the termination of this contract.

At the termination of this contract, the Westinghouse optical modules were unavailable for testing.

During the first quarter of 1981, testing and development on a unit level basis was conducted on the eight optical translator units [illegible] delivered by Singer Kearfott.

Certain problematic conditions existed concerning the TU compatibility with the DAIS multiplex bus specification--the bus polarity and termination in particular. After these problems were resolved, work efforts continued in the areas of hardware debugging and unit level characterization. Under a combined effort involving IRW and AFWAL, OTU-1 that had been characterized underwent an upgrading process whereby the light emitting diodes were replaced with high power, high reliability components. This provided a greater signal to noise ratio and, theoretically, an improved bit error rate.

Because the optical star coupler was not available for implementation, OTU testing was conducted on a partial system level evaluation only. The results of OTU testing are documented in IRW publication 6402-[illegible]-01.

The Fiber Optic Evaluation Software is the support software required to perform system level evaluation of the fiber optic bus and associated fiber optic Translator Units (OTUs). The Fiber Optic Evaluation Software consists of six functional software routines:

a. Command Generation Software, responsible for control of the Master Processor/BCIU for the [illegible] Transport Delay Measurement,

b. Multiplex Software, responsible for control of the Master Processor/BCIU for the Data Transfer Test,

c. Remote Software, responsible for control of the Remote Processor/BCIU(s) for all tests,

d. Optical Dynamic Range Test (ODRT) #1 Software, responsible for control of the Master #1 Processor/BCIU for the Optical Dynamic Range Test,

e. ODRT #2 Software, responsible for control of the Master #2 Processor/BCIU for the Optical Dynamic Range Test, and

f. Word Error Rate Test (WERT) Software, responsible for control of the Master Processor/BCIU for the Word Error Rate Test.

9.0 SOFTWARE ASSESSMENT

9.1 Application Software Assessment

Both the J73/1 and the J73 versions of the application software were assessed. The assessment addressed the following:

- compliance of the software to specifications
- performance of the software
- adherence of the code to standards
- usage of the JOVIAL Compiler and the DAIS Executive

The specifications, the code, the program algorithms, and the implementation were evaluated. Concentration was on the documentation, and it was evaluated against strict standards. Therefore, the discrepancies found indicate deviations from the standards and do not necessarily reflect on the usability of the product. A few minor inadequacies exist in the application software, but these are not serious when viewed in the perspective of the entire system.

The discrepancies found in the documentation include the following:

- deviation from format specified in MIL-STD-490, MIL-STD-480
- typographical errors
- inconsistent organization among documents

The documentation is, however, extensive, complete and accurate.

Problems found in the software include the following:

- some poorly-commented code
- some large program modules

The software does, however, function [illegible]

9.2 DMSS Assessment

The DMSS software was thoroughly assessed to evaluate the following:

- documentation completeness, accuracy, and usability
- acceptance test results
- algorithms and satisfying requirements
- software usability, maintainability, and portability
- overall design and user interface

Overall, the DMSS software and documentation was judged to meet or exceed most requirements. The following problems found are minor and have only a small impact:

- User's Manual lacks background information
- Format of tables and of some documents is confusing
- Interface and functional test was delayed and is not fully documented
- Some coding does not conform to standards

The DMSS does function as required. The evaluation shows this software to be very usable, maintainable, modifiable and portable.

9.3 PMC Assessment

The PMC assessment consists of the following five sections:

1. Part I specification review and assessment
2. User's manual review and assessment
3. PMC system capabilities assessment
4. Assessment of PMC capability to support system testing
5. PMC usability assessment

The assessment was performed by a five-member team. Upon the completion of the testing an assessment report was written to the DAIS Document Control Board (DCB).

The PMC assessment resulted in the following recommendations:

1. Keep the current system, as is, for possible future use in the AN/AYK-15 ITB/STS. Most of the basic functions work and might be useful for future debugging/development. In particular, the team feels it could be useful for single processor module interface testing. Additional development efforts to improve the current system are not recommended.

2. For future DAIS developments, do not use the current PMC as a baseline. The concepts and some of the functions may be useful.

3. For future PMC-type developments, consider small, easy to use, stand-alone tools which execute fast and will satisfy system requirements. Insure that needed capabilities are provided. Avoid complex functions which may not be used. User instructions should be well documented and easy to follow.

During the assessment the following basic problems were discovered:

1) The PMC is a large software system. It contains a number of features that are complex and were difficult to implement. For example, interface with J73/I to allow debugging at the HOL level; start/stop/restart/ of the system, etc. As such, it is difficult to use, some of the functions do not work, some produce incorrect results and the system crashes under certain conditions. It should be noted that the hardware, such as the BMU and SCADU, also impose some PMC limiations. Since its delivery/installation, the PMC has never been used as a system development/debugging tool. Users have used some of the functions that the PMC performs, but have not done so through the PMC, i.e. they have used the functions in a manual stand-alone mode.

2) The system lacks several capabilities needed for system development/debugging. The major capabilities needed are: 1) [illegible] analysis; 2) Saving, restoring the [illegible] rate and registers, etc. in processors, BCEUs, and RIs; and 3) Tracing of bus traffic concurrently with process data.

