A SURVEY OF NAVY TACTICAL COMPUTER APPLICATIONS AND EXECUTIVES

Report of a Study by:

Dorian Punj
Prof. Stuart E. Madnick
John D. DeTreville

CISR REPORT-19

October, 1975

Center for Information Systems Research
Massachusetts Institute of Technology
Alfred P. Sloan School of Management
50 Memorial Drive
Cambridge, Massachusetts, 02139
617 253-1000

Contract No. N00039-75-C-0312

FOS No. 010-D

Deliverable No. A01

Facilities Orientation Report

Volume I

A SURVEY OF NAVY TACTICAL COMPUTER APPLICATIONS AND EXECUTIVES

Report of a Study by:

Dorian Punj
Prof. Stuart E. Madnick
John D. DeTreville

CISR REPORT-19

October, 1975

Principle Investigators:

Professor John J. Donovan
Professor Stuart E. Madnick

Prepared for:

Naval Electronics Laboratory Center
271 Catalina Boulevard
San Diego, California 92152

Sponsored by:

Naval Air Systems Command
Washington, D.C. 20360

Naval Electronics Systems Command
Washington, D.C. 20360

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER FOS 010-D	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Facilities Orientation Report, Volume I A SURVEY OF NAVAL TACTICAL COMPUTER APPLICATIONS AND EXECUTIVES		5. TYPE OF REPORT & PERIOD COVERED 6. PERFORMING ORG. REPORT NUMBER FOS 010-D
7. AUTHOR(s) Mr. Dorian Punj Professor Stuart E. Madnick Mr. John D. DeTreville		8. CONTRACT OR GRANT NUMBER(s) N00039-75-C-0312
9. PERFORMING ORGANIZATION NAME AND ADDRESS M.I.T., Sloan School of Management E40-365 Cambridge, MA 02139		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Air Systems Command Washington, D.C. 20360		12. REPORT DATE OCT 1975 13. NUMBER OF PAGES 320
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Naval Electronics Laboratory Center 271 Catalina Boulevard San Diego, CA 92152		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A

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)

Computer executives, computer operating systems, tactical computer systems, tactical computer applications

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

This document is the final report of a study of current Navy operating system milieu. It includes information about Naval tactical computer applications, and in particular, information on the major executives used in these applications. Analysis and evaluations of the various operating systems are presented together with implications that these systems have on the design of a family of operating systems for future Navy tactical systems.

DD 1

S/N 0102-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

NAVAL ELECTRONICS LABORATORY CENTER
271 CATALINA BOULEVARD
SAN DIEGO, CALIFORNIA 92152
714-225-6011
AUTOVON 933-1011

IN REPLY REFER TO:

To the reader --

A word of explanation.... When faced with the requirement to design a tactical operating system for a proposed advanced architecture machine, the Navy had two basic alternatives for selecting designers. We could either select a group of people well-qualified in potential application areas and give them a background in operating system theory, or we could select a group of people well-qualified in operating system theory and orient them to potential application areas. The latter choice was made, and the MIT Sloan School was selected, with technical leadership provided by Dr. Stuart Madnick and Dr. John Donovan.

This report presents the information that MIT found as a result of their investigation into current Navy tactical operating systems milieu. Due to the constraints of time and money, it was impossible to include all tactical systems of interest; it can only be hoped that at least one representative system (i.e., one illustrating the requirements and constraints) was examined in the most demanding tactical operational areas.

In the preparation of this report, the choice was made for comprehensiveness over polish; that is, the effort that might have been spent on polishing the document, improving the English, or verifying details of oral folklore was instead used to extend the coverage or to quantify better the lessons learned. Thus, it is almost certain that errors of commission and omission have occured. The sole responsibility for this lies with the Contract Monitor.

We hope that some other agency at a later time will want to extend this work. As an aid to future work, we strongly encourage you to send us any comments you may have. The comments need not be formal -- send them handwritten, or send a copy of our pages with marginal notations, or whatever is convenient. Information and recommendations about systems that place unusual requirements or constraints on an operating system are particularly welcome.

A note to other agencies.... If you ever use this report for background material, request that a set of comments be prepared and send us a copy.

A note to contractors.... If you are ever requested to use this report for background material, insist on preparing a set of comments and send us a copy.

Comments and other related correspondence should be sent via U.S.Mail to:

Naval Electronics Laboratory Center
271 Catalina Blvd
San Diego, CA 92152

Attn: Code 5200, FOS Contract Monitor

Mail may also be sent via the ARPAnet to NOEL@ISID.

Credit where credit is due.... This work is sponsored by the Naval Air Systems Command (ADPO-34, Mr. Bernard Zempolich, Project Manager) in connection with the development of an advanced architecture tactical computer which is to be available in the fleet in the 1980's. It was their courageous decision to consider the software implications in parallel with the development of the hardware. Close co-operation has been received from the Naval Electronics Systems Command (ELEX-330, Mr. Robert Kahane and Mr. John Machado) which is responsible for Navy software research. Technical responsibility for the work has been at the Naval Electronics Laboratory Center(Code 5200, Mr. Russell Eyres, Division Head, and Mr. Warren Loper, Technical Manager).

J. Gregory Noel
FOS Contract Monitor
NELC Code 5200

FOREWORD

This study is the first step in the design process of a family of operating systems for future Navy tactical systems. It was begun in August, 1974 as deliverable #A01 to NELC, San Diego, under Contract No. N00039-75-PR3K137.

Contributions to this report were also made by the following people at the Center for Information Systems Research at M.I.T.'s Sloan School of Management: Leonard Goodman, Brad Albom, Yuval Gilbert, Adam Schneider, Dan DuBoff, Shing Chiu, and Mike Wilens.

We wish to express our appreciation for the help provided by many people associated with the AADC program in supplying documents and information about operating systems in use by the Navy, and for cooperating with us on our "trip visits" to various Navy installations.

PREFACE

The Center for Information Systems Research (CISR) is a research center that is located and managed in M.I.T.'s Sloan School of Management, and consists of a group of Management Information Systems specialists, including faculty members, full-time research staff, and part-time students. The Centers' general research thrust is to devise better means for designing, generating, and maintaining applications software, information systems and decision support systems.

Within the context of the FOS effort, CISR proposes to develop and test a set of techniques for designing, generating, and maintaining a modular family of tactical operating systems to last for twenty years or more.

The project has been divided into the following three phases:

Phase 1: A Facilities Orientation, including a field and literature study of existing military software facilities, development techniques, and operational environment.

Phase 2: Detailed design of system concepts and techniques, including the implementation and testing of a prototype system.

Phase 3: A Computer Program Design Specification which is adequate for the open competitive procurement of the entire family of operating systems.

The project is estimated to be completed by mid-1976.

To form a basis for our design work, we have analyzed the current Navy operating system milieu in detail in this report. We have gathered information about Naval tactical computer applications, and in particular, information on major executives used in these applications. FOS personnel have visited Naval installations and facilities (see Appendix A) to discuss current

as well as future computer applications with Navy computer personnel, for which we have prepared a number of trip reports. We have studied and analyzed a large number of documents concerned with Navy tactical executives. Moreover, we have attended a number of Navy-sponsored meetings concerned with the effect of related AADC efforts to the design of a family of operating systems (e.g., the design of the CS-4 programming language).

This document is the report of this study. It presents the information we have gathered and is intended to portray the current Navy operating system milieu in a meaningful way. Our analysis and evaluations of these operating systems is also presented together with any implications we feel that these systems have on the design of a family of operating systems for future Navy tactical systems.

The FOS group has made every attempt to ensure the accuracy of the information contained in this report. However, this report was prepared primarily as an orientation to Navy tactical systems for the team at M.I.T. designing a family of operating systems (F.O.S.). This report is by no means completely comprehensive and accurate. The contractors are primarily interested in Navy operating systems, and hence the descriptions generally discuss the operating system environment in greater depth than other seemingly relevant topics.

Most of the information in this report has been gathered from verbal communciations, group discussions, and a limited number of technical reports. In fact, there are a number of systems (e.g., SMACK/7) that we have heard of, but have not had the opportunity to study. Hence, the F.O.S. group at M.I.T. regrets any inadvertent inaccuracies or misinterpretations contained in the reports.

GUIDE TO THE REPORT

This report is divided into four parts. Part I should be read by anyone interested in gaining an overall picture of current Naval tactical applications, computer systems, peripheral equipment and operating systems. Part II will be of interest to those concerned with a summary and analysis of particular Navy executives. Part III discusses specific features of importance in Navy executives, and analyzes these executives in the light of these features. Part IV presents our conclusions.

Part I consists of Chapter 1 and discusses the general characteristics of Navy tactical applications. It presents a framework within which to analyze tactical computer systems and demonstrates the applicability of this framework in analyzing one example, specifically, a Navy fire control system.

Part II of the report consists of Chapters 2 - 6 and discusses each of the major operating systems in detail. These Chapters discuss the operational environment that the executives are used in, their general characteristics, as well as more detailed information such as the devices they interface with and other special features. Specifically, Chapter 2 discusses major executives commonly used in the Navy, Chapter 3 discusses executives used in shipboard applications, and Chapter 4 discusses executives used in airborne applications. Chapter 5 summarizes the Navy's communications system, and Chapter 6 presents current and proposed applications by the Marine Corps.

Part III of the report consists of Chapters 7 - 13, and presents a feature by feature analysis of the principal components of tactical executives. Schedulers, memory management schemes, message handling features and interrupt management are examples of these features that are grouped together for comparison and analysis.

Part IV of the report consists of Chapter 14 which summarizes our findings and presents recommendations for future operating systems.

TABLE OF CONTENTS

PART IV -- SUMMARY

PART I

GENERAL CHARACTERISTICS OF NAVY TACTICAL APPLICATIONS

CHAPTER 1

OVERVIEW OF NAVY TACTICAL COMPUTER SYSTEMS

1.0 Need for a Framework

A framework for viewing Navy tactical systems is essential if a meaningful analysis of tactical systems and applications is to be made. The use of computer systems in the Navy has grown in recent years, and, although the basic functional requirements of Navy tactical systems are not too dissimilar, the proliferation of computer hardware and software in the Navy is largely a result of the lack of such an overall framework.

In this chapter we propose a specific framework which we have found to be useful in the analysis of Navy tactical systems. We believe that this framework offers us a methodical means of isolating the key characteristics of the various functional capabilities that the Navy's tactical systems possess. We shall then use this framework to analyze the types of resources that are required in the different functional areas in tactical applications, demonstrate the feasibility of this framework by analyzing a specific Navy tactical application. In subsequent chapters we discuss the major tactical operating systems in existence today.

1.1 Differences Between Tactical and Non-Tactical Applications

Navy tactical computer applications have many characteristics that distinguish them from commercial applications. Because tactical systems must support a set of functions that differ markedly from commercial applications (except perhaps in the area of process

control), significant distinguishing features are found in the executives and operating systems, architectures, and capabilities of Navy tactical computers.

For example, because tactical computers are used for functions such as launching missiles and guiding aircraft, response must invariably occur in real time. Because they are often used in airborne and wartime applications, a high degree of reliability is required. Functions must be implemented with some form of backup and provisions must be made for "graceful" performance degradation. Because the computers are carried in ships and aircraft, packaging and weight contraints become more important. For example, disk drives on a ship must be able to withstand the roll of a ship, and computers in an airplane must be light weight and compact.

1.2 Framework of Functional Characteristics

The major functions that Navy tactical systems perform can be broken down into the major categories of (a) computational functions, (b) human interaction function, and (c) multi-computer coordination functions.

1.2.1 Computation Functions

The substantial computational or "number crunching" functions that these systems perform represent the vast analytic and data reduction functions that tactical systems must be capable of supporting. These computational functions fall into the following three classifications: data gathering, data reduction, and data analysis.

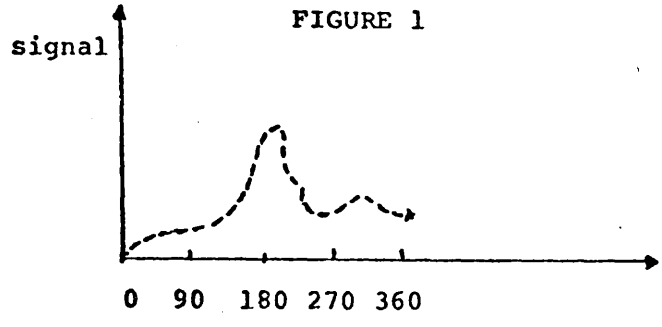
1.2.1.1 Data Gathering

Tactical systems invariably receive data from sensors that are unique to Navy applications. Radars performing the tracking function are not an uncommon type of sensor. Most data from radar is usually received by periodic data sampling. Here data is stored in tables in a prearranged fashion, and there are constant validation checks on the data for any deviation from the norm. The data is usually fed into a tracking routine for identifying radar responses, and tracking target paths (heights, distances, etc).

In addition to radars, many other types of data gathering sensors are prevalent in tactical systems. Sonar is used on many ships where the periodic input rate is in the 40 ms. range. Surface scanners are used on many ships with substantially faster data rates. Airborne sensors are also used for gathering navigation information on air speed and wing and stabilization controls. A substantial amount of data is also received from other computers as messages. This is discussed in the section entitled, "Inter-Computer Communication."

To summarize, data collection is a function that is found in all tactical applications. It is the means by which the system gathers information about its operational environment and formats it in a manner that is acceptable to the next phase of the computation function, data reduction.
This data gathering function can be described graphically as follows, by visualizing the signal received by the radar as it makes a 360° sweep.

FIGURE 1



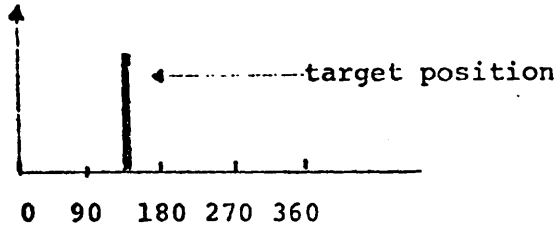
(The dotted lines are used to stress the fact that data is gathered at a rate of say, 32 times per second).

1.2.1.2 Data Reduction

Data reduction is that process by which data from sensors is interpreted and modified into a form useful for analysis. For example, when using periodic sampling techniques, the useful data may consist of the deviations or abnormalities from expectations of incoming signals.

Thus, in our example, the data reduction process would be responsible for determining the target position from the signal derived from the sampled data displayed in the previous figure. The process would deduce that the target is currently located at approximately 150°, which can be graphically represented as follows:

FIGURE 2



In our example, this data reduction process would determine target position at, say, 4 times per second.

1.2.1.3 Data Analysis

The process by which the system activates the necessary response to the data is called data analyses. This can range from responses to radar tracking information, automatic adjustments to speed and altitude in response to tracking information, as well as the in-flight guidance of missiles in response to updated data on the location of an enemy target.

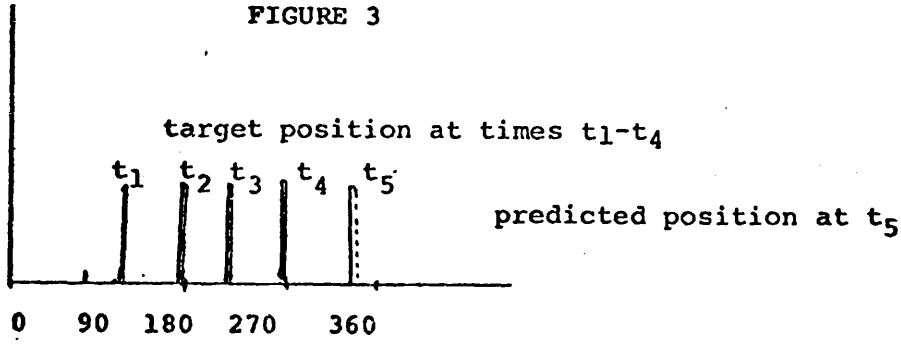
The analyses programs can be very complex. A tracking program for the E-2C early waring system uses 6K 32 bit words. A typical analysis program is the calculation of the "optimal gun solution," which screens radar information and then

computes the bullet trajectory as fast as possible. Other examples include the analytical functions provided by the Air Data Computer for the F-14 fighter. It receives data and then is automatically responsible for wing speed control, stabilization control, determining "true air speed," and indicating the angle of the plane to the air-stream.

A more complex example of this analytical function is the Identify Friend or Foe (IFF) function performed by the AEGIS system. The operational sequence starts with a search for a target (missile or aircraft) at different altitudes. If a contact is found, the IFF function is invoked. If the contact is determined to be a threat, the interception track is estimated and the interceptor is fired.

For example, the data analysis process could be responsible for receiving data from the data reduction process and determining the future position of the target. This would occur at a lower frequency of perhaps once per second and can be graphically described as follows:

FIGURE 3



1.2.2 Human Interaction

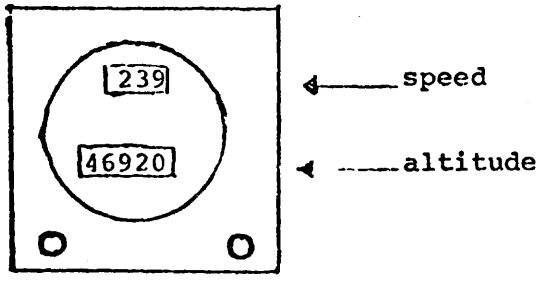
In addition to the computation and analytical functions, tactical systems must usually provide for a great deal of human interaction.