3) For large system developments/testing, the PMC lacks sufficient storage for data logging and does not contain any post-run analysis capabilities. For a long mission simulation run, large amounts of data may need to be collected, then a program would be needed to 'reduce' this data, selectively, into readable form.

4) A usable PMC "type" of system should be an integral part of software/hardware development/testing in a DAIS type of laboratory environment. Difficult problems which may take weeks to find may be solved quickly with proper PMC tools.

Based on the above problems, seven lessons learned are presented:

1) Software tools, such as PMC, should be constrained in size such that they are well understood and all features can be used with a minimal operator effort.

2) The development requirements should be established in concert with the full integrated support facility requirements and capabilities considering such aspects as the integration of the [illegible] mode data recording, physical configuration of the support facility, etc.

3) The need in the future is to develop software support tools which are easy, stand-alone tools--yet provide appropriate interfaces to other tools, provide needed capabilities, avoid complex functions which may not be used as expressed in the PMC assessment report.

4) Software tools must be integrated [illegible] of the established requirements [illegible] properly.

5) Software tools must be accompanied with adequate [illegible] that operators need not consult the software tool developer for [illegible]

6) Timely reduction of large amounts of data into a form which is easily understood and usable is a requirement which must be satisfied for adequate system testing to be accomplished.

7) Several small, less complex, software tools, performing selected PMC functions are essential to software/hardware development, testing and maintenance.

10.0 MISSION DEMONSTRATION

The overall DAIS objectives are to (1) reduce unnecessary development proliferation, (2) improve operational efficiency including availability/maintainability, and (3) improve flexibility to changes. These objectives must be achieved without sacrificing the mission avionic functional capability attainable with conventionally configured complements of sensor and weapon subsystems. The satisfaction of these objectives and the requirements for acceptable system functional capability have been illustrated by successfully operating DAIS under simulated mission conditions. Specifically, the mission demonstration has illustrated: (1) the pilot interface with Controls and Displays (C&D), (2) the operation of the processors, mission software and multiplex system under realistic workload conditions, and (3) the ability of the system to perform weapon delivery and navigation functions.

The DAIS demonstrations were performed in the Integrated Test Bed facility. The purpose of this facility is to test DAIS mission software and core element hardware under real-time conditions. The facility provides a real-time simulation of a military aircraft performing an operational mission. The simulation generates the interface signals so that the DAIS equipment and mission software is subjected to a data signal environment which is nearly identical to actual flight.

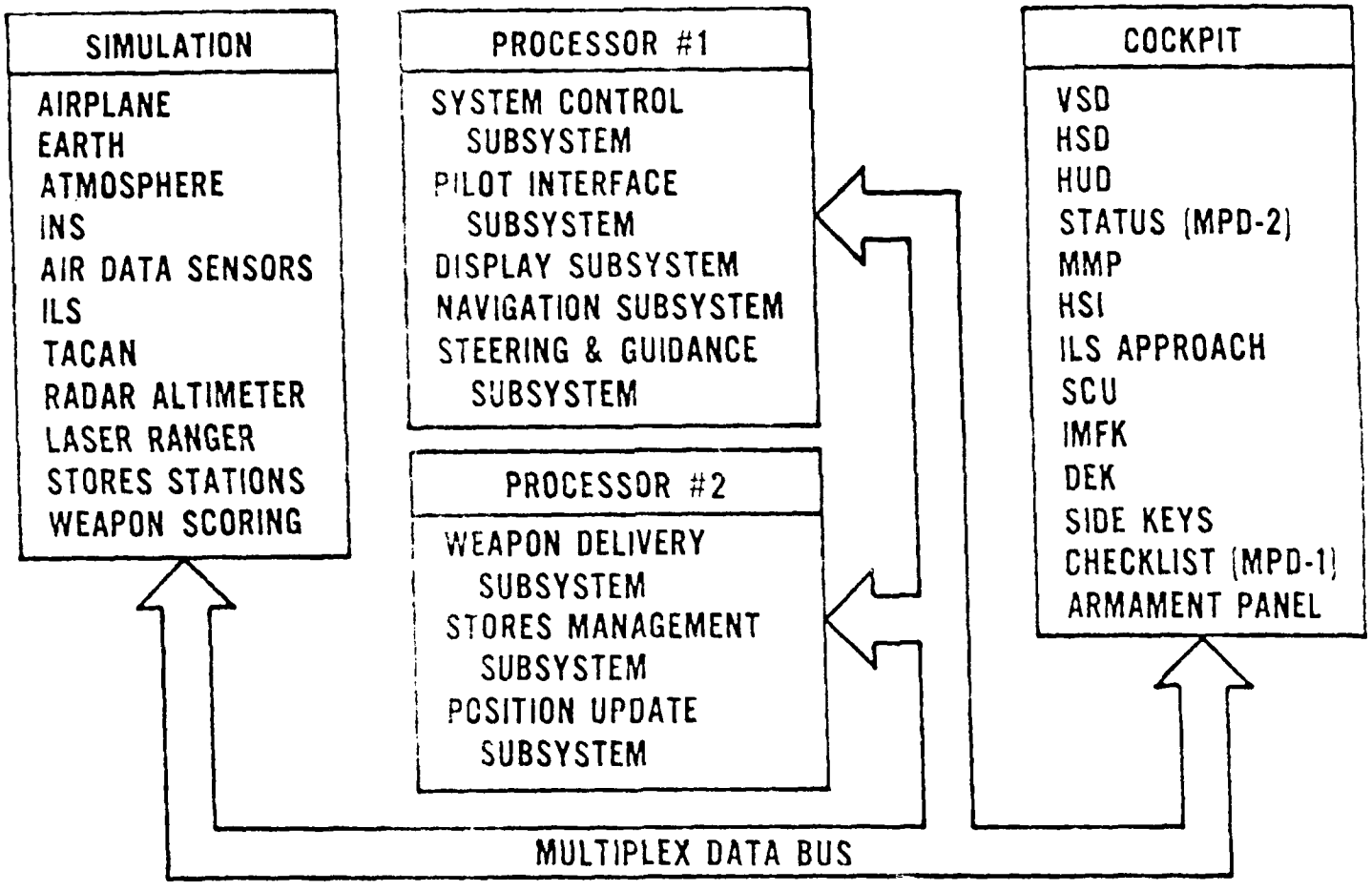
Simulation software was developed to provide real-time simulation of a military aircraft in an operational environment including the aircraft dynamics, the aircraft sensors and weapon targets. The simulation is driven from the cockpit by an operator acting as a "pilot". This simulated cockpit is equipped with the DAIS controls and displays so that the various modes of a mission may be "flown" by a "pilot" with an out-the-window background scene.

Four mission demonstrations were originally planned to satisfy the overall DAIS objectives. However, as the DAIS program progressed the final two demonstrations were deleted, and the remaining demonstrations were redesigned and redirected to satisfy the program objectives and validate new technological capabilities (MIL-STD-1553B multiplex architecture, MIL-STD-1750 processor instruction set architecture, and MIL-STD-1589B (JOVIAL).

10.1 Baseline Demonstration (Mission α)

The Mission α demonstration was successfully performed during September 1978 to demonstrate the DAIS system. This demonstration was the first of the four originally scheduled Close Air Support (CAS) demonstrations. This initial demonstration included functional capabilities such as: Inertial/Baro-Damped Navigation; Command Navigation, TACAN and ILS Steering, MK82 Weapon Delivery (with weapon scoring); Stores Management; and PREFLIGHT, TAKEOFF/CLIMB, CRUISE and APPROACH/LAND checklists.

The simulation was flown from the DAIS cockpit using active DAIS Controls/Displays and simulation models which reside on the DECsystem-10. No real sensors were used in the demonstration. The avionic system consisted of a two-processor DAIS configuration as shown in Figure 56 and Table 27. All communication between the DAIS processors/Bus Control Interface Units, the DAIS cockpit and the simulation models was accomplished over a dual redundant MIL-STD-1553A multiplex bus using DAIS/1553A system protocol. A DAIS remote terminal was used to interface the DAIS cockpit electronics to the dual redundant 1553A multiplex bus.



RE 56. MISSION CONFIGURATION

TABLE 27. MISSION α CONFIGURATION

Weather:	Night - Clear (VFR)
Target:	Fixed Ground Target
Weapons:	MK-82 LDGP Bombs
Threats:	None
Simulated Sensors:	INS Laser Ranger Air Data Sensors Radar Altimiter ILS TACAN
Core Element Hardware:	DAIS Processors (2): Master & Remote #1 BCIUs (2) RTs (2) Controls and Displays RT (1) — C&D Mass Memory VSD — SCU HSD — HUD MPD-1 — MPDG (1) MPD-2 — DSMU IMFK — AP DEK — MMP
Support Facility:	ITB Functional Diagram as shown in DA 100 102-1
Functions:	Navigation - Inertial/Baro-Damped Steering - Command, TACAN, ILS Navigation Update - Flyover, HUD/Laser Ranger, FLIR/Laser Ranger Acquisition/Cueing - Pilot/HUD, Pilot/FLIR Target or (OAP) Fix - HUD/Laser Ranger, FLIR/Laser Ranger Weapon Delivery - CCIP/Auto, CCIP/Manual Stores Management Communications - UHF Checklist

10.2 Upgraded Mission Demonstration (Mission β)

Using the Mission α demonstration as a baseline for development, several features were added which led to the Mission β demonstration. The Mission β demonstration was the second and final demonstration for the DAIS program. It was actually performed four times: first, to demonstrate the functional enhancements made to the Mission α demonstration baseline; second, conversions were required to accommodate the MIL-STD-1553B Multiplex System; third, the software was re-targeted to run on the AN/AYK-15A (MIL-STD-1750) processor; and fourth, the software was converted from JOVIAL (J73I) to JOVIAL (MIL-STD-1589B)).

The Mission β demonstration included functional capabilities such as: Command Navigation, Command Heading, Command Altitude, Command Track, TACAN, and ILS Steering; Dead Reckoning, and TACAN Area Navigation; MK82 Weapon Delivery (with scoring); Stores Management; and PREFLIGHT, TAKEOFF/CLIMB, CRUISE, and APPROACH/LAND checklists.