Almost all tactical systems interface with consoles and displays for human operators. The simplest example of this human interaction is in updating readouts such as the altitude or speed of a plane on a pilot's dashboard. (Figure 4)

More complex displays require a large amount of computation for graphical displays. These include the tracking of enemy targets on a radar screen relative to the current

FIGURE 4

Updating Readouts



position (see Figure 5) A more advanced example is the A6B system which is a weapons delivery platform(predicted to be in service through the mid-1980's). The digital display unit displays air speed, longitude, latitude, and altitude. The pilot navigates by attempting to keep a small square on the screen in the middle of the display.

FIGURE 5

Graphical Interaction

Some airborne systems carry even more sophisticated displays, combining navigational information (such as yaw, roll and pitch analyses, air speed, and altitude) with tactical information (informing the pilot of his distance and position relative to an enemy airplane).

An important facet of this human interaction function that must be supported by tactical systems is the option of human initiation of many tactical functions. This means that although a critical tactical function has been automated, the process can always be either initiated or stopped by a human being. This is as a result of Navy philosophy, which generally dictates that manual initiation and negation of all missile launching orders must be possible at all times.

In the previous section we discussed the Identify Friend or Foe process, whereby if an enemy target is identified, an interception track is estimated, and the interceptor is fired. During this process, no human interaction is required during the actual interception, but it is possible at all times.

Another typical example is the Air Data Computer System (ADC) used on the F-14 fighter. Although this is an elaborate automated system for wing sweep control, stabilization control,

determining true air speed, and indicating the angle of the plane to the air stream, it must be noted that if the computer system ever fails, the pilot always has recourse to using his manual controls.

1.2.3 Multi-Computer Coordination

Many tactical processes which must be coordinated run on separate computers. This is because different computers are often used for specialized functions. Hence there is the need for complex communication between systems, and most tactical systems communicate via one or more of the several sophisticated communications links, (see the example in Figure 6).

For example, many shipborne tactical systems communicate via NTDS, which is a command and control system that has been implemented on about sixty ships. NTDS is an automatic method of transferring all tactical data from one NTDS unit to another. Prior to NTDS all tactical information in a combat situation was transferred via voice. In addition to acting as a two-way link between aircraft and carrier, the NTDS computers interface with special purpose computers, displays, fire control systems and radar tracking systems. Other computer based communications systems in the Navy include CUDIXS, (a digital message handling system), and NAVMACS, which is a family of communications systems for message handling.

1.3 Example System

It would be useful to analyze the various features of a typical tactical application in the light of the model for tactical applications that we have presented.

The example we choose here is the Navy's TARTAR missile system which is a fire control system aboard a ship (DLGN-38) which is responsible for finding a target as well as firing a missile when a target comes within range. Computers are used to locate the position of a target and to control the guns.

The architecture of the TARTAR system is shown in Figure 1. It consists of a track radar, a search radar, a fire control computer, a signal digital convertor, and a missile launcher computer. The fire control computer used to be a UNIVAC 1219, but is now a UYK-7. The tracking system is an SPG-60 radar. A typical system can consist of 2 radars, 3 launchers, and 3 weapon direction systems. We will now analyze the functions and capabilities of the TARTAR system within the framework of our model of tactical systems.

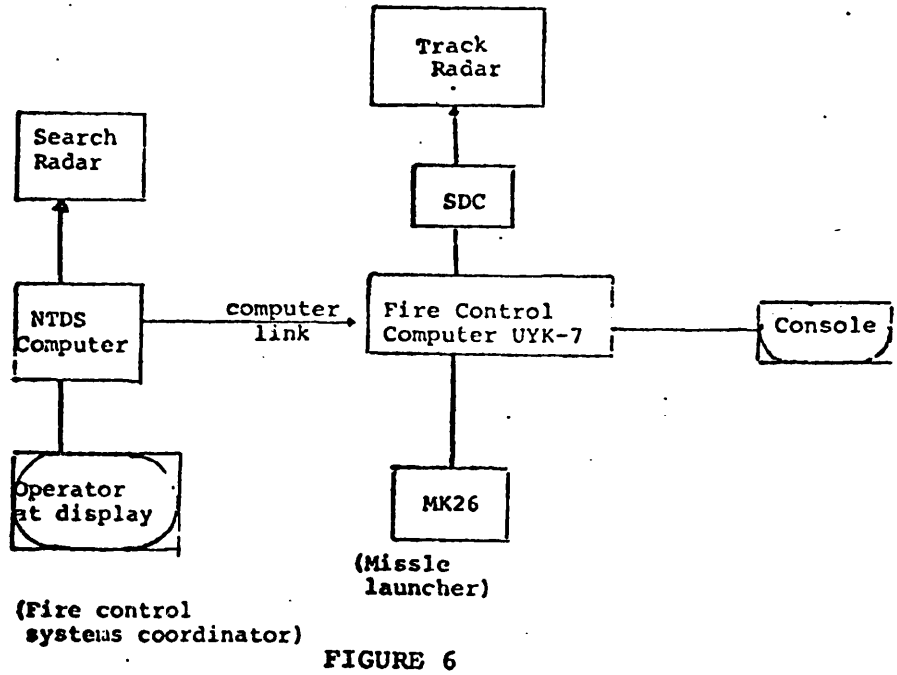


FIGURE 6

1.3.1 Computation Functions

The data gathering function is performed by the SPG-60 radar, which sends data to the fire control system. A common stumbling block in developing fire control systems is that computers are sequential, synchronous machines, whereas data from radars is statistical and random in nature. Sampling theory solves this problem for the TARTAR system. A bandwidth of 31.25 ms. (32 per second) is used to sample the data from the radars. Sometimes the sampling rates vary according to the type of data that is being sampled, but the sampling rates used in the TARTAR system are always either 4, 8, 16, or 32 per second.

The SDC or signal digital convertor is used to convert the signals from the radars into a form acceptable by a digital system. Without the SDC an analog system would have to be used.

The data reduction functions performed by the UYK-7 uses feedback loops to send orders to the missiles and launchers.

1.3.2 Human Interaction

Human interaction is performed by the Fire Control Systems Coordinator sitting at the UYK-7 console. This "FCSC" initiates all the automatic firing and controlling of the missiles. This is in accordance with Navy philosophy that a computer cannot make the decision whether or not to fire a missile. (It can, however, fire it at the optimum time once the decision to fire has been made).

1.3.3 Inter-Computer Communication

Inter-computer communication is shown by the many links to and from the main fire control computer (FCC). Within the system the FCC communicates with the missile launching computer (MK26). The FCC communicates with other ships and the "outside world" via the NTDS computer which communicates through standard data links.

1.3.4 TARTAR Executive

The executive that Raytheon built for the TARTAR system essentially consists of two executives--a foreground executive and a background executive. The foreground executive is an interrupt driven real-time executive with a predetermined periodic scheduling of tasks. The background executive basically manages non time-critical tasks (such as background test, maintenance, and performance evaluation routines), and schedules these tasks on a round robin basis.

Peripheral devices are handled in a way that is fairly typical of most tactical Navy systems. The routines to handle the I/O to any given device is the responsibility of the program that generates the data for that device. This procedure eliminates the necessity for storing I/O routines that may never be utilized.

1.4 Summary

In the following chapters we will be discussing the variety of executives used by the Navy to support a host of tactical applications. Where possible, we introduce the description of the operational environment that this executive is used in. In analyzing these environments it will be useful to

keep the framework we have presented in mind in order to obtain a better understanding for the functional capabilities that the executive must support.

PART II

MAJOR EXECUTIVES

USED IN THE NAVY

CHAPTER 2

MAJOR EXECUTIVES USED IN THE NAVY

2.0 Introduction

Of the vast number of executives used in the Navy, four are important enough to be discussed in detail in this chapter, either because they have become very popular throughout the Navy, or because they were developed with the aim of bringing more standardization to the Navy operating system milieu. The executives discussed here are:

a. The AEGIS Tactical Executive Program which is the standard executive for the AEGIS program. It has been the focus of much attention because it has been quite successful in AEGIS, and it has been modified to serve other applications.

b. SDEX/7 was developed for the UYK-7, and has been proposed as a candidate for a standard executive for the UYK-7.

c. SDEX/20 was developed as a highly flexible executive for the UYK-20 minicomputer.

d. COMMON PROGRAM (CP) is an AN/UYK-7 executive that is very widely used in the Navy. It is very popular for use in Fire Control and Command and Control systems.

2.1 ATEP

2.1.1 Overview of Operational Environment

RCA is under contract to the Navy to develop the AEGIS defensive missile system, which is intended to answer the Naval need to contend with airborne threats and advances in electronic warfare techniques in the 1970's and 1980's. The key components of AEGIS are phased array multifunction radars, defensive missiles, and multipurpose missile launchers, guidance illuminator radars, and computerized command and control.

AEGIS is integrated and controlled through three relatively autonomous multipurpose computer-centered groups: MFAR Segment, Weapon Direction System, and Command and Decision (C&D) segment. Operations of these groups are facilitated by a complement of AN/UYK-7 computers, AN/UYA-4 display consoles, and associated peripheral equipment.

An initial system has already been successfully tested on a test ship, U.S.S. Norton Sound. The systems' executive program is called ATEP (AEGIS Tactical Executive Program) and it controls the processing in each of the AN/UYK-7 computers. Each AN/UYK-7 computer is dedicated at the time of system initialization to a specific role, and contains the application computer programs responsible for carrying out an

assigned tactical mission. Each computer is linked to tactical mission equipment specific to that role, and is also linked to general purpose equipment such as disks, tape drives and operator consoles. The AN/UYK-7 computers also interface with each other where a tactical mission requires more than one computer, or where an interchange of information is necessary between computers assigned to different tasks.

The AEGIS operational sequence starts with a search for a target (missile or aircraft) at different altitudes. If a contact is found, an Identify Friend or Foe (IFF) function is performed. If the contact has been determined to be a threat, the interception track is estimated, and the interceptor is fired. (No human interaction is required during the actual interception, but it is possible at all times). Terminal guidance is performed by shining an illuminator on the target, hence reflecting energy from the target to the interceptor, which is then used to home in on the target.

The Operational Readiness Test System is a sophisticated system that is responsible for assuring that AEGIS is always maintained in a state of operational readiness. The system performs the following major functions and checks:

*Fault Detection Coverage

*Self Testing

*Reliability

*Failsafe

*Efficiency Improvement

*False Alarms

ORTS uses AN/UYK-7's and uses a large data base resident in the radar computers. ORTS is usually called automatically by the executive, but it can also be activated manually.

2.1.2 System Description

The AEGIS described in the previous section is a fast-reaction weapon system possessing a high degree of system reliability designed to counter the AAW threats of the 1970's and 1980's. AEGIS is to be integrated with other ship sensing and weapons systems consisting of the IFF systems, Navigation Systems, Gun Fire Control System, Electronic Warfare System, and Surface Search Radars. AEGIS will interface with these ship systems through the Command and Decision Control Segment (C&D) of the AEGIS Command and Control Group (C&CG). This interface is designed to provide digital data compatible with the C&D requirements. The AEGIS C&CG will provide the integrated system complex that controls and directs AEGIS actions. The C&CG also integrates AEGIS with other subsystems that comprise the ship combat system. Although in operation the C&CG must operate as an integrated unit, the group is divided into three distinct functional and equipment segments: Command and Decision (C&D); Weapon Direction System, and Multifunction Array Radar (MFAR). Each of these segments is computer-controlled, and ATEP controls the processing in each of the AN/UYK-7 computers.

2.1.3 The Hardware Environment

The complex of radar, missile, and display systems that make up the AEGIS Weapons System is controlled by AN/UYK-7 computers. Each of these computers contains tactical and fault analysis programs (referred to collectively as the "operational programs") that are centrally monitored and scheduled by the real-time executive, ATEP.

The UYK-7 is a 32 bit word machine with a cycle time of approximately 1-1/2 microseconds. It is a 256K machine, that is, expandable to 256K in 32K increments. It is a modular system and memory can be either local or shared (although some of the memory must always be local). The UYK-7 can also be configured with more than one processor.

2.1.3.1 Peripheral Equipment

ATEP controls the following peripheral equipment:

a) An RD-281 magnetic disk, which is a standard military disk, (similar to the single spindle IBM 2314). It has a storage capacity of 1.7 million 32-bit words contained on 20 disk surfaces each with 100 tracks of 836 words. Nominal rotation speed is 1,500 rpm and maximum access time is 315 milliseconds. External functions include seek, status, and various write and read commands.

b) A mass memory multiplexer which serves as the RD-281 disk storage interface for all AN/UYK-7 computers. It connects the RD-281 with the computer generating the highest priority service request, and provides all existing data interrupts and function word transmission functions between the RD-281 disk and the computer being serviced.

c) Modular magnetic tape sets, alphanumeric display consoles (including CRT's), paper tape readers and punches, and a high speed printer.

d) In addition, the following tactical equipment interfaces indirectly with ATEP:

1. AN/UYA-4 Data Display consoles providing basic radar-type displays with pertinent symbology which is computer-refreshed approximately 16 times per second.

2. The Digital Clock which is updated every millisecond and provides to all AEGIS AN/UYK-7 computers a one-word count of the elapsed time in milliseconds.

3. An MK 72 Signal Data Converter which provides analog-to-digital, digital-to-digital, and digital-to-analog capabilities for time-multiplexed interfacing for two channels between the Weapon Direction System and MFAR Computers and their resepctive peripherals.

4. An MFAR Signal Processor, containing an I/O Buffer-Synchronizer and a Digital Target Simulator, which provide the operational interface between the radar and the MFAR computer, and store test target digital data patterns to provide stimuli for test and fault diagnosis of the digital portions of the Signal Processor, respectively.

5. An MFAR Display Video Formatter for providing formatted displays of MFAR Video data, an MFARC Test and monitor console, and a WDS/Fire Control System Test and Monitor Console.

6. A Digial Data Converter, which acts in response to a command from the C&D computer, interrogates the ship sensor system, and returns digitized measurements of the ship's speed and heading to C&D.

7. The GMLS-MK-26, which is a twin rail launcher capable of using a mixed load missile magazine.

2.1.4 The Operating System (Executive)

The AEGIS Tactical Executive Program was designed to effectively manage the real time process, which is an inherent characteristic of the AEGIS system. In brief, ATEP performs the following six tasks:

1. Processing Interrupts generated by programs and equipment, according to the priority structure of equipment faults, program faults, I/O errors, and requests for ATEP services.

2. Scheduling, via a priority-ordered queue of task modules awaiting execution.

3. Storage Allocation that can be requested and released at any time.

4. Message Processing - by providing a capability for modules to send and receive messages.

5. I/O Control - managing the operation and usage of the I/O channels.

6. Dynamic loading of non-core-resident modules.

ATEP also provides linkages between application modules and user-supplied service routines, as well as error handling facilities and utility services. ATEP's generalized design although conceived for a shipboard system, provides it with the capability of operating in varying tactical environments that utilize the AN/UYK-7 computer, including airborne, shore-based, and subsurface applications.

2.1.5 ATEP Performance Requirements

The fast reaction time, high firepower, and constant operational availability of AEGIS require a high degree of performance by the executive, especially in the areas of core utilization and peripheral devices. The standard AN/UYK-7 Common Program Executive did not provide enough flexibility for the system.

It was a basic requirement from the start that the ATEP's contained in the AEGIS be essentially identical and modularly structured to permit deletion of facilities not required when a ATEP is assigned to a specific role. In order to satisfy this requirement of standardization and commonality, and yet at the same time provide a computer executive program of sufficient effectiveness to support the different tactical missions to which it can be assigned, the performance and

design of the ATEP evolved as a consequence of the meshing of the performance requirements of the individual segments with their distinct tactical missions. Thus, the EDM-1 ATEP is identical to the EDM-3 and Operational Ship ATEP to the maximum extent possible within the scope of the task.

2.1.6 ATEP Functions

ATEP operates in each AN/UYK-7 computer of the AEGIS C&CG to control and provide services to the application modules which reside in each of the three segments of that group. The ATEP provides interfaces between application modules and ATEP managed system equipment, and between application modules and the operators of the system. Additionally, ATEP loads and links modules as they are operated in AN/UYK-7 object code from CMS-2. (ATEP is written to be initially compiled using CMS-2).

The following major logical functions are performed by ATEP:

a) Integrated real time process control which is responsive to the dynamic real-time tactical mission needs.

b) Data Management, by managing access to common, private, temporary, and scratch pad areas by application modules according to their specific needs.

c) Interface management, including initiation and control of I/O and providing the capability for routing intra- and inter- computer messages within a segment.

d) Resource management.

e) Error management.

f) Application module configuration management, and services management, (library subroutines and utility functions).

2.1.7 ATEP Design Overview

The primary functions of ATEP are the supervisory allocation of AN/UYK-7 resources, the scheduling and monitoring of real-time computer program tasks and the routing of error associated interrupts to fault analysis programs. In addition, ATEP ensures the integrity of the computer system through the optimum implementation of memory protection features and memory retrieval in the event of power level faults. To achieve these ends ATEP is structured into several small modular computer programs or "processors". In many cases the processors can operate in a "stand-alone" environment, thus providing the capability to discard unwanted functions and conserve core storage.