As with the Mission α, Mission β demonstration was "flown" from the DAIS cockpit using active DAIS Controls/Displays and the simulation models which reside on the DECsystem-10. No real sensors were used in the demonstration. The avionic suite for Mission β consisted of a three-processor DAIS configuration as shown in Figure 57 and Table 28. All communication between the DAIS processors, the DAIS cockpit and the simulation models was accomplished over a dual redundant MIL-STD-1553B multiplex bus, using DAIS/1553b system protocol. A DAIS remote terminal was used to interface the DAIS cockpit electronics to the dual redundant 1553B multiplex bus. Although each of the Mission β demonstrations was conducted on the basis described above, each was somewhat different in the technology used. Table 29 indicates the "user-transparent" differences in the three Mission β demonstrations.

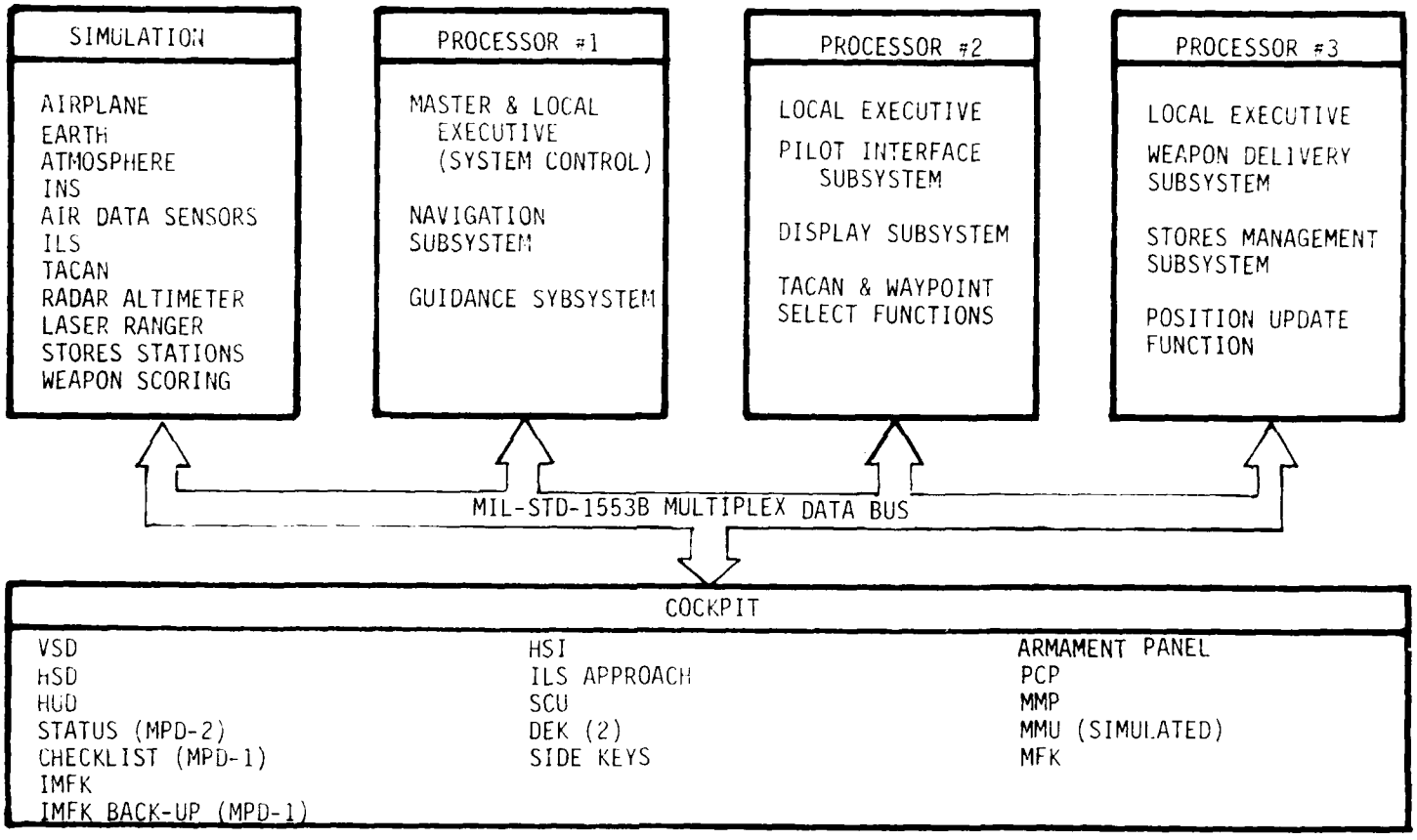


FIGURE 57. MISSION β CONFIGURATION

TABLE 28. [illegible] CONFIGURATION

Weather:	Night - Clear (VFR)
Target:	Fixed Ground Target
Weapons:	MK-82 LDGP Bombs
Threats:	None
Simulated Sensors:	INS (SKN2416) Laser Ranger Air Data Sensors Radar Altimeter (APN-141) ILS (ARN-58A) TACAN (ARN-118)
Core Element Hardware:	DAIS Processors (3): Master, Remote #1 and Remote #2 BCIU/BCI (3) RTs Simulated Mass Memory Controls and Displays RT (1) VSD — SCU HSD — HUD MPD-1 — PCP MPD-2 — MPDG (2) IMFK — DSMU DEK (2) — AP MMP — MFD MMU (1)
Support Facility:	Per ITB Functional Block Diagram DA 100 102-1 Per STS Functional Block Diagram DA [illegible]

Software:

- DAIS Executive
- DAIS Mission Software (Application)
- SSDF (URT)/Simulated Mass Memory
- MPDG1
- MPDG2
- Simulated Models
- E&S

TABLE 28. MISSION B CONFIGURATION (Con't)

Functions:	Navigation - Inertial/Baro-Damped TACAN Area Navigation Dead Reckoning
	Steering - Command NAV Command Track Command Heading Command Altitude TACAN ILS
	Navigation Update - Flyover HUD/Laser Ranger FLIR/Laser Ranger
	Acquisition/Cueing - Pilot/HUD Pilot/FLIR
	Target or (GAP) Fix - HUD/Laser Ranger FLIR/Laser Ranger
	Weapon Delivery - CCIP/Auto CCIP/Manual
	Stores Management
	Communications - UHF
	Checklist
	Startup/Restart
	Systems

TABLE 29. MISSION DIFFERENCES

FEATURES	MISSION				
	A	A_1	B	B_1	B_2
Multiplex System	MIL-STD-1553A	MIL-STD-1553A	MIL-STD-1553B	MIL-STD-1553B	MIL-STD-1553B
Avionics Processor	AN/AYK-15	AN/AYK-15	AN/AYK-15	AN/AYK-15A (MIL-STD-1750)	AN/AYK-15A (MIL-STD-1750)
Application Software Language	JOVIAL J73I	JOVIAL J73I	JOVIAL J73I	JOVIAL J73I	JOVIAL J73 (MIL-STD-1589)
Application Software Load Method	From DECsystem-2 or Using Bootstrap Loader from PDP-11	Bootstrap Loader PDP-11	From Simulated Mass Memory on PDP-11	From Simulated Mass Memory on PDP-11	From Simulated Mass Memory PDP-11
Processor Startup Method	Manual	[illegible]	Processor Control Panel (PCP) in Cockpit	PCP in Cockpit	PCP in Cockpit

AD-A107 906 TRW DEFENSE AND SPACE SYSTEMS GROUP REDONDO BEACH CA F/G 9/2
DIGITAL AVIONICS INFORMATION SYSTEM (DAIS): DEVELOPMENT AND DEM--ETC(U)
SEP 81 M J COOK, R C MASON, J L STAUTBERG F33615-78-C-1502
UNCLASSIFIED AFWAL-TR-81-1165 NL

3 of 3
AD
A107906

END
DATE FILMED
-82
DTIC

11.0 TECHNICAL SUPPORT

Throughout the DAIS program a high degree of technical interchange with other programs and other Air Force organizations was maintained. The most significant of these was the interaction/contribution DAIS made to the development of MIL-STD-1553B, MIL-STD-1750A and MIL-STD-1589B. Other technical support supplied by DAIS is identified in the following paragraphs.

11.1 Hot Bench Development for PAE

DAIS supported the development of the Precision Attack Enhancement Program at Martin-Marietta Corporation, providing numerous software packages including Diagnostics software and URT software for checkout and use on the PAE hot bench. DAIS also assisted in the installation of several versions of the Executive and supported investigation of problems related to the interface of these software packages and hardware at the PAE facility.

11.2 Fault Tolerant Assessment of DAIS

This study was conducted under Air Force Contract F33615-77-C-1232. DAIS provided technical consultation in the areas of reconfiguration, monitor processor plans and terminal failure/recovery operations. Documentation of the System Control Procedures and core element hardware specifications were provided.

11.3 Advanced Avionics Systems for Multi-Mission Applications (AASMA)

This study/implementation effort was performed under Air Force Contract F33615-77-C-1252. DAIS provided documentation of the System Control Procedures and Executive software as well as consultation on technical issues. A version of the DAIS executive was used as the starting point for the development of the Single Processor Synchronous Executive (SPSE). The contractor reported that such a development/implementation would normally have been allocated 52 weeks but was successfully accomplished in 3 weeks.

11.4 Fiber Optics Bus Receiver Requirement

This study/implementation was performed under Air Force Contract F33615-78-C-1561. DAIS supplied technical documentation and consultation on multiplex bus and RT operations and supported prototype testing in the ITB.

11.5 Remote Link Unit Design

This study/implementation was performed under Air Force Contract F33615-78-C-1634. DAIS supplied documentation and consultation on the System Control Procedures and multiplex system operation and supported prototype testing in the ITB.

11.6 Fault Tolerant Computer Network Study

This study is being conducted under Air Force Contract F33615-79-C-1180. DAIS has supplied documentation and consultation on the startup/loader operation, reconfiguration and difficulties with the monitor processor concept.

11.7 Mission Management Software for the KC-135 Avionics Modernization Hot Bench

This design/development is being conducted under Air Force Contract F33615-80-C-0274. DAIS supplied technical documentation and consultation on the system control procedures, Executive software, models, simulation, and other support software. A version of the DAIS executive was supplied as the starting point for this development.