2.1.7.1 Loading Processor

The ATEP loading processor loads operational computer programs from magnetic tape into AN/UYK-7 memory. Loading is executed using relocatable addressing logic, check summing, and memory bank separation of instructions and data. Instructions are stored in different memory banks than data, which permits use of the AN/UYK-7 overlap feature. This decreases memory access conflicts and produces faster instruction execution times. The loading processor also loads non-core-resident (NCR) programs, when scheduled, from the high-speed magnetic disk (RD-281).

2.1.7.2 Interrupt Processor

The interrupt processor traps and analyzes all interrupts emanating from the AN/UYK-7 central processing unit. Error interrupts cause a transfer to an appropriate error processing program either in ATEP or in an operational program. Scheduling and service requests from operational programs cause a transfer to the appropriate ATEP processor for further analysis and action.

2.1.7.3 Scheduling Processor

The scheduling processor inserts entries into the system task scheduling queues according to priority. Tasks are categorized as periodic, successor, and message. Successor tasks are scheduled according to a dynamic priority, which corresponds to their absolute position in the scheduling queue. Message tasks are scheduled in an identical manner to successor tasks, except that each task has a tactical message associated with its scheduling. This mix of scheduling capabilities is the aspect of ATEP that permits AEGIS to adapt to an ever changing tactical environment.

2.1.7.4 Dispatching Processor

The dispatching processor scans the scheduling queues in search of the highest priority tasks, and transfers central processing unit control to the selected candidate. Tasks are scheduled for dispatch according to clock-timed interrupts and/or task priority. A "preemption" capability permits current operating tasks to be temporarily suspended in favor of a pending task of higher priority.

2.1.7.5 Common Peripheral Processor

The common peripheral processor is scheduled by the ATEP scheduling processor as a result of operational computer program requests for I/O services to system-shared AN/UYK-7 peripheral equipments. This processor contains the AN/UYK-7 I/O controller channel program logic required to communicate with the magnetic tapes, magnetic disks, operator alphanumeric CRTs, teletypes, and high-speed printers.

2.1.7.6 Timing

The executive program must accommodate the requirements of AEGIS tactical programs for precise periodic and demand scheduling of the AN/UYK-7 central processor. These requirements are based primarily on the interface timing of the tactical programs with their associated equipments and vary from several milliseconds to several seconds.

2.1.7.7 Core Utilization

The executive program is allocated a maximum of 9000 words of core-resident storage in the AN/UYK-7. This ensures that sufficient stoarge is available for the operational programs to fultill their mission requirements.

2.1.7.8 Peripheral Services

Common peripheral devices, such as magnetic tape units, disk memories, and display consoles, are shared by more than one system of the AEGIS complex. Input/output programs that provide interface between these shared devices and the operational programs are an essential design requirements of the executive program.

2.1.8 Future Development

The present ATEP design has been formulated to fulfill all computer systems supervisory requirements for use in the AEGIS Engineering Development Model (EDM 1). Continuing development may require additional capabilities such as memory sharing and/or multiprocessing. Studies were made to determine the general design changes required to provide these capabilities and to "slim down" ATEP. This slim down process was accomplished by optimizing the object code, thereby reducing the executive's core storage and timing requirements.

ATEP's generalized design, although conceived for a shipboard system, provides it with the capability of operating in varying tactical environments that utilize the AN/UYK-7 computer, including airborne, shorebased, and subsurface applications.

ATEP has been extended to support multiprocessing (ATEP/MAX) and also to support shared memory multiprogramming (ATEP/MMS). These executives are discussed in Chapter 4.

2.1.9 Analysis of ATEP

Features of ATEP that are very useful and should be retained in a standard Navy executive include ATEP's memory management and interrupt handling schemes.

ATEP supports "common" data areas and instructions along with dynamic allocation and control of memory. Additionally, ATEP manages the access to common, private, temporary, and scratch data areas by application modules according to their specific needs, and provides for data base protection and integrity. Flexible memory management is desirable because of the functional organiation of Navy tactical systems. Usually organized into separate sub-tasks, a tactical system consists of many tasks, each performing its own job, working out of a common database. Dynamic memory control must also be present to allow changing system demand (e.g., more targets to watch) to be handled by their respective tasks (i.e., if there are more targets to watch then the target monitor task will need more buffer space). Flexible memory management also benefits the system by increasing the amount of reentrant code that can be used (reentrant code must use different storage for each task executing).

ATEP's interrupt management is process driven--that is, upon reception of an interrupt, the appropriate task is scheduled. This is desirable for many reasons. Among these: the interrupt handling mechanism is much "cleaner"; the interrupt handler is able to take advantage of other parts of the system (e.g., the scheduler will handle the priority problems); and the interrupt handler is much more flexible. It is suggested that a Navy standard executive adopt interrupt strategies with the same features.

Although ATEP met the desired specification for an executive for the AEGIS missile system it would not serve as the standard Navy executive for several reasons. ATEP was designed to be an integral part of the AEGIS defensive missile system, and hence contains many features that are necessary for the AEGIS system but are not generally useful. It also lacks many features that were not necessary for the AEGIS system but are very important in a more general system. For example, features such as a more flexible scheduler, a file system, an improved inter-task message system, and an inter-task synchronization facility are not easily installed into the ATEP executive. This difficulty in modifying the ATEP executive exists because it lacks modularity, a feature that is extremely useful and would have allowed the missing features to be added. Additionally, ATEP does not support multiprocessing (although ATEP/MAX is a later version of ATEP that does), or non-standard peripheral devices.

2.2 SDEX-7

The AN/UYK-7 is the standard military computer for shipboard applications. It is a 32 bit word machine, with storage expandable to 256K in 32K increments, with a cycle time of approximately 1-1/2 microseconds. On shipboard it usually uses one or two RD281 disk drives with a capacity of 50 million bits per drive, with an average access time of 184 microseconds. Typical peripheral devices include a teletype, CRT displays and magnetic tape. There are three languages used on the UYK-7: the higher level languages CMS-2 and FORTRAN, and the machine language ULTRA-32.

SDEX-7 (Standard Executive) is the operating system for the AN/UYK-7 that is intended to serve the needs of resource management in a multiprocessor environment. The executive is designed to be fully independent of the user modules or applications programs. The Standard Executive (SDEX/7) has the following functions for the control of user modules:

1) Initialization
2) Scheduling
3) Interrupt Management
4) Input/Output Management
5) Error Management

2.2.1 Initialization

The purpose of the initialization portion of the SDEX/7 is to load and set both the SDEX/7 and user modules to initial states during a startup or restart procedure. This function determines the memory configuration of the AN/UYK-7, initialization of SDEX/7 and each AN/UYK-7 central processor, loading of user modules, and initialization of the SDEX/7 interface for data transfer and communication and central processor control.

2.2.2 Scheduler

The scheduling function determines the allocation of the central processor's resources. The scheduling function provides a method by which the user module tasks can request and subsequently be allocated central processor control for processing. The CP's resources are distributed to user modules depending on the user module processing requirements. When a scheduled task is completed, CP control is always returned to the scheduling function which again begins to search for the next task to be run. The scheduling algorithm may be selected at the SDEX/7 compile time. If no algorithm is given, the default is selected. This default allows successor processing tasks or tasks requiring CP control in response to a user module request to receive top priority; message processing tasks or those tasks used to receive and process messages are given next priority; time dependent tasks or tasks requiring periodic control of the CP are considered next; finally, those background tasks which are run on a time available basis are given lowest priority.

To summarize, Modules are scheduled in four ways:

1) As successor tasks.

2) As message receiving tasks.

3) As time dependent tasks (cyclic).

4) As background tasks.

Users can select task priorities, or if not stated, default ordering is:

Successor > Message > Time Dependent > Background

The scheduling function is multi-tier. All modules are categorized by type (tier); different criteria can be applied to each tier. Tier priorities and presence/absence of tiers is controllable at operating system compile-time. Tier priorities determine the order in which they are searched.

When a task starts, a clock is loaded with the maximum allowable run-time for that task. If the clock runs to zero, an interrupt is generated. If the task can be suspended, its environment is saved and restored later. If no suspension is possible, an error is generated.

2.2.2.1 Successor Scheduling

A running module can give the executive a list of modules which should be scheduled as successors. The modules are place on the scheduling queue and dispatched on a priority basis. Successor tasks may be suspendable.

2.2.2.2 Message Scheduling

A module may be scheduled to receive a message from another module. Messages are processed in a first-in-first-out (FIFO) manner: the first message posted is the first to be sent to its receiver.

2.2.2.3 Time Dependent Scheduling

There is no priority among time-dependent tasks; they are tested for execution on a round-robin basis. Time-dependent list is searched for a task whose time has come due. When one is found, the task is dispatched; and for a cyclic task (one that will be repeated at a certain frequency) the time of the next call is placed on the time-dependent list. These tasks can be suspendable.

2.2.2.4 Background Scheduling

Background tasks (low priority, non-time-critical) are time sliced; if a task doesn't finish in its allotted time, it is re-scheduled. Tasks are tested for dispatching in a round-robin manner. User-supplied time parameters can determine when tasks are due to execute, and also what is the interval between slices.

2.2.2.5 Additional Notes on Scheduling

Time slicing can occur for background, time-dependent, and successor tasks.

A module can send a message to four receiving modules at one time.

Modules can be dedicated to run on a particular CPU.

Only one level of preemption exists; the preemting module must be of short duration. There are 64 different priority levels.

There is no assurance of periodic accuracy for cyclic tasks; since periodic entrances are scheduled only after successor and message tasks have received control, cyclic tasks may not run on an exact cyclic basis.

There is a lack of performance monitoring in the system; the system is written in assembly language.

2.2.3 Interrupt Management

The interrupt management function receives and decodes all interrupts. If an interrupt is associated with executive processing, SDEX/7 will perform the required corrections, otherwise control is transferred to a user designated module defined during the initialization phase. These modules are usually handled through the error management function.

2.2.4 I/O Management

The purpose of the Input/Output management function is to provide a means by which user modules can initiate and control computer I/O operations. Through this function, the user can register responsibility for I/O interrupts on a channel basis, define interrupt actions taken by the I/O function on a channel basis, selectively enable and disable interrupts on I/O channels and initiate I/O chains.

2.2.5 Error Management

The purpose of the error management function is to identify all hardware and software errors upon their occurrence and take actions as directed by the user modules. This function allows the user modules to selectively register responsibility for processing any and all errors. If an error occurs for which no user module has registered responsibility, the error management function conditionally stops the CP, and waits for the computer operator to process the error. Upon the processing of an error, the function is able to resume processing as directed by the user module or the computer operator.

2.2.6 Other Features

Specific features of the SDEX/7 include a maximum of four central processors in the configuration with at least one IOC connected to all central processors. The SDEX/7 operates in the interrupt state of the AN/UYK-7 while the users are in the task state. All user modules must have an interface with SDEX/7 through which they can communicate and pass data. This interface is set up in the initialization phase. In terms of protection, two task base register/storage protect registers are used to represent base, displacement and memory protect information for each segment. There must be a minimum of one instruction per segment.

2.2.7 Analysis of SDEX/7

Designed as a candidate for the standard operating system for the AN/UYK-7, SDEX/7 supports many features that are desirable in a standard Navy executive. Multiprocessing with effective synchronization facilities, memory management that matches the hardware (i.e.g, uses the full capability of the AN/UYK-7 segmentation hardware), and application independence with a general user interface are facilities that should be retained. SDEX/7's interrupt and scheduling management is very similar to that of SDEX/20. SDEX/7 can support multiple AN/UYK-7 configurations and is structured such that the program for single processor configurations does not contain any instructions or data pertaining to multiple processor configurations. This feature is very necessary in a standard Navy executive as multiple processor configurations play an ever increasing role in tactical systems. Further, this configuration flexibility allows a common executive to be used for most configurations with only additional functions added for particular applications. The memory management facilities of SDEX/7 take full advantage of the AN/UYK-7 segmentation hardware. For reasons of operating system design and implementation, such an addressing scheme is very desirable (issues such as naming, protection, etc. are involved and are discussed in other F.O.S. reports). As in other Navy executives, SDEX/7 is application independent and communicates with the user modules via a general interface (it uses an ESR scheme identical to that of SDEX/20).

Although SDEX/7 addresses the requirement for configuration and application flexibility it offers very limited solutions. Basically, it is not modular in design and allows only a limited set of compile-time options to control final executive configuration. This is a severe restriction that requires the user to support an executive that probably will have many features (and hence the overhead of those features) that it does not need and allows no easy method for modification of the executive. A good example is the lack of explicit memory management features to

support stacks or heaps; such a facility is not easily added to the non-modular SDEX/7. Additionally, SDEX/7 lacks effective task synchronization facilities. Such facilities will become ever increasingly important as more applications will be programmed in a multitask manner. SDEX/7 does support a dynamic scheduler and a message communication scheme but this will not provide the necessary system primitives that are required for proper multitask synchronization. SDEX/7 also totally lacks a file system or any facsimile thereof. Hence responsibilities for file storage are left to the user.

2.3 SDEX/20

The AN/UYK-20 Standard Executive (SDEX/20) is an advanced Naval Executive that is designed to be functionally independent of user modules. It runs on the UYK-20 minicomputer, in a single processor system.

SDEX/20 is designed to be a flexible operating system, and the interface between it and user modules is general in nature. However, this interface may also be used by modules requiring special purpose handling. A wide variety of options exist in regards to error registration and I/O handling; user modules may take all or part of the responsibility in these areas, eliminating the need for the Executive to handle special configurations or equipment. However, little error handling or I/O processing will be done by SDEX/20; user modules must generally do all their own processing. SDEX/20 is designed to be independent of the particular UYK-20 computer configuration; any features of SDEX/20 that are dependent on the actual machine configuration are changeable at compile time. An example would be the particular device through which SDEX/20 is initialized into the system.

SDEX/20 presents its standard user interface through ESR requests. Through this interface, tasks may be scheduled, messages sent, and error handlers defined. SDEX/20 provides the following functions for user modules:

1. Initialization
2. Scheduling
3. Interrupt Management
4. I/O Management
5. Error Management

2.3.1 Initialization

SDEX/20 performs initialization whenever the computer is started or restarted. The initialization routine controls all processor functions and must first initialize the processor, load the initial system configuration as defined by the user, and then pass

processor control to each module for local initialization and registration of processing requirements/responsibilities. This would include registering for handling particular interrupts, for processing certain errors, etc. When initialization is complete, control is turned over to the SDEX/20 scheduler.

2.3.2 SDEX/20 Scheduler

The SDEX/20 scheduling function provides the means by which user modules are given processor time. Upon completing any user task, control again returns to the scheduler.

The following scheduling types are supported by SDEX/20; they may be selectively dropped at compile time (with the exception of message scheduling):

a. Successor Scheduling
b. Message Scheduling
c. Time-Dependent Scheduling
d. Background Scheduling

2.3.2.1 Successor Scheduling

A module is considered 'successor scheduled' when it is scheduled in response to a user module request at its successor entrance. A module may also be scheduled at this entrance through other means, so this scheduling type is more general than it appears. Successor tasks are scheduled before any other class of task, regardless of the individual tasks' priority. Within the class, scheduling is accomplished on a priority basis.

2.3.2.2 Message Scheduling

Message processing tasks are those tasks which require processor control to receive and process messages initiated by other tasks, including the executive. Message tasks are selected for processing on a first-in-first-out basis, assuming all higher priority requests have been filled. Any message task will be scheduled before a task of a lower class.

2.3.2.3 Time-Dependent Scheduling

Time dependent tasks are those which require processor time on a cyclic basis, often for repeated I/O or processor "bursts." The cycle time must be greater than a certain minimum that can be set at compile time.

The scheduler will not schedule a time-dependent task until its time to execute has passed; there is also no guarantee that rigid times will be observed. Special facilities allow very time-critical tasks to be run automatically without actually being processed by the scheduler (a very time-critical task can set a real time clock to interrupt the processor and force the task to be run at its time-critical entrance). If a module is due to be run, the scheduler updates its next time-to-execute and runs the job.

2.3.2.4 Background Scheduling

Background tasks run only when processor time is not needed elsewhere. They may also specify a minimum clock time at which to be run, in a similar nature to that of time-dependent tasks, but strict timing is not guaranteed. All background tasks are always candidates for suspension by a higher priority job.

2.3.2.5 Scheduling Policies

Although strict time-slicing is not performed, all background jobs are always suspendable and certain successor jobs can declare themselves suspendable for certain types of interrupts. The executive apparently lets a running job run until a significant event--e.g., termination or I/O forces the job into an idle state.

2.3.3 Interrupt Management

This executive function receives and decodes external interrupts. If a module has not registered to handle the specific interrupt, SDEX/20 will process the interrupt. If a module has been registered, processor control is given to the required module. Certain interrupts, of course, may only be handled by SDEX/20.

These would include timer runout, hardware failure, and similar serious errors.

2.3.4 Input/Output Management

The I/O management function enables the user to do all I/O, with or without obtaining explicit control over the I/O device. User modules may register to process any or all interrupts on a particular channel; this might be useful when one program is controlling a specific device (e.g., radar) that is not similar to the normal computer peripheral equipment.