12.0 CONCLUSIONS/RECOMMENDATIONS

The DAIS program successfully demonstrated the concept of a multiprocessor avionics system configuration in which standard, general purpose, digital processors, programmed in a standard high order language, communicate with each other and other avionics system elements via a standard, dual redundant, multiplex data bus.

More than just a concept, the DAIS integrated test bed represents an actual implementation of the following military standards:

- MIL-STD-1553B
- MIL-STD-1750
- MIL-STD-1589B

In addition DAIS represents a comprehensive, fully tested, mature baseline for the potential development of new standards in the areas of:

- Executive Software
- Executive to Applications Software Interface
- System Control Procedures

The DAIS architecture is a proven technology which can and should be applied to real aircraft and missions. It lacks only demonstration in a real flight environment (vibration and acoustics, EMI, etc.) The Air Force should vigorously pursue the infusion of this technology into the fleet.

In addition to the above overall conclusions/recommendations, a number of specific items are covered in the following subsections.

12.1 Core Element Hardware

The AN/AYK-15A processors represent an advance in avionics technology. They demonstrate the feasibility of embedding a bus interface/controller in the processor box and of implementing the MIL-STD-1750 instruction set architecture. Of major importance was the successful dual source procurement and subsequent compatibility demonstration. There is little doubt the AN/AYK-15A processors will find widespread application.

The Air Force should continue to track any problems or variances in operation in these units and seek resolution of any which threaten compatibility The specification SA 421 205-1 should be actively maintained and revised as necessary.

Evolutionary enhancements to the AN/AYK-15A are in effect being scoped by the MIL-STD-1750A working groups. The area that should receive first attention is the extended memory capability. Other areas, not covered by the standard, which should be investigated as an extension of the AN/AYK-15A technology are an increased intelligence in the controller to handle error and exception processing and the expansion to multiple buses to support future hierarchical bus architectures and the like.

The DAIS BCIU proved to be a useful step in the development of the AN/AYK-15A technology, but these units have served their purpose and should not

be developed further. In combination with AN/AYK-14 processors [illegible] can continue to perform useful laboratory functions (e.g. [illegible] for example).

The DAIS RTs demonstrated that a single unit could interface a large variety of subsystems and signal types. However, the clear trend in avionics systems is toward embedded bus interfaces and the DAIS RTs cannot be expected to find much applicability. The one potential application would seem to be in a "clustering" architecture in which RTs would be strategically placed in the aircraft geometry and interface all equipments in that vicinity. Apart from this, the RT's should be viewed as having served a useful role in development of the multiplex technology and not pursued further.

The MIL-STD-1553B multiplex was demonstrated by DAIS to be an effective data communications technology. Evolution of this standard is to be expected and the DAIS experience should be brought to bear on that process. As identified in the conversion discussion, some shortcomings in the standard are evident (for example, lack of a mechanism to clear the vector word). At the same time it should be recognized that when MIL-STD-1553B deleted the mode code "interrogate module error register" DAIS resorted to local error handling in the RT with the net result of better error recovery and reduced executive software. The lesson learned is that when an avionics system designer's favorite technique seems incompatible with the standard it may well be that the standard supplies or suggests a better approach.

The DAIS Controls and Displays proved to be an effective, flexible, pilot interface. The advantages of multiple purpose displays are evident and there is no doubt they will persist for generations of avionics systems to come. It is significant that in formatting displays DAIS found it necessary to generate graphic representations of the HSI, the FPM, and other symbology. The human factors considerations must not be overlooked, and the graphic representation of earlier generation equipment must be viewed as a useful general purpose technique.

The DAIS system mass memory was actually a simulation of a bubble memory device. The actual bubble memory device did not reach acceptable [illegible] status to be integrated into the IFTS. The feasibility of implementing a memory interfaced via the multiplex bus was demonstrated but neither the mass memory record formats nor the interface protocol bear any essential relation to the bubble memory technology. The interface protocol was driven by requirements of compatibility with other Air Force organizations. The system mass memory should be the subject of further investigation. The question of interfacing via the bus or directly to processors should be evaluated in light of advances in memory technology (including expanded processor memory) and the Air Force should reassess the compatibility requirements on the interface protocol. A record-(rather than word-)addressable interface may be more applicable and a re-read function would be useful.

12.2 DAIS Software

The DAIS Master and Local Executives form a powerful, general purpose software package that can be the basis of future avionics systems. The use of the executive for other applications has already been demonstrated. The DAIS executive was the baseline for the Single Processor Synchronous Executive (SPSE)

developed in the AASMA program, and the DAIS executive is the baseline for the current KC-135 hot bench development program. In the former case, the contractor reported the successful generation of an operable executive in three weeks time whereas normally they would have allocated a full year to achieve this important first milestone.

The Air Force should continue to supply the DAIS executive as the baseline for new developments and should support those efforts with strong technical assistance. Moreover the Air Force should follow up those efforts, soliciting feedback on problem areas and specific accomplishments.

The initial implementation of the DAIS executive was excessively large and inefficient. Improvements have been made continuously as the system matured from one demonstration to the next. The combined effect of better software structuring, the removal of the memory boundary constraints in the AN/AYK-15A processor, and the exploitation of the single word instruction format of MIL-STD-1750 by the J73 compiler resulted in an overall reduction of approximately 30%. One specific function, minor cycle setup, was initially a very complex function which was measured to be responsible for 25% overhead. Restructuring bus lists and reworking this function reduced that figure to less than 5%.