2.3.5 Error Management

The error management function identifies and acts upon hardware and software errors. It allows user modules to register reponsibility for certain errors, and either pass control to the module or stop the processor when no module has been designated. Processing is resumed as indicated by the module or the computer operator.

2.3.6 Other SDEX/20 Features

SDEX/20 is completely independent of the user modules it services, and thus is not restricted to any particular environment. Communication with the executive is done only through ESR requests. The user-executive interface is general in nature but can easily be used by a job for specific and unusual purposes. This is true because in most cases SDEX/20 will not process user-caused interrupts or errors; therefore the user can easily use non-standard procedures or peripheral devices.

SDEX/20 will not process unanticipated errors, and stops the processor if they occur. This is a weak feature seen throughout the entire system. Much more executive support for both I/O interrupts and error handling should be available, perhaps as a compile-time option.

2.3.7 Analysis of SDEX/20

Designed to be functionally independent of user modules, SDEX/20 is an advanced Navy Executive for the AN/UYK-20 minicomputer. Features of SDEX/20 that are desirable and should be retained in a standard Navy executive are SDEX/20's application independence, scheduling capabilities, configuration control, and I/O flexibility. SDEX/20 is designed to be a flexible operating system with a general interface between it and the user modules. This generality allows it to be used for a wide variety of applications--more specifically, it is not "tuned" to one particular tactical job. SDEX/20 scheduling facilities are fairly general in nature and allow great flexibility in task scheduling; further, particular scheduling priority structures can be varied at SDEX/20 compile time. Although SDEX/20's particular method for achieving configuration flexibility (that is, giving the user modules the responsibility) is not desirable, configuration independence is a valuable concept and should be retained in a standard Family of Operating Systems.

I/O flexibility in SDEX/20 is achieved by allowing the user modules a wide variety of options in regards to error registration and I/O handling. Such flexibility is a desirable feature; however, SDEX/20's methods for achieving this flexibility (i.e., by giving responsibility to the user) are not advocated, (i.e., there are no default I/O handlers that can be used).

Although SDEX/20 was designed to be a general purpose Navy executive, it has several weaknesses that preclude it from serving as such. User and unanticipated errors must be processed by user module. If such errors are not handled by the user, SDEX/20 stops the processor; certainly not a form of graceful system degradation that is desirable in tactical executives. Additionally, SDEX/20 has very little I/O support facilities but instead places such responsibilities upon user modules.

Because this is a general purpose executive, support for standard devices should be available as an option so that standard I/O support facilities are not reprogrammed for each new application. Although SDEX/20 is somewhat configuration independent, it does not support multi-processing; further, such configuration independence is achieved by moving configuration dependent support responsibilities from the system to the user. Other weaknesses include lack of memory management facilities and no explicit support for reentrant code. Although SDEX/20 has some compile time options (e.g., scheduling options) it is not completely modular and allows no flexibility in replacing particular executive functions with other more applicable versions. This is seen as a major weakness that makes SDEX/20 both difficult to "tune" to particular tactical system demands and difficult to extend to fulfill future executive requirements (e.g., paging, multi-processing, alternative scheduling methods, etc.).

2.4 COMMON PROGRAM (CP)

The COMMON PROGRAM is the AN/UYK-7 executive software for the SSN-688 Central Computer Complex and the TRIDENT Command and Control System. It was introduced to the FOS group at N.U.S.C., Newport, Rhode Island, where it is being used for developing and testing systems for on-board use. These are principally systems for Fire Control which interface with other systems such as Sonar and Navigation.

2.4.1 Overview of Environment

CP runs on a standard AN/UYK-7 computer. Typical devices include teletypes, CRT displays, and magnetic tape. On the SSN-688, Fire Control shares the Central Computer Comples, CCC, (which consists of two AN/UYK-7's) with the Navigation System. Sonar is a separate system using its own AN/UYK-7. On the TRIDENT, Fire Control and Sonar share the CCC with other subsystems.

2.4.2 The Executive

The size of the CP used on SSN-688 is about 15,000 words. The COMMON PROGRAM provides little in the way of dynamic memory management. The programs are loaded from disk when necessary. Although the AN/UYK-7 has relocation registers, a fixed partition size is usually employed because of the separate functions of most of the programs. CP allows the user module to build messages in an executive data store and to receive messages in another executive area.

The scheduling algorithm provides priority, periodic, and time sliced options, but the majority of processing is performed using priority or periodic 5 msec "run to completion". CP allows 64 different scheduling levels within its first level of scheduling known as the "Priority Entrance".

Message handling is necessary for two types of messages - 1) those within a user (i.e., an application area, such as fire control) and 2) those between "users" (e.g., between sonar and fire control). The CP message handlers are used primarily between users. Special procedures and conventions are often used to handle messages within a user for efficiency.

CP supports multiprocessing and multiprogramming. It also allows modules to be dedicated to given CPU's.

2.4.3 Analysis of CP

A survey was made of COMMON PROGRAM users at NUSC in June 1974. In general, users felt that COMMON was geared mainly towards an operational system and was inadequate in supporting the development aspect as it lacked many development aids. There is a lack of utility programs (e.g., there is no way to dump core). There are no tracing facilities to follow the flow of execution of programs. Moreover, the CMS-2 programming language does not run under COMMON, making a compile, load and go environment impossible. The conclusions arrived at from the NUSC survey are reproduced below:

"The Common Program in the operational environment for which it was designed appears to perform its executive task adequately. The primary shortcoming is lack of, or unnecessary complexity in, development aids such as utilities, linking and loading, and diagnostics.

This shortcoming results in significant expense and time loss in user software development which can be avoided, or at least reduced, by incorporating the development aids of the compiler (operating) system into the executive system. For example, utilities and a more adaptable I/O handler philosophy could be provided for the user, if the executive interfaced with the operating system (or compiling system) under which the executive users operated The present Common Program was written and in use before the CMS-2Y existed, and there is no way that modules of one can be utilized by the other.

In the future, separate development of I/O handlers, utility packages, and loaders for executive systems should not be necessary. If the operating system that includes the compiler is constructed in a modular enough fashion, the executives for the various Navy systems could be constructed, at least in part, from pieces of the operating system. For example, in terms of the present Common Program, instead of having a unique Common Peripheral module for the handling of I/O, the CMS-2Y CINOS module might have been utilized."

CHAPTER 3

EXECUTIVES FOR AIRBORNE APPLICATIONS

3.1 Introduction

Avionics applications in the Navy perform a wide variety of real-time functions, such as automatic flight control, radar signal processing, controlling displays, navigation, firing missiles, and communications control. As almost all the work is real-time, few capabilities are provided for program generation (a serious problem for those involved in software development efforts).

The variety of functions performed in avionics applications fall very well into the categories described in Chapter 1. For example, navigation is usually performed by sampling signals from radars. Weapons control invokes a large amount of data reduction (e.g., controlling the flight of a missile). The tactical situation that an aircraft encounters invokes a large amount of data analysis.

This chapter discusses the P-3C Update executive program, used on the P-3C fighter aircraft, and the PROTEUS executive for the PROTEUS system. Also discussed are two applications which were studied at Grumman Aircraft Corporation at Bethpage, Long Island, namely the F-14 fighter system and the E-2C early warning system.

3.2 The P-3C Update Executive Program

The P-3C Update is the executive program for a complete tactical system to be used on board the P-3C aircraft. The main system consists of a CP-901 computer (similar to a Univac 490 in instruction and character sets) as well as an enhanced CP-642B.

3.2.1 Hardware

The CP-901 is a 65K, 30 bit machine with no floating point. It has a number of control registers, 16 I/O channels, a real time clock, and an access time of a little under a microsecond. It has a cycle time with overlap of 1 microsecond, with an average execution time of approximately 5 microseconds.

3.2.2 Devices

Only the lower 32K of memory can be accessed for I/O purposes. The major devices used on P-3C are:

1) Magnetic tapes, on line (the Honeywell airborne type with a regular 2400 ft. reel, but very slow transfer rate).

2) A 338,000 word drum with an average access time of 12-1/2 ms. The drum spins at 4800 rpm, and because a checkerboarding technique is used, two (2) revolutions are needed to get all the 1024 words per track off the drum. Write protect on the drum is on a 32K basis.

3) Sensors and Displays. There are four (4) multi-purpose displays used for tactical coordination purposes on board the new P-3C aircraft. All the displays are refreshed from the computer's memory but it would be preferable to have more capability external of the displays.

   The first display is used in the pilot's station. Basically, the pilot tries to follow a point on a 10" display, which helps him know where he is going. The second display is a horizontal situation indicator which outputs a series of banking commands. The third is a navigational control station with an auxiliary readout device which provides navigational and tactical information, and provides a means of accessing and modifying the data base.

   The fourth is a set of displays called "sensor stations", which are 15" multi-purpose displays providing radar and contact information and monitoring returns from buoys. Each of the I/O handlers contain the characteristics of the device.

3.2.3 The Executive Program

The P-3C Update Executive program runs on the CP-901 computer and provides a multiprogramming environment on a uniprocessor system. The P-3C executive is designed in a modular fashion, and provides for additions and modifications to the system in a simple yet controlled manner. The system is designed to be failsoft and will operate in a degraded mode.

3.2.3.1 Memory

Memory is divided into pages of size 2K. The upper 4 bits (11-14) is the page number and is translated to the appropriate paging register. The effective address is then generated. (Because the half word size is 15 bits, the greatest access is approximately $32,768$ words.)

One of the problems that was mentioned in a trip visit was that memory is protected only on the 2K boundaries (only write protected). A system that could offer protection within the page itself would be very useful to the P-3C's.

Core management is provided only internally, for management of transient tasks and files. Drum management is provided. Additionally, a file management system is supplied for use with files on the drum (or in core when the drum is not available).

3.2.3.2 Scheduling

There are a number of user tasks, each of which has associated with it some number of entries at which the task can be scheduled. Tasks can be permanent or transient; transient tasks are swapped between core and the system drum as necessary.

The scheduler is fairly sophisticated, involving five (5) preemption levels and 16 priorities within each level. The details are available in the previously cited reference. It must be remembered that of the 60 "periodic tasks" (units of work as seen by the executive), 15 of them are always active. The average execution time per task is approximately 3-5 ms.

The scheduler accepts requests from executive interrupt processing modules and from executive service request (ESR) modules. ESR requests introduce new work into system. Any necessary core allocation and drum reads (to bring modules into core) must be done.

If a task to be scheduled is already active, or in ready-to-run state, then:

1) queue shall contain an entry for each request of a task

2) all entries for a task must be at the same state/priority level

3) order is FIFO

Certain tasks (usually low priority periodic tasks) are not candidates for scheduling if they are active or ready-to-run, i.e., they have already been scheduled but have not finished running.

There is a limited set of tasks which should run within 10 ms from request time (data acquisition and transmission). There is a set of periodic tasks which run repetitively every 50 ms. There are several sets of long running tasks (700 ms) which share memory and processor time with each other. The majority of tasks are demand and should run within 500 ms of request. Also there are background tasks.

There are 5 preemption levels, with 16 priorities per level. A preempted task in any level has the highest priority in that level.

Preemption levels:

Level	
1	Tasks here cannot be preempted; they are dispatched in priority order.
2	Tasks in this level can preempt any lower level tasks unless they are 10 ms tasks or they have locked the data base.
3	These tasks can only preempt level 5 tasks (if they have not locked the data base or are not 10 ms tasks).
4	Tasks here can only preempt level 5 tasks (unless they are of the two (2) special types). A preempted level 4 task is the last to run within its priority level.
5	Lowest priority in the system. These tasks run only when no higher level tasks are ready.

The dispatcher places the highest level ready-to-run task in control of the CPU. The dispatcher must restore the physical environment of a preempted task when it is restarted.

Other considerations:

1) Tasks which can complete within 10 ms are allowed to run to completion.

2) A task which has issued a "lock data base" command and has not unlocked it will be allowed to complete its accesses and not be a candidate for preemption until the data base is unlocked.

3.2.4 Other Features

There are approximately 300 tasks in the system, of which 60 are periodic in nature. There is a common data base available to all 300 tasks. Displaying of data usually involves only a read only access. The display maintenance program is the only one that uses entire data base.

The executive includes an input/output control module, which includes handlers for system-required devices. This module can be expanded to contain handlers for other devices in the system. The P-3C Update Executive also includes extensive system analysis aids.

Most of the programming for the P-3C is in CMS-2, although the executive is written in direct code.

There is no dynamic linking or loading available. The memory replacement algorithm attempts to keep a task in memory as long as possible on the principle that if a task is used once, there is a high probability that it will be used again.

The executive time stamps all data that enters the system. The short term tasks can run for as little as 10 milliseconds. Long term tasks can run for as long as a maximum of 400 to 500 ms.

3.3 The PROTEUS System

The Proteus system is an accoustic processing system for real-time analysis of incoming sonar data. It consists of three Proteus Advanced Signal Processor units (ASPs)-- the analyzer, post-processor, and display processor--each consisting of at least a general purpose processor (CP/IO) and optimally additional hardware to aid in each processors respective functions. It is intended to be the Navy standard interim airborne signal processor with planned utilization on the P-3 aircraft on helicopters and on the "lamps" aircraft.

3.3.1 Hardware

The general purpose processor (CP/IL) is architecturally the same in all configurations of the ASP; however, it is extensively microprogrammible. There exist tailored CP/IO instruction sets for each of the different configurations of the ASP (analyzer, pest-processor, a display processor). The basic instruction sets are very close to that of the IBM 360/370 computer.

3.3.2 The Proteus Executive

The Proteus ASP has two executives currently undergoing design and implementation--IBM's originally specified executive, and the Naval Air Development Center (NAVAIRDEVCEN) general purpose executive. The later will probably be the executive chosen for fleet utilization and will be discussed here. Basically, the objective of the executive is to allow tasks to execute in a periodic fashion to analyze incoming sonar data. Depending on the particular ASP configuration (i.e., analyzer, post-processor, display), additional hardware must be kept (e.g., the arithmetic processors in the analyzer configuration).

3.3.2.1 Memory Management

In the initial design there is no dynamic management. Although all the programs currently fit in core, the system is being provided with facilities to support swapping.

3.3.2.2 Scheduler (Task Management)

There are three (3) types of scheduling algorithms used on PROTEUS.

a) A fixed priority scheme whereby a task gets control of the CRU according to its priority.

b) A "least time to go" algorithm for tasks whose run time has been very accurately predetermined. This algorithm appears to be the best for their use as far as efficient scheduling is concerned. If a very critical task must be performed, it is simply given a very high priority level.

c) Whenever the system is idle, a "background monitoring" program will be run.

3.3.2.3 Initialization

This is accomplished with an Initial Program Load (IPL) process whereby the Executive and user modules will be loaded into memory. Control is then transferred to the Executive initialization routine.

This routine determines where the Executive itself ends and the "IPL table" begins. This table contains a list of entry points to receive control of the central processor during initialization. The executive scans each entry and sends an initialization message. This "wakes up" each module, and when finished, the executive continues internal initialization.

3.3.2.4 Input/Output

Only the executive can initiate I/O directly; user modules must issue an "I/O Request" ESR to the Executive. I/O requests are queued if an open channel is not available.

3.3.2.5 Interrupt Processing

During Executive initialization, software linkup with the interrupt mechanism is set up to the appropriate interrupt processor. All status information is saved for restart. During interrupt processing, higher priority interrupts remain enabled.

The executive is subject to "pseudo-periodic" interrupts, mainly from an intercomputer channel. In general quick responses will be required from the system, with certain functions having to be performed every 50 ms.

3.3.2.6 Error Management

Proteus's hardware automatically traps storage protection and privileged instructions. The Executive also validates all user supplied parameters, such as entry point addresses, I/O requests, etc.

If an arithmetic error occurs, the Executive suspends the task and sends a message to its parent module. This module may take any corrective action required. If the task is not aborted, execution will resume.

If the error was an addressing or privileged instruction violation, a message is sent to the parent module and the task is aborted. No restart is attempted.

I/O errors cause the Executive to attempt to recover and make the channel usable. The Executive then reports to the task, via the I/O status word.

Main errors, such as power failure, halt the machine with the error displayed in a fixed part of memory, or on a panel readout.

3.3.2.7 Performance Monitoring

Proteus monitors both the computer system and itself continually. This allows optimization, system debugging aid, and fine tuning.

3.3.2.8 Data Management

Data management is different in scope for different functions, usually consisting of tables of data in core memory. Data is always recorded, with the system keeping track of information, (such as: how many times the interrupt occurred; when a task was last scheduled; and how many times it was scheduled).

3.3.3 PROTEUS CP/IO Executive Analysis

The PROTEUS CP/IO executive is a general purpose operating system designed to handle user needs in any of the configurations of the advanced signal processor (ASP). It has a standard user interface and is basically responsible for the organization, control, and supervision of the application programs. Such user programs are preplanned into a system and finally "compiled" with the executive so that a fixed system is the result. This executive offers few new facilities, the most outstanding being a limited synchronization capability. Such a feature is highly desirable but as this executive has no time-slicing ability much of the power of this feature is lost. As user modules are preplanned and linked into the system, the PROTEUS executive offers no memory management facilities and does not have the capability for non-resident task execution. Further, it contains no I/O drivers but rather acts as a channel manager for user tasks who must have the I/O channel programs themselves. The facilities offered by this executive are a subset of the desired facilities in a standard Navy executive. Modularity of design could not be determined from the referenced literature.

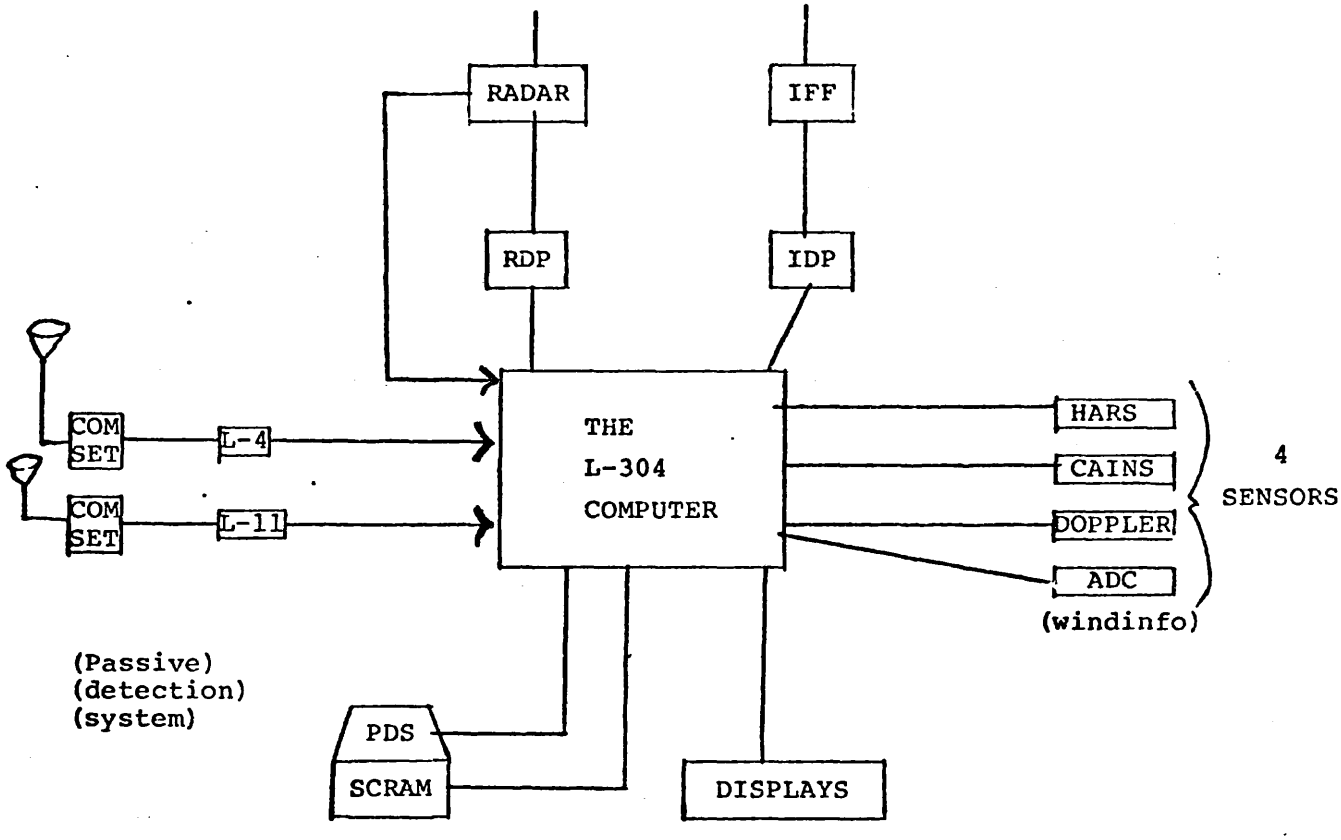
3.4 The E-2C Airborne Early Warning System

There are four (4) functions performed by the E-2C system, each of which is supported by a different computer as follows:

	FUNCTION	MODEL
1)	Radar Tracking and Data Management	LITTON/304F
2)	Navigation (Advanced Radar Processing System)	LITTON/728
3)	Air Data System	CONRAC/1085
4)	Passive Detection	ARMA/MICRO D

Different formats have to be used for each computer system as all the four (4) computers have different word lengths. Backup systems are available for each of these systems. It must be noted that the function of the E-2C aircraft is of early warning and not of a fighter. The basic system aboard the E-2C aircraft can be described best by the following diagram:

THE E-2C SYSTEM



3.4.1 Subsystems

The subsystems and programs on the E-2C are as follows:

a) The IDP is a very busy routine which records data such as radar responses and height information as inputs to its track establishment scheme. The tracking program receives data from the IDP to track the target paths such as ranges and azimuths. The tracking program uses 6K 32 bit words.

b) The Link-4 and Link-11 are air-to-air links and air-to-ground links respectively, which are used for both transmitting and receiving. The L-11 program consists of 7K 16 bit words and is tied in to the aircraft carrier The L-4 program consists of 2692 16 bit words, and ties the system in to other aircrafts. These links basically extend the capability of the standard Navy links so that they can be used by the Army and Air Force. The links are used for controlling the fighters and processing command intercepts on flight missions.

c) The "CAINS" system (one of the four (4) sensors connected to the L-304) is a navigations system, and is claimed to be the most accurate form of navigations system that is available.

d) The Air Data Computer analyzes wind information and is discussed later in this report. "SCRAM" is the Signal Conditioner Readout Alarm Monitor, and it interrogates a "scrambler" approximately every two seconds.

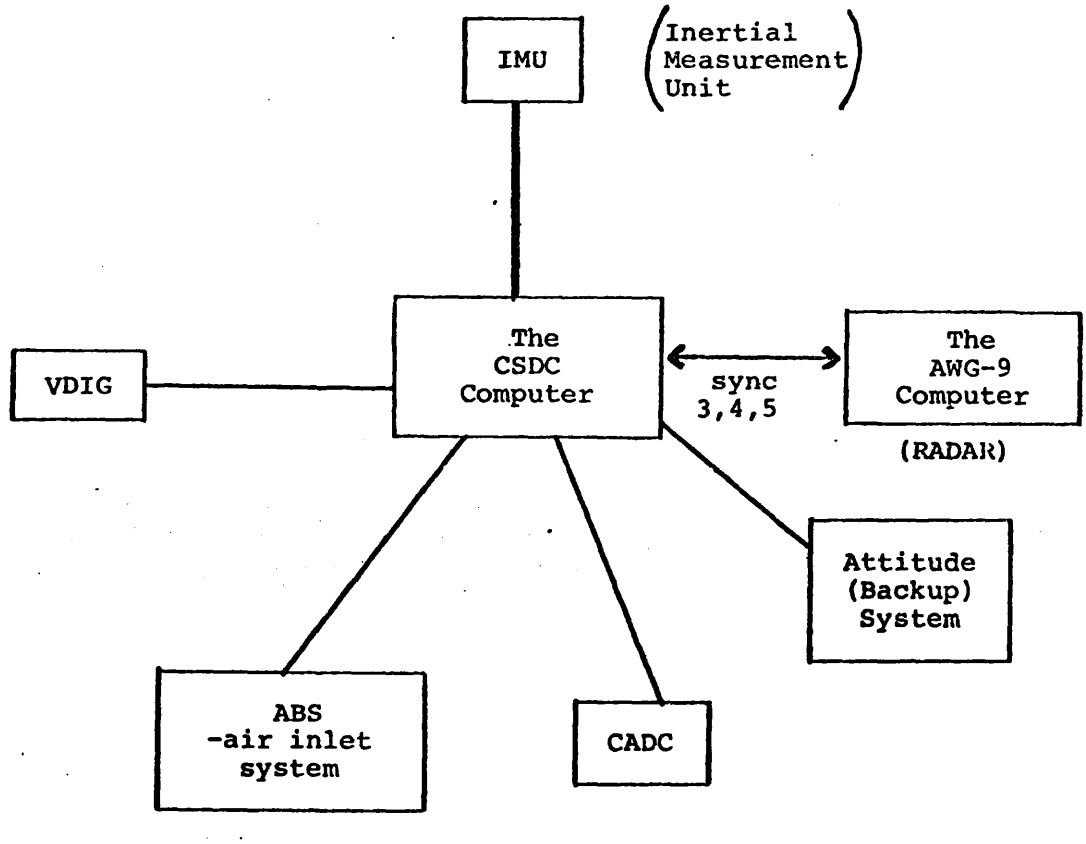
3.4.2 The L-304

The Litton 304 computer is the center of the system and controls all the displays and responds to the inputs from the sensors. It is a 64K, 32 bit machine, with an additional 16K of core that can be made available. It is connected to a detection front end via a radar detection processor; (RDP - which employs methods known as "sweep the sweep" and "pulse the pulse"). The L-304 also provides navigational information, produces reports on target position, and drives the stabilization controls.

All the programs in the L-304 are dedicated, and at times one of the processors can be idle. When the programs are not running the system usually performs machine checks. The Grumman personnel felt that more supervisory capability was required by the operating system. The task scheduling algorithm has 64 levels of priority, but the tasks are never guaranteed to run to completion.

3.5 The F-14 Fighter Aircraft

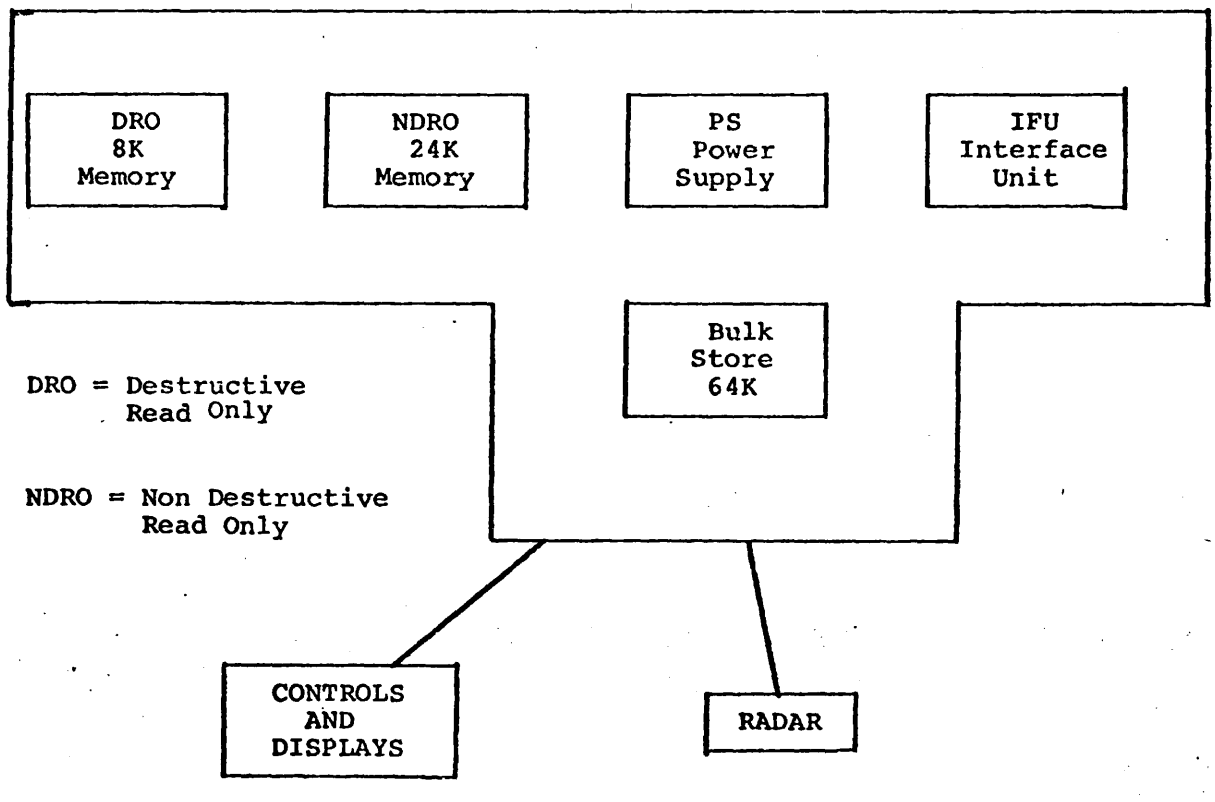
The total system architecture of the computers on board the F-14 aircraft is shown below:



3.5.1 The AWG-9 Computer

The AWG-9 is the basic computer aboard the F-14. It is responsible for communication with radars, controlling missiles, driving displays, and accepting inputs from other parts of the system. It is a 24 bit machine with an add time of 1 microsecond. The basic architecture of the AWG-9 is shown below:

The AWG-9



The 24K available in the NDRO is divided into an executive routine, a radar processor, a general processor, and a navigational processor. In addition, there is a built-in test routine, as well as an on-board checkout routine that verifies that all the avionics equipment is functioning correctly.

The executive routine basically performs memory management and relocation functions. In general, programs are loaded from the tape into the DRO. However, the hardware cannot affect programs resident in the NDRO.

3.5.2 Languages

The programs in the AWG-9 are vast attack programs (known by names such as "Phoenix", "Sidewinder", etc.). The programs are written in METAPLAN, (the language developed by Hughes Aircraft Corp.).

3.5.3 Data

The data links interface via sync lines. Data such as clock data and sync data from radar are continually coming into the DRO of the AWG-9. "Sync 3,4,5" are the Standard Serial Interfaces.

3.5.4 The CSDC

The Computer Signal Digital Computer (CSDC) processes analog signals. It is necessary to utilize the CSDC because the interfaces onto the CSDC are not all digital. The CSDC is a 20 bit machine with an add time of 7-1/2 microseconds. The CSDC performs many of the same functions that the CAINS System performs on the E-2C aircraft. Programs on the CSDC are all assembly coded.

3.5.5 The Air Data Computer

The Air Data Computer System (The ADC) is a two channel computer which sends data to the CSDC. Its main function is to position the wings of the aircraft according to its altitude and speed. It is responsible for wing sweep control, stabilization control, determining true air speed, and indicating the angle of the plane to the airstream. (However, if the computer system ever fails, the pilot still has recourse to using his manual controls.)

A dual channel computer was chosen because it lent itself to computing high order polynomials serially. The ability to compute high order polynomials was considered important because the equations for the wing sweep of the plane were all n^{th} order polynomials of the form $ax^n + bx^{n-1}$..... .

3.5.6 Other Subsystems

The "VDIG" is a display used by the pilot of the aircraft. The Attitude System is a backup system that informs the pilot of the "attitude" of the aircraft - (e.g., tracking information, yaw, roll and pitch analyses, navigational information, and informing the pilot of his distance and position relative to an enemy airplane). Data comes in at the rate of 16 words every 120 milliseconds.

In general, the AWG-9 and the CSDC have to have a high throughput time. The fastest program is the RTGS which provides the "optimal gun solution". This program has to screen radar information, and then compute the bullet trajectory in a very short time period.

3.6 Summary

Most of the airborne applications that were studied by the FOS group consisted of standardized real-time functions, with a high degree of interdependency between components of the system. For example, targets must be identified, the plane's flight must be adjusted accordingly, missiles must be launched and guided and communications must be kept open. In an attack situation these functions often occur almost simultaneously.

Because of the number of different functions that must be performed it is not unusual for these sytems to use several mini-computers running in parallel. There is usually a central computer system (e.g., the L-304 in the E-2C System) while the others function on different subsets of the data.

Realiability is very important in airborne applications. Most of the systems have some sort of redundancy built into them, and in most cases all computerized functions can be taken over by a human operator.

Very little standardization of equipment or operating system was found in the avionics applications that were studied. It seemed that much time, money, and effort could have been saved by using standard equipment for the fundamentally similar functional requirements.

CHAPTER 4

EXECUTIVES FOR SHIPBOARD APPLICATIONS

4.1 Introduction

The AN/UYK-7 is the standard military computer for shipboard applications. Although the standard executives for the AN/UYK-7 are ATEP and SDEX/7, many other special purpose executives are used. Examples of these are ATEP/MAX and ATEP/MMS which are two extensions of the basic ATEP executive.

The BQS-13 is an executive for the AN/UYK-7 which has been designed for use with a sonar system. It was introduced to the FOS group at NUSC, New London, CT.

These executives are discussed in the following sections. The UYK-20 is a newer mini-computer that has been developed for the Navy. The communications oriented operating system for the UYK-20 is discussed in Chapter 5, while the standard executive for the UYK-20, SDEX-20, was discussed in Chapter 2.

4.2 ATEP/MAX

ATEP/MAX was originally designed to meet the requirements of the AEGIS system and was subsequently nominated as a standard executive because of the flexible nature of the program. The original ATEP (discussed in Chapter 2) utilized multiple AN/UYK-7 computers in a uniprocessing environment, and was implemented in Engineering Development Model 1 (EDM-1). (In EDM-1 there were three distinct functional segments: Command and Control (C&C), Weapon Direction System (WDS), and the multifunction array radar [AN/SPY-1]).

Because of its flexibility, ATEP/MAX was being considered, along with SDEX/7 to comprise the standard executive for use with various configurations of AN/UYK-7 computer systems. The ATEP/MAX flexibility supports not only, uniprocessing, but also the multi-processing. ATEP/MAX remains completely downward compatible with the current ATEP concept and also meets the requirements of other systems employing either dedicated or non-dedicated CPU tasking.

The AEGIS environment for the ATEP/MAX uses a particular complement of AN/UYK-7 computers and related military electronic equipment. In the more general case, however, ATEP/MAX operates in a multi-CPU AN/UYK-7 environment. The AEGIS environment, which ATEP supports requires only uniprocessing, and is a special case of the more general ATEP/MAX capability.

4.2.1 The Executive

ATEP/MAX was developed at FCDSSA, San Diego, to not only satisfy the requirements of AEGIS as a minimum, but also to:

- Efficiently exploit the higher computational power of a multi-CPU AN/UYK-7 environment.
- Provide for the sharing of frequently accessed data among the several computational elements.
- Provide an environment wherein several CPU's have access to all the resources (e.g., modules and data) needed to perform any given task.
- Provide one copy of an executive in AN/UYK-7 memory, servicing the several CPU's, providing core savings as compared to the uniprocessing case in which each CPU requires its own copy of the executive (as in).

4.2.2 ATEP/MAX Functions

ATEP/MAX provides effective management of real-time processes in a single or multi-processer AN/UYK-7 environment by processing interrupts, scheduling application modules, servicing I/O operations, routing user-messages and providing for linking, loading, and graceful degradation.

4.2.3 ATEP/MAX Design Overview

ATEP/MAX supports multiprocessing with only one copy of ATEP/MAX necessary. Uniprocessing is provided for as a special case. ATEP/MAX is designed to be a failsoft system which will continue to operate in a degraded mode in the case of certain system components.

ATEP/MAX is designed as a modular system, with modules tailorable at compile-time. ATEP/MAX recognizes a number of entry points associated with each user module, each to be entered upon a specific occurrence. It provides storage management for the user modules. There are four types of segments provided: data private to the module, common data, temporary data (which can be dynamically requested), and scratch pad data. Some modules reside permanently in core, while some are transient, being rolled-in upon scheduling. Roll-out is not provided.

ATEP/MAX provides input/output management for the user programs, and includes (subject to selections at compile-time) standard peripheral drivers. There is also a selectable set of file management routines.

The system includes a comprehensive set of utility programs. There are also a large number of selectable measurement tools available.

4.3 ATEP/MMS

The ATEP/MMS (Multiprocessing and Memory Sharing) is a further extension of ATEP which supports each of the three major processing concepts presently in use in the AN/UYK-7 computers; i.e., uniprocessing, multiprocessing, and the concept of multiple CPU's, each with its own executive, communicating through shared memory. It was designed to adequately service the planned improvements to the AEGIS system, some of which are as follows:

4.3.1 Overview of Operating Environment

In EDM-1 AEGIS, the radar handling computer (formerly MFAR control, now called AN/SPY-1 control) used two AN/UYK-7 computers each with its own single CPU. Messages between the CPUs had to go via inter-computer I/O channels; this was costly in overhead. For the future AEGIS the AN/SPY-1 control CPUs will be placed in the same computer so they have access to common memory.

Other multi-CPU computers will be provided for the other AEGIS functions. A multi-CPU computer will communicate with other multi-CPU computers via inter computer channels.

The planned AN/SPY-1 computer will have four CPUs. The revised ATEP will have multi-processing capabilities, whereby two or three CPUs share the same ATEP and other programs. However, AEGIS will probably operate using a separate copy of the executive with each CPU and with dedicated application modules. Modules in the same computer will be able to access the same data even though the modules run on different CPUs and special features will be added to ATEP to allow for extensive communication among ATEPs on different CPUs in the same computer.

Loading of the program was difficult in EDM-1 because it involved many operator steps and insufficient diagnostics and recovery steps in the initialization program, particularly when a tape failure

occurred. The EDM-1 system now includes a fast reload from disk following a failure. It is intended that the whole error processing and recovery area will be enhanced for ATEP for the next AEGIS. The error recovery was generally deferred during the EDM-1 ATEP design.

In the improved system, although debugging capabilities will be available on-line, programmer tools etc. would not run under ATEP, but under CMS-2Y, (CMS-2 hosted on the UYK-7). CMS-2Y is the operating system that supports CMS-2. The system would also have several copies of the system state, with the added capability that the saved PSW's can communicate with each other.

4.3.2 Design Overview

The ATEP/MMS is composed of the ATEP/MMS Kernel and the AEGIS Dependent Executive Program (ADEP). Different subsets of these components are selected for use in a particular AN/UYK-7 configuration for a particular application.

The ATEP/MMS Kernel provides the central core of the executive control functions and operates in the AN/UYK-7 interrupt state. It provides centralized services which are application independent, such as the handling of interrupts, the scheduling of the overall systems operation, the apportionment of I/O. The ADEP consists of application-oriented functions such as loading, standard peripheral device handlers, utility services, and common subroutines, and are provided within the ATEP/MMS.

(NOTE: The ADEP Computer Program Performance Specification is not yet available.)

4.3.3 The ATEP/MMS Kernel

The ATEP/MMS Kernel is that portion of ATEP/MMS that performs the basic services of resident initialization, interrupt processing, scheduling, dispatching, I/O processing, memory management, message processing, fault processing and utility interface.

The ATEP/MMS Kernel provides executive functions for real-time applications which operate on the AN/UYK-7 computer. It operates in a multiprocessor system, in a memory sharing system, or in a unit computer system. The ATEP/MMS Kernel allocates the computational resources of the AN/UYK-7 and provides an interface between the computer and the user modules. The interface between ATEP/MMS and the user modules permits ATEP/MMS to be used in a variety of application systems without modifications to that interface.

In any particular configuration there may be various capabilities that are not used. It is possible for such unused capabilities to be deleted at compile time.

The ATEP/MMS Kernel is tailorable. There is a provision for dropping at compile time those functions not required for the particular application. For example, in a unicomputer system all multiprocessing and shared-memory features (and overhead) can be dropped. Table sizes can be selected.

Also tailorable at compile time and at load time is the choice of the AEGIS Dependent Executive Program (ADEP), including the choice of routines to control the standard computer peripherals (magnetic tapes, disc and printers) and the choice of a program loader.

4.3.3.1 Capabilities of the ATEP/MMS Kernel

Since the ATEP/MMS Kernel is a standard executive for the AN/UYK-7 computer, it must provide services for many of its possible configurations. It is capable of providing services for:

- Multiprocessing systems with or without dedicated tasking for up to three CPU's and up to two IOC's or two CPU's and four IOC's and all memory addressable by such a system; full interconnectivity is assumed.
- Memory sharing systems with up to four CPU's and four IOC's with memory addressable as limited by physical connectivity rules of the AN/UYK-7 computer.
- Unit processing systems with or without extended memory.
- Combined multiprocessing and memory-sharing systems.

For the multiprocessing case, the ATEP/MMS Kernel provides load leveling and graceful degradation. Load leveling is that procedure whereby any processor does any job available except for tasks that may be dedicated to a particular CPU. Graceful degradation is that procedure whereby after casualty of any CPU, IOA, IOC, or memory unit, the surviving equipment takes over the tasks of the casualty unit and the entire operation continues but perhaps at a slower rate.

3.3.2 Structure of the ATEP/MMS Kernel

In order to satisfy the broad range of possible configurations, system architectures, and processing requirements, the ATEP/MMS Kernel is constructed in modular fashion. Inherent in the ATEP/MMS Kernel are the scheduling, dispatching, interrupt processing, and I/O control services. The Kernel consists of the following entities:

a) Resident Initialization Function, which initializes ATEP/MMS tables and dispatches user modules at their initialization entrances.

b) Interrupt Processing Function, which receives all AN/UYK-7 computer interrupts and calls the appropriate Kernel or ADEP routines.

c) Scheduling Function, which enters all module scheduling requests into a priority-sequenced queue. Modules may be scheduled upon the receipt of a request from another module; or periodically, after a specified interval of time; or upon the receipt of an I/O interrupt.

d) Dispatching Function, which selects the request with the highest priority from the scheduling queue and dispatches this module at the specified one of seven possible entrances; it also provides module termination services.

e) Input/Output Processing, which provides intercommunicatio between task-state modules and their associated IOC channel programs.

f) Memory Management Function, which controls the assignment of core storage that can be dynamically assigned.

g) Message Processing Function, which provides for the transmission of messages from module to module(s).

h) Fault Processing Function which makes up a data packet concerning any discovered fault and passes it on to an appropriate error processor.

i) Utility Interface Function, which provides special features that enhance debugging and measurements of the system performance.

The ATEP/MMS Kernel operates in the interrupt state. It provides the following executive functions for a wide range of AN/UYK-7 configurations:

a) Unit processor - One CPU; one to four IOC units, and up to 16 memory units.

b) Multiprocessor configuration - Systems with or without dedicated tasking for up to three CPU's and up to two IOC's or two CPU's and four IOC's and all memory addressable by such a system; full interconnectivity is assumed. A single copy of the ATEP/MMS Kernel is used by all CPU's. Individual user modules may be dedicated to a given CPU or shall be dispatchable in any available CPU.

c) Memory sharing configuration - two to four CPU's with two to four IOC's and up to 16 memory units. A separate copy of the ATEP/MMS Kernel is to be stored for each CPU. Individual user modules are dedicated to the individual CPU's. The ATEP/MMS Kernel provides the capability among the user modules so that a module can communicate with another module in a standard way whether the other module runs on the same CPU or a different CPU in the computer. The communication among CPU's is through memory units having data access to all CPU's and also by use of the Inter-Processor Interrupt (IPI).

d) Configurations with combined memory sharing and multiprocessing group of CPU's has memory sharing with other CPU's. Some examples are:

- Two or three multiprocessing CPU's using a single copy of the ATEP/MMS Kernel, memory sharing the same computer with an additional CPU that has its own copy of the ATEP/MMS Kernel.
- Two two-CPU multiprocessing CPU groups connected by memory sharing.

In all configurations, application modules operate in the task state, and a standard interface between these and the ATEP/MMS Kernel is provided. In any configuration the selected ADEP routines may include routines that operate in the interrupt state and routines that operate in the task state.

4.3.4 AEGIS Dependent Executive Program Portions of the ATEP/MMS

Application-dependent portions of the ADEP are selectable by the user. These may include routines that perform the following functions:

a) Initialization/Loading

b) Error Processing

c) Reconfiguration

d) Control of common peripherals

e) Utility functions

f) Communication among computers via intercomputer I/O channels.

4.3.5 Basic Operation of a Computer Program using ATEP/MMS

A computer program that runs under the control of ATEP/MMS contains the following components:

- ATEP/MMS Kernel
- Selected AEGIS Dependent Executive Program routines
- User supplied program modules and data areas
- Application Dependent Executive Tables.

The primary element of ATEP/MMS Kernel is the priority scheduling queue which shall contain a list in priority sequence of those program modules which are to be dispatched (i.e., given control of the CPU to carry out some task). The priority scheduling queue shall change dynamically as time progresses. As modules are dispatched in priority sequence at one of seven module entrance points as specified in the queue entry, their queue entries shall be removed from the queue; other task requests shall be entered at their priority level into the priority scheduling queue due to input/output interrupts, the passage of clock time, or a scheduling request from some user module while it is in control of the CPU. Thus, the ATEP/MMS Kernel provides the mechanism whereby a dynamically changing sequence of user modules is dispatched.

The ATEP/MMS Kernel is given control following an interrupt signal in the CPU. The Kernel processes the particular interrupt appropriately and then, depending upon the particular circumstances, turns over control to one of the following:

- The task state module or that portion of the Kernel that was running when the interrupt occurred.

- A different task state module, as specified in the priority scheduling queue.

- An ADEP routine. This routine may perform some special function and then return control to the Kernel, or, in the case of the detection of a serious error or failure, the ADEP routine may abort the system or initiate a reload.

A task state module turns over control to ATEP/MMS Kernel via Executive Service Requests (ESR's). The ESR call causes an interrupt to the CPU and the Kernel then carries out the particular executive service requested by the module before returning to task state. Typical services to be carried out by the Kernel include: assigning a temporary storage data area to the module, scheduling another module as a successor sending a message to another module, and initiating a specified I/O command to an IOC.

4.3.6 Example Computer Loads

The following paragraphs discuss the components of a computer load for each of the four modes of operation (uniprocessing, multiprocessing, shared memory, and combined multiprocessing with shared memory). Note, however, that there is a considerable amount of flexibility in setting up a system in terms of which components are included and also in specifying which ATEP/MMS Kernel features are included.

4.3.6.1 Components of a Unit Processor Load

a) ATEP/MMS Kernel (instructions and data) - The Kernel instructions and data areas will normally be placed in different memory banks to provide for memory overlap. Alternate load algorithms apply for systems which feature interleaved memory.

b) Other ADEP interrupt-state routines - These may include:

- Error Processing Routine - Although the Kernel will carry out minimal error processing, it may be desired to add a program to: analyze the error further, and/or report the error, and/or determine corrective action, and/or initiate recovery.

- Intercomputer Message Processor.

- Routines to interface with Common Peripheral modules. Such a routine is required if the user uses any Non-Core Resident modules.

- Recovery and reload programs.

Any such interrupt-level routine or routines is combined with the Kernel and the Kernel data to form a combined element in core.

c) User supplied Application Dependent Executive Tables.

d) Task state modules - These may have instructions and private data in different memory units or stored together on a single base register.

e) Data areas - Common data areas, temporary, and dynamic storage areas which shall be formatted by the Kernel after loading.

f) Common Service Routines.

4.3.6.2 Components of a Multiprocessing Load

The components of a multiprocessor load are identical to those of the Unit-processor load except as follows:

a) ATEP/MMS Kernel - A single copy of the Kernel instructions is used by all CPU's. However, the Kernel data is split into a part that is common to all CPU's and a part that has one copy per CPU. The common part holds constants and data items used in common by all CPU's (e.g., the priority scheduling queue). The part that is private to each CPU contains status data and Kernel work storage associated directly with the individual CPU.

b) Common Service Routine - There may be multiple copies or a single (re-entrant) copy to be used by all CPU's.

4.3.6.3 Components of a Shared Memory Load

The components of a shared memory load are identical to those of the unit-processor load except as follows:

a) ATEP/MMS Kernel - A separate copy of the Kernel and its data is stored for each CPU. These copies need not all use the same Kernel features and hence need not be identical.

b) User-supplied Application Dependent Executive Tables - A single copy will be used by all CPU's.

c) Inter-CPU Request Table - A memory area, accessible by all CPU's is used for passing requests from one Kernel to the Kernel in another CPU of the same computer.

d) Common Service Routines - A set of those common service routines used in each CPU is loaded for that CPU.

e) Other ATEP/MMS interrupt-state routines - Copies of these routines are included with each copy of the Kernel when the associated CPU may have a need to use that routine. For example, each CPU will have an error-processing routine.

4.3.6.4 Components of a Combined Multiprocessing Shared Memory Load

The components of this load shall be identical to that for a shared memory load, except that those CPU's using a single copy of the ATEP/MMS Kernel in multiprocessing mode, shall also share the same Inter-CPU Request Tables for communicating with other copies of the Kernel. The multiprocessing copy of the Kernel shall have a data area that is partially duplicated for each CPU in the multiprocessing group.

4.3.7 Peripheral Equipment

The ATEP/MMS Kernel fields interrupts from the peripheral equipment via the IOC and passes on requests form user modules to the IOC channel programs. However, the Kernel does not interface otherwise with any peripheral equipment.

4.3.8 Interfaces

The ATEP/MMS Kernel interfaces with application modules making up the specific application and interfaces with the Application Dependent Executive Tables. In shared memory operation, multiple copies of the Kernel program interface with each other. The Kernel also interfaces directly with the ADEP as selected for the particular application. Elements of ADEP which interface with the Kernel may include the following:

a) Initializer/loader, which loads the system.

b) Error Processors (task-state and interrupt-state).

c) Reconfiguration and reload routines.

d) Common Service Routines.

e) Common Peripheral Control programs (task-state and interrupt-state).

f) Utility programs (task-state).

g) Intercomputer message processors.

4.4 BQS-13

This Executive performs task scheduling, input/output control, interrupt handling and initialization for the AN/UYK-7 computer. It is designed to run the DNA Sonar system. Under this system several independent modules will run simultaneously.

4.4.1 Interrupt Handling

The BQS-13 treats interrupts as follows:

1) Fault and Hardware interrupts

These interrupts are generated by the AN/UYK-7 computer logic when a power tolerance or hardware fault error is detected. All interrupts are disabled, with the exception of power tolerance which is always enabled. A message is sent to the Performance Monitoring/Fault Location Program for further action.

2) Program error and clock interrupts

The standard interrupt processor is executed. Execution is continued at this point.

3) Input/Output Interrupts

These interrupts are generated by the AN/UYK-7 Input/Output Controller, and submitted to the running program. A Class 3 interrupt locks out all other Class 3 interrupts until processing is completed.

4.4.2 Initialization

This module has responsibility for initializing the system and allows for an orderly restart should a transient hardware failure be detected.