This progress in continuing. The final executive for the beta demonstration was about 4K words each for the Master and Local. With technical advice from the DAIS program, the KC-135 hot bench development has brought these figures down to about 3K and 2K respectively while at the same time approaching overhead figures worthy of a well organized assembly language executive. The lesson learned is that oversized, inefficient software is a consequence of specific implementation techniques and not due to the use of a higher order language. An efficient, reasonably sized executive can be written in HOL.

Two implementation approaches were identified as particularly harmful to the DAIS executive. One of these is giving strict, pre-emptive priority to asynchronous bus operation. The other was the adoption of a virtual memory approach for the software architecture. The asynchronous message handling is discussed more fully below, but briefly, the problem is that suspending and restoring bus operation is a very expensive process both in the software required to accomplish it and in wasted processor time. The virtual memory approach is a useful concept for software development, but it should not be carried over into the operational system. The partitioning of related tasks to different processors results in additional bus traffic and even if it is not done, just the potential of it places unnecessary constraints on the executive software.

In practice, task partitioning is along functional lines, keeping related tasks resident in the same processor to minimize interprocessor communication. The executive should be allowed to make use of knowledge of the system partitioning. One example of this is that upon a mission mode change the Master Sequencer signals all events individually as if the tasks were resident in the same memory. The event handling software must then determine what tasks are not resident and generate requests for asynchronous messages (one per event) to communicate the signals. Without the virtual memory constaint, the Master processor would simply signal the mode change to the remote processors. As noted elsewhere, the virtual memory approach also inhibits reconfiguration.

PALEFAC was originally intended to automatically perform the system partitioning based on some optimizing algorithm to strike a balance between processor loading and generated bus traffic. This was never accomplished due primarily to the fact that the actual partitioning is driven by a number of external factors not easily encoded to an algorithm. PALEFAC is nonetheless a useful support tool for building executive tables. The Air Force should maintain this tool and consider upgrading it to format and output some of the considerable data it has on a system/mission/configuration. Without much difficulty PALEFAC could be made to generate a model of expected system performance including a timeline of bus loading across minor cycles, peak processor loads, potential system bottlenecks, etc.

The DAIS startup/loader was fully demonstrated to accomplish all variations of startup, restart, reload, and transient recovery for all combinations of processor configurations. It is a complete and general purpose capability which is wholly independent of the DAIS executive software. The Air Force should make this technology available to other programs and provide technical support to adapt it to applications implementing a different mass memory technology or a different processor control panel approach. Adaptation will also be required if the startup/loader is to be applied to configurations of more than four processors. None of these adaptations are significant and apart from them the startup/loader is capable of handling any configuration of processors interconnected via a MIL-STD-1553B multiplex data bus. (Not even processor compatibility is required; only the standard bus interface.)

12.3 DAIS Support Hardware

The DAIS complement of support hardware proved to be an effective set of tools for the checkout/integration of the DAIS core elements, but for the most part these are specific to the ITB configuration. They should be maintained and assessed for applicability to future laboratory programs. No efforts to disperse this technology to other programs is recommended.

12.4 DAIS Support Software

The extensive repertoire of support software developed by the DAIS program should be conscientiously maintained and made available to other Air Force programs. To a large extent this has already been accomplished, but effort should be expended to sustain a widespread awareness of the capability and availability of the DAIS support software. A presentation or two at appropriate national conferences would seem to be in order.

In particular, the MIL-STD-1750A acceptance test program should find wide applicability. The 1750A ATP extensively tests the instruction set and many of the available options, thus providing a useful tool for verifying simulator operation or detection of hardware errors. Possible improvements for the ATP could include expanding the testing field to cover all options not currently tested and combining various instructions to verify actual programming situations.

In contrast, the performance monitor and control (PMC) software, upon careful assessment by an evaluation team, proved unwieldy and unstable.

The PMC software evaluation team advised that software not be changed, but since it had a tendency to instability, they recommended the software not be used for a baseline for future software.

12.5 DAIS System Control Procedures

The DAIS System Control Procedures as documented in MA 221 200-1 and SA 221 200 proved to be an effective vehicle for identifying and resolving system-level issues. They not only present the overall system operation (hardware/software/procedural) but also illuminate the underlying rationale. MA 221 200-1 documents the final configuration of the ITB while SA 221 200 is in essence a system designer's guidebook for the application of the DAIS technology. The Air Force should give wide distribution to these documents to aid in the infusion of the DAIS technology to other programs.

Several System Control Procedures topics deserve separate discussion, including synchronous operations, asynchronous operations, error/failure handling and the Monitor/Backup concept.

DAIS demonstrated that effective synchronous communications could be accomplished based on the period and phase concept. Further, by adopting the constraint that periods should be a power of two and adopting the convention of transmitting largest period data first, DAIS was able to define a static bus list requiring only that the BCM be started at the correct point in the list. Thus even though different sets of synchronous bus operations are performed each minor cycle, no dynamic bus list manipulation is required.