4.4.3 Task Scheduling

This function determines all job scheduling. It is executed whenever a Class 3 interrupt has been processed, or a program terminates. Scheduling is strictly on a priority basis. If no programs are ready for execution, background tasks are executed. This continues until an interrupt continues and a higher priority program is ready for execution.

4.4.4 Input/Output Control

This module handles all I/O requests. When an I/O request is received, this function examines the operational status of the required I/O channel. If it is not busy, the I/O request is executed and the channel set busy. If the channel is busy, the request is chained to a list for later examination.

4.5 The TARTAR System

4.5.1 Overview

The TARTAR system is a fire control system aboard a ship which is responsible for finding a target, as well as firing a missile when a target comes within range. The TARTAR system is currently implemented on DLGN-38 type ships.

The system searches for aircraft using a search radar. If it finds one it will track the aircraft using a tracking radar. If a missile is to be fired, it will calculate an intercept point, and it will also illuminate the target to let the missile home in on it.

4.5.2 Hardware

The fire control system consists of a MK74 mod 5 computer. The weapon direction system is a MK13 mod 0 computer. The gun fire control system consists of a MK86 computer with an SPG-60 radar. A typical system is shown in Figure 6 (pg 1.10).

4.5.3 Executive

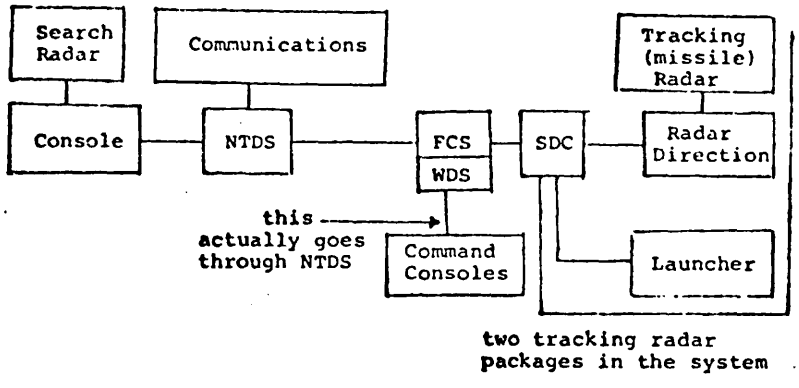
The TARTAR system uses a feedback control loop to track an aircraft. This loop must be executed exactly once every 32nd of a second (31.25 msec). The frequency 32 times/sec was chosen because 32 is a power of 2, but the periodic scheduling of the loop must be extremely precise due to the interface requirements with the radar tracking hardware.

The executive is very simple. There are two kinds of tasks, the time-critical periodic tasks associated with the control loop, and time-non-critical background jobs. One such background job is controlling the command consoles.

The periodic tasks are run in a fixed sequence, so the executive has no problem figuring out what to run. Each task has a maximum allowable run-time. The executive sets the clock to generate an interrupt when it is time to run the next periodic task. These tasks are run in the machine's executive state, on the UYK-7, and they run until completion.

When the task has finished, it calls the background scheduler, which picks a background job to run. The background task runs, in the task state, until the time for the next periodic task is up and the executive takes control. The background scheduler is not part of the executive.

FIGURE: The TARTAR System



In the TARTAR System, the sequence of operation of time-critical periodic tasks is explicitly defined in advance. Thus, there is never any queuing delay since no two periodic tasks will ever be "runnable" at the same time. This eliminates a considerable amount of complexity from the executive while, at the same time, guaranteeing precise periodicity with minimal executive overhead. These benefits are gained at the expense of considerable manual effort to determine the precise timing sequences (and occasional redesign in order to guarantee that no schedule conflicts occur).

If a time-critical task doesn't finish in its time quantum, the executive notes this in a table and will generally give the task additional time.

Each task handles its own error processing. The executive merely provides switching back to the task in the case of a program error.

Each task handles its own I/O -- it is not one of the executive's functions. I/O is controlled by the IOC through the execution of a channel program. I/O interrupts are not generated at completion, but each task performing I/O will poll the device each time it runs.

I/O to the radars is performed every 1/32 seconds and consoles are refreshed every 1/16 second. Data is transferred between the FCS and each radar at a rate of about 30-40 words (36 bit words)/32nd of a second.

4.6 Summary

Special executives (ATEP/MAX and ATEP/MMS) had to be designed to manage the substantial computer loads because the standard executives could not provide the necessary support. The lack of development aids for software generation is a problem in shipboard applications, which was resolved to some extent by ATEP/MMS.

CHAPTER 5

COMMUNICATIONS SYSTEMS USED IN THE NAVY

5.1 Introduction

The Navy has several advanced communications facilities some currently existing and some under development, which combine to form a world wide communications network. This network is capable of sending ship-to-ship, ship-to-shore, and shore-to-ship messages. The newer Navy systems are being developed to take full advantage of current computer technology and satellite links.

A brief overview of the Navy's network is as follows:

AUTODIN is the world-wide switching system. A message can be switched through AUTODIN to appropriate Navy bases.

NAVCOMPARS receives the message at the base and prepares it for transmittal using either conventional systems or CUDIXS, a new satellite system. CUDIXS is NAVCOMPARS's satellite link; there are several ordinary (HF, etc.) communications systems also attached.

Aboard ship, NAVMACS receives and processes incoming messages. It also transmits shipboard messages to other units, including ship and ground receivers.

This section provides an overview of the communications facilities used in the Navy. Specifically we discuss:

a. NTDS the standard Navy tactical data links.
b. CUDIXS, the Common User Digital Information Exchange System.
c. NAVMACS, primarily meant for automating message handling.

d. COS/UYK-20, the communications oriented executive for the UYK-20.

5.2 NTDS

The Navy Tactical Data System (NTDS) is a command and control system that has been implemented on about sixty ships. The system is primarily used as a combat direction system in war time, the communications links being via terminals between NTDS units.

Prior to NTDS (before 1961) all tactical information in a combat situation was transferred via voice. It also provides a method of communication to non-NTDS ships (via links such as L-14) and also to some fire control systems such as the MK-IV. It also acts as a two-way link between aircraft and the NTDS ship.

A typical system consists of three (sometimes four) UNIVAC 642 (A's or B's) sharing 256K of extended memory. Each UNIVAC 642B has a CPU, 32K of 30 bit words and 16 I/O channels. Each computer interfaces with other systems and also with special purpose computers, displays, fire control, and radar tracking. A common data base residing in the shared memory can be accessed by all three computers. The common part of the executive routine resides in the shared memory and the specialized parts in the various computers. The executive routine is used mainly for intertask message passing and scheduling.

In this system, three processors are used to give the required capacity. Even with a faster processor, however, several CPU's would be a useful configuration because of the large number of tasks run on the computer complex.

The system can exist in several versions having full and partial capabilities. In case of hardware failure, the system can continue to run in a degraded manner using only two computers. Doing this, however, involves reloading and relocating programs,

because I/O channels are connected to specific devices. Hardware maintenance is done on board ship.

In case of software failure, the system is restarted. If this doesn't work, a description of the problem is sent to FCDSSA which maintains the software. FCDSSA sends back a software "patch," or provides on board assistance.

On the present system, the interconnected computers do not share common memory. The computers in the system had identical copies of the executive routine and maintained identical copies of the data base. This used memory inefficiently and introduced additional complexities in system coordination.

Recently AN/UYK-7 computers have been used in place of UNIVAC-642 computers in NTDS systems. To take advantage of the increased capabilities available in the new machine, a new methodology is needed for implementing executive functions. A future change in methodology and in the application programs is needed in order to take full advantage of the AN/UYK-7's capabilities.

One of the requirements that made the use of the UYK-7 necessary was the growth of functions that NTDS was being required to support, and the number of carriers that NTDS was being implemented on. Because NTDS is a very real-time system, high I/O rates virtual memory, more core memory and high precision fixed point arithmetic are some of the features that NTDS needs.

The ITAWDS system on the LHA is an example of a system that is using UYK-7's to support NTDS. The executive being used for this system is discussed under a separate section.

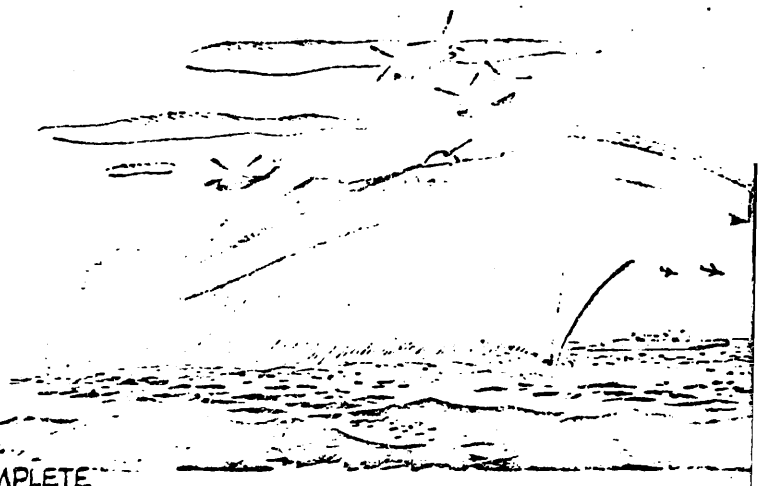
5.2.1 The LHA System

The Integrated Tactical Amphibious Warfare Data System (ITAWDS) is a tactical system for an amphibious assault ship, LHA. (See Figure 5.1) The executive which is used is called EXOS.

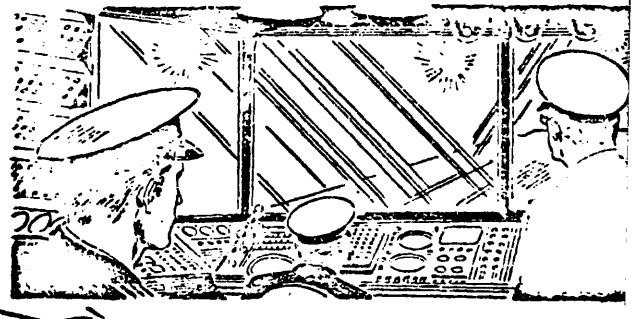
FIGURE 5.1

OUR NEW AGE BEACHBUSTER

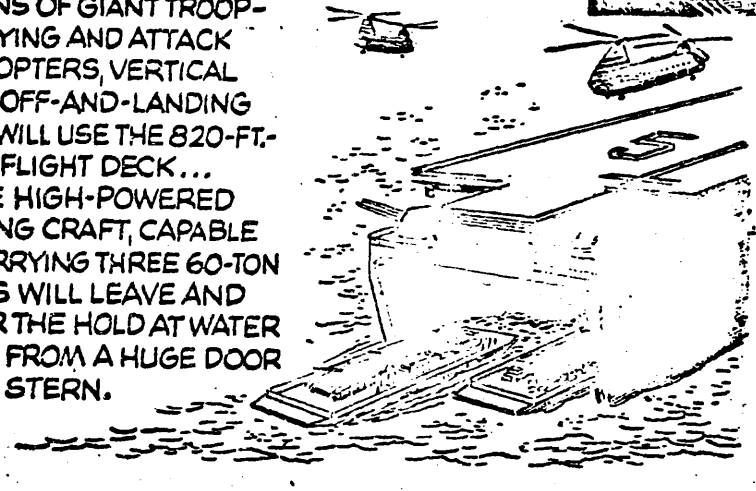
THE LHA (Landing Helicopter Assault) A MIGHTY ATTACK SHIP—UNLIKE ANY OTHER—IS NOW UNDERGOING SEA TRIALS. DESIGNED TO BE THE NEW BACKBONE OF U.S. AMPHIBIOUS FORCES, EACH COULD PUT AN ENTIRE ARMORED BATTALION OF 1800 MEN WITH TANKS AND OTHER NECESSARY EQUIPMENT AND SUPPLIES ON AN ENEMY SHORE—ALL IN ONE LIGHTNING OPERATION.



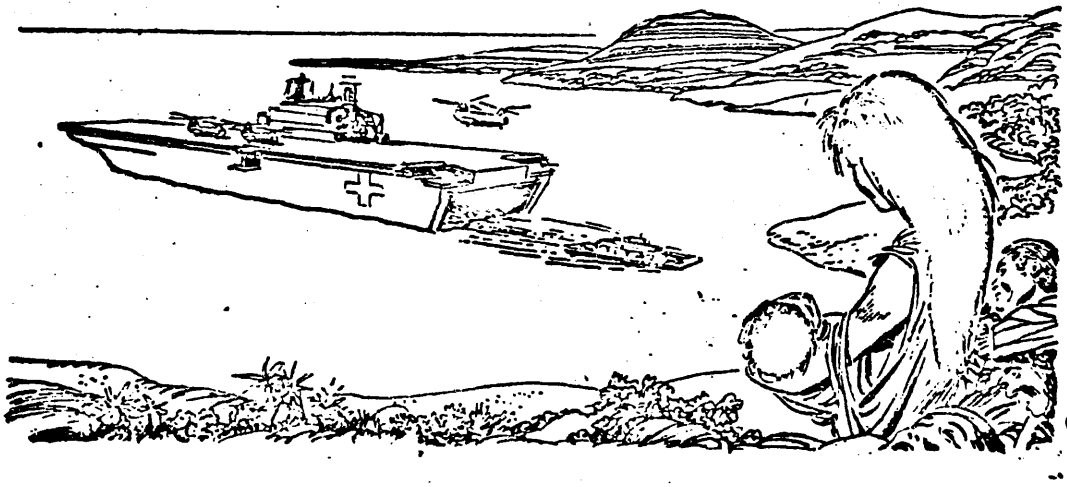
THE SUPERSTRUCTURE HOUSES A COMPLETE COMPUTERIZED COMBAT CONTROL AND COMMUNICATION CENTER OVERLOOKING BATTERIES OF 5-IN. GUNS, MACHINE GUNS AND ROCKET LAUNCHERS FIRING DEADLY HOMING MISSILES.



DOZENS OF GIANT TROOP-CARRYING AND ATTACK HELICOPTERS, VERTICAL TAKE-OFF-AND-LANDING JETS WILL USE THE 820-FT.-LONG FLIGHT DECK... WHILE HIGH-POWERED LANDING CRAFT, CAPABLE OF CARRYING THREE 60-TON TANKS WILL LEAVE AND ENTER THE HOLD AT WATER LEVEL FROM A HUGE DOOR IN THE STERN.



BUT WITH A 300-BED SICKBA LARGER THAN MANY CITY HOSPITALS, TWO X-RAY ROOM FOUR SURGICAL AND THREE DENTAL OPERATING ROOMS, A BLOOD BANK AND COMPLETE PHARMACY, THIS SHIP OF WA CAN ALSO BE A SHIP OF MER WHEN QUAKES, STORMS, FLOODS OR FAMINE STRIKE LHA COULD MEAN— LANDING HELP ANYWHER



Courtesy of the Boston

The system consists of two UYK-7's. One is solely dedicated to support NTDS and the other partially serves NTDS and partially supports the data management function. The LHA executive was developed to use the full capabilities of the UYK-7 computer for NTDS. This was necessary due to the growth of functions that NTDS has been supporting since its inception. The increased number of ships using NTDS also influenced the design. NTDS is a very real time system, and as such needs high I/O rates, virtual memory, more core memory, and high precision arithmetic. The ITAWDS is an attempt to use an UYK-7 instead of the usual CP-642B to support these needs.

EXOS is the LHA executive. It performs the basic system functions of system loading, initialization, termination, scheduling, central I/O control, inter-module communications, and interrupt processing. It also supports certain auxilliary functions for systems operations, such as disk allocation and conversion routines.

Some of the problems that systems programmers have been having with the LHA executive are as follows:

*Very slow throughput because there is not enough core, and thrashing often occurs.

*There is no deadlock avoidance built into it.

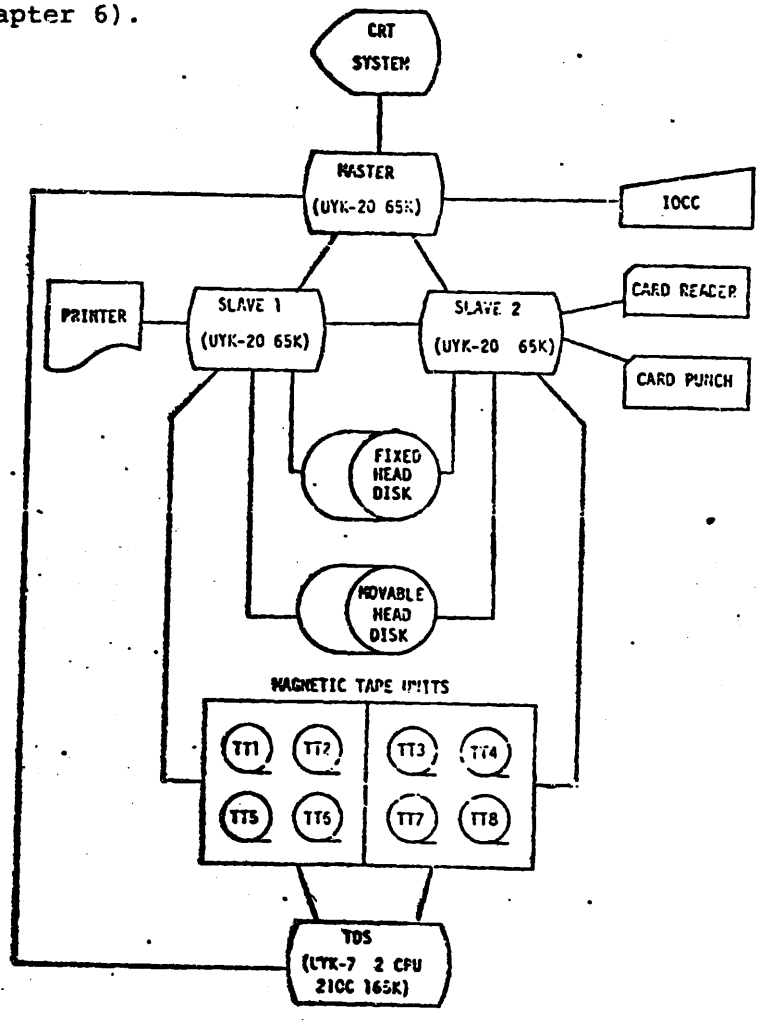
*There is no centralized peripheral management.

*There is little casualty recovery. If a file is in the middle of a data base update during a fault, they must return to a checkpoint. Periodic snapshots of core are taken at the rate of about one per minute.

The LHA system is in need of a fast-access mass storage device so that 35K of the executive need not be kept in core at all times.

There have been many problems with using the UYK-7 for the Tactical Information Processing (TIPS) part of the system (for example, there is insufficient core and the executive is unsuitable for this kind of application). Hence consideration has been given to developing a Tactical information Processing System (TIPS) from a network of UYK-20's. The proposed configuration consists of a master UYK-20 and two slave UYK-20's with the TIPS operating system performing the tasks of resource and process management.

The following diagrams provide an overview of the proposed TIPS system: (Note: TIPS is an example of a typical DBMS capability required by the Navy. This is discussed further in Chapter 6).



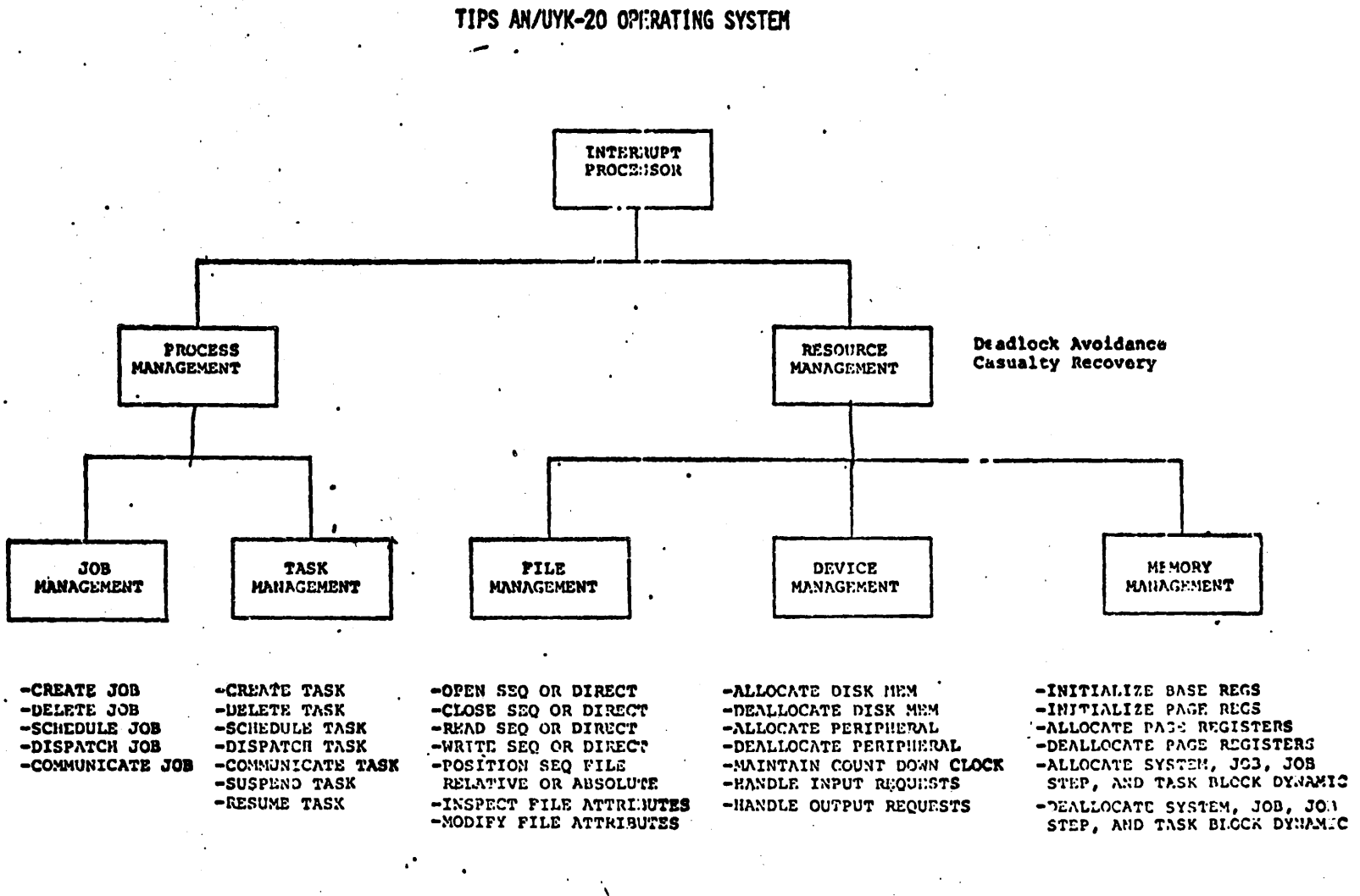

TIPS AN/UYK-20 OPERATING SYSTEM
INTERRUPT PROCESSOR
PROCESS MANAGEMENT
RESOURCE MANAGEMENT
Deadlock Avoidance
Casualty Recovery
JOB MANAGEMENT
TASK MANAGEMENT
FILE MANAGEMENT
DEVICE MANAGEMENT
MEMORY MANAGEMENT
-CREATE JOB
-DELETE JOB
-SCHEDULE JOB
-DISPATCH JOB
-COMMUNICATE JOB
-CREATE TASK
-DELETE TASK
-SCHEDULE TASK
-DISPATCH TASK
-COMMUNICATE TASK
-SUSPEND TASK
-RESUME TASK
-OPEN SEQ OR DIRECT
-CLOSE SEQ OR DIRECT
-READ SEQ OR DIRECT
-WRITE SEQ OR DIRECT
-POSITION SEQ FILE RELATIVE OR ABSOLUTE
-INSPECT FILE ATTRIBUTES
-MODIFY FILE ATTRIBUTES
-ALLOCATE DISK MEM
-DEALLOCATE DISK MEM
-ALLOCATE PERIPHERAL
-DEALLOCATE PERIPHERAL
-MAINTAIN COUNT DOWN CLOCK
-HANDLE INPUT REQUESTS
-HANDLE OUTPUT REQUESTS
-INITIALIZE BASE REGS
-INITIALIZE PAGE REGS
-ALLOCATE PAGE REGISTERS
-DEALLOCATE PAGE REGISTERS
-ALLOCATE SYSTEM, JOB, JOB STEP, AND TASK BLOCK DYNAMIC
-DEALLOCATE SYSTEM, JOB, JOB STEP, AND TASK BLOCK DYNAMIC

FIGURE 5.3
5.7

5.3 CUDIXS

CUDIXS, the Common User Digital Information Exchange System, is an advanced communications system designed as a store and forward system using a communications satellite to transmit between the shore installation and CUDIXS subscribers. CUDIXS acts as AUTODIN's link to ships at sea, working through satellite channels. AUTODIN, the Automatic Digital Information Network, is a shore-to-shore message routine system.

The system is able to support large volume traffic: ten ships, ship/shore/ship, 1200 messages/day, average message length of 300 words. Also there is support for small volume traffic: fifty primary subscribe ships, ship/shore, ten messages per day per ship; ten special subscriber ships, ship/shore/ship, 100 messages per day per ship.

High precedence traffic response is less than three minutes with a two minute cycle time.

There are currently four installations. Messages go out by precedence. Polling techniques are used. However, there are up to four random access time slots (RATS) into which high priority messages can be put. It is also possible to send "flash" messages. CUDIXS interfaces with COMPARS. COMPARS handles all message-sending circuits except satellites. COMPARS also generates statistical reports on number of messages sent, as well as, precedence of messages, etc.

There is also an interface with AUTODIN. This is a switching system for sending digital information from point to point. AUTODIN is a Defense Communications Agency system.

The main processor is an AN/UYK-20. Other hardware includes a message coder, a paper tape reader/punch, a high speed printer, teletypes, and magnetic tape. In the shore installation, there is also disk storage.

5.3.1 Executive and Other Software

Neither SDEX-20 nor COS/20 are used. A special executive was written for this application. It is very similar to the executive NAVMACS uses (see below). Three types of tasks are scheduled: real-time, periodic, and background. Much of the I/O is done with parallel interface. The basis of the executive was founded on COMMON/MODX-2.

One operator interaction with the system is to change the cycle time. The cycle time can be cut down to 20 seconds. Another control mechanism is to have a message repeated on request.

A complete description of the system was given to us in the form of a CPPS for CUDIXS (shore subsystem).

This application illustrates the designing of a new executive for a system because the existing executives were not considered to be adequate, and also because support and maintenance for existing executives was not considered to be adequate.

The following diagrams summarize the architecture and capabilities of the CUDIXS system.

COMMON USER DIGITAL INFORMATION SYSTEM (CUDIXS)

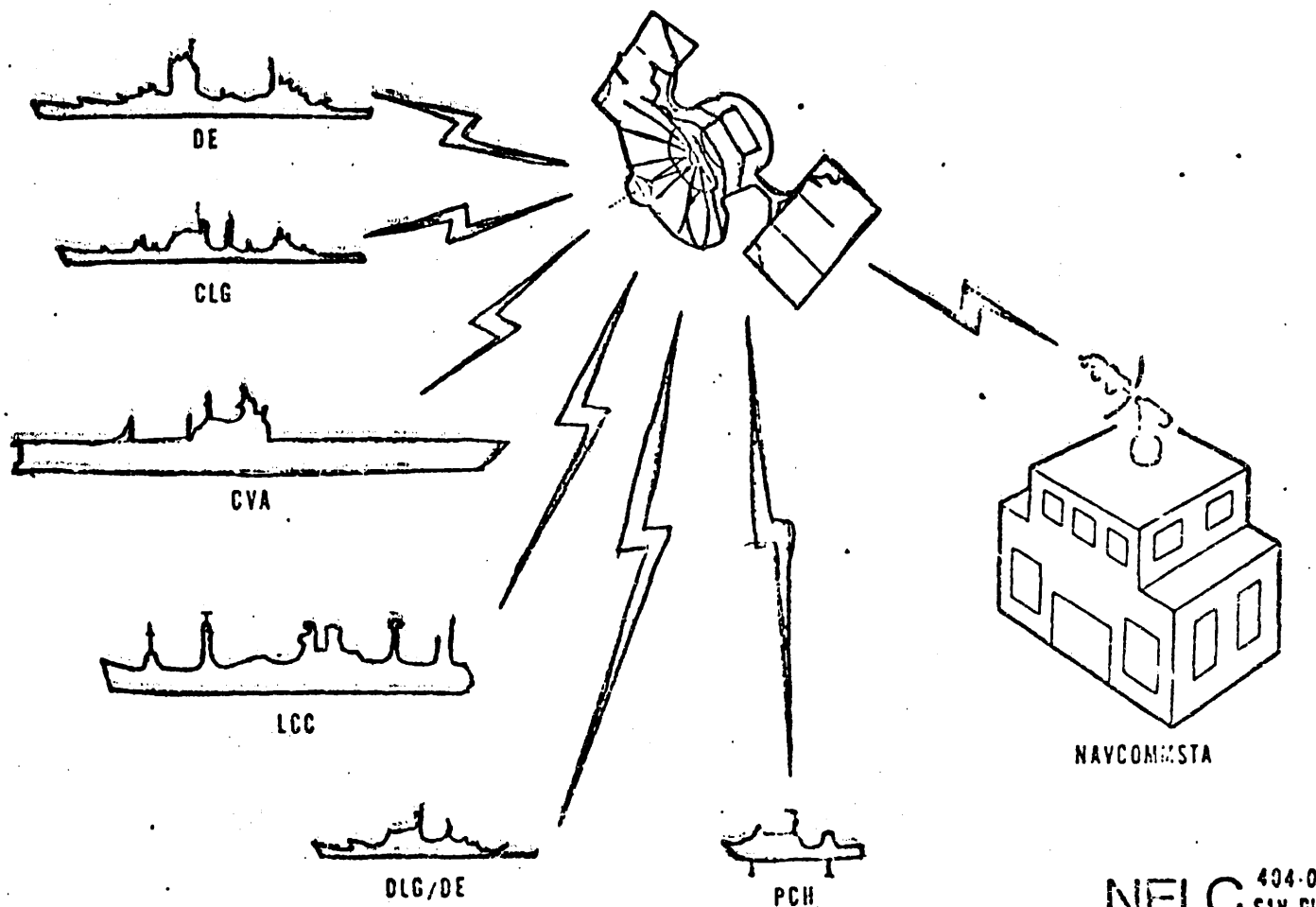


FIGURE 5.4
5.10

CUDIXS DATA LINK CONFIGURATION

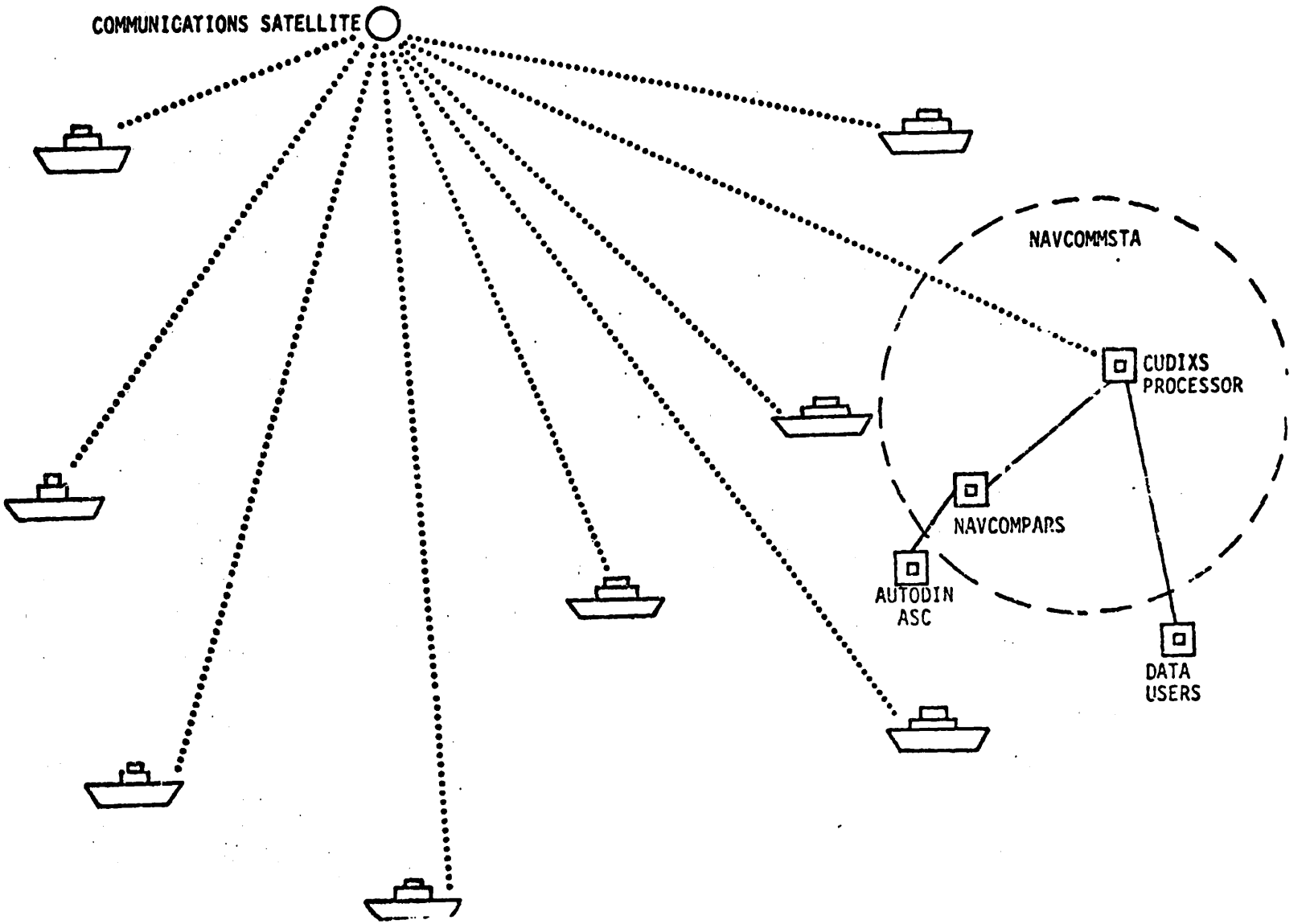


FIGURE