DAIS demonstrated that asynchronous message operations are strongly implementation-dependent. DAIS implemented a strict priority for asynchronous operations requiring that synchronous operations be suspended upon the recognition of an asynchronous request and that the complete decoding of the request and transfer of data be accomplished before resuming the synchronous operation. This proved to be very expensive in terms of the amount and complexity of software required and it resulted in processor throughput degradation. It also was the source of several difficult-to-isolate problems of a hardware/software interaction, timing dependent, variety. While alternate procedures were defined, they were not implemented in DAIS. They are being pursued in the KC-135 hot bench development.

Another aspect of the asynchronous implementation was that a remote processor advanced its transmission queue when it finished sending an asynchronous message. This necessitated a special re-transmission procedure when errors occurred. Interestingly enough the conversion to MIL-STD-1553B aided the solution to this problem. The deletion of the mode code to "reset status code field" forced a rework of this area and the revised control procedures withheld the transmission queue update until the reception of a "synchronize mode code." The net result of this modification was a reduction of software in both the Master and local executive. The important corollary of this result is that the software expediency of performing buffer management in the transmitter at the completion of message transfer actually generates additional software requirements in both the transmitter and master controller.

The DAIS experience therefore is that asynchronous bus operations can have a severe impact on size and performance. This, however, is not a property

of asynchronous messages but rather of the implementation technique. [illegible] cient, fully deterministic, asynchronous implementation is possible.

Message error handling and retry is an important related area that the System Control Procedures must deal with. At one point in the DAIS program, when the control procedures applied error considerations to all message types and combinations of transmitters and receivers, it became necessary to define seven separate classes of message retry. By the time of the final DAIS demonstration this had been reduced to two types of retry plus special handling for the system mass memory.

The careful retry procedure was introduced to handle the situation of a device of subsystem which would function improperly if sent data twice. An early DAIS configuration had an inertial measurement unit (IMU) which operated in this fashion. The procedure to handle this type of device is to perform a "last command" mode code operation to first confirm that the device received the command work and then a "transmit status" to confirm the device did indeed detect an error prior to repeating the message. And, of course, proper operation requires this be performed immediately, before resuming normal bus operations. MIL-STD-1553B removed the need for a separate transmit status operation, but the last command sequence remains an awkward procedure with undesirable software and performance impacts. The Air Force should eliminate this requirement. Subsystem procurement specifications should include the requirement that the subsystem shall continue to function properly when intermittently sent repeated data. The subsystem should similarly be tolerant of occasional brief lapses in the data flow. The Air Force should place the same requirements on application software and should include data repeat and data lapse tests in acceptance test programs for application software and subsystems.

Another type of message that represents a severe difficulty for the system control procedures is the "critical message." This is a message which must succeed for proper operation. If that is indeed the requirement, the control procedures have no choice except to terminate in endless retries, device failure, or system failure. The Air Force should endeavor to eliminate "critical messages" whenever possible.

The Monitor/Backup operation is an area in which DAIS did not accomplish its original objective. The startup/loader provides a complete reconfiguration capability and (for certain systems and missions) this is sufficient to accomplish the Monitor/Backup function, but DAIS did not succeed in solving the general case problem. The Monitor/Backup concept is one in which a spare processor, the Monitor, is capable of detecting a failure of the Master and then of assuming the functions of the Master to maintain system operation. DAIS defined and demonstrated the mechanism for the monitor to quickly and reliably detect a master failure and to take control of the system. The stumbling block is in the definition of the contents of the monitor. The nominal definition of a duplicate of the master leaves unaddressed the question of how the monitor obtains adequate data on the state of the system to actually perform the master function. The approach of configuring the Monitor as a remote during normal operations in order that the master can send him the required data has several flaws. It adds software requirements on the master, negates the concept of the duplicate master and creates a spectrum of failure cases for the monitor to deal with. This approach must also answer the difficult philosophical and practical question of

how can the monitor reliably begin system operation based on data received from a failing master.

Another aspect of the Monitor/Backup problem is the monitor assumption of the role of a remote processor. A reload is required in this case and it is desirable for the balance of the system to maintain a degraded mode of operation while this is taking place. The virtual memory architecture of the software, however, permits each remote processor to depend on every other for correct, continued operation. In principle, then, the degraded mode of operation is not possible. The system operation would have to stop while the monitor was reloaded and reinitialized to perform the remote function. This approach is within the scope of the current DAIS technology but it was not implemented since the startup/loader already accomplishes the equivalent end result and is a more general solution to the restart problem.

Thus, DAIS has solved the error detection/rapid failure portion of the Monitor/Backup problem, but lacks a definition of the Monitor content, the protocol to update it with current data, and the ability to continue operation following a remote failure. The Air Force should continue to address the difficult question of the Monitor/Backup technology. The most promising avenues appear to be in first controlling the software partioning to remove the interdependency of remote processors and secondly of expanding the bus controller capability to include an operational mode which is truly a monitor, i.e. one in which the required system state data may be obtained without perturbing the normal system operation (in which case the duplicate master approach is feasible).

These issues must be addressed in future programs, but always in the context of the system control procedures so that all pertinent considerations can be brought to bear. It is recommended the Air Force give a high priority to this area of advanced avionics technology.


FILME
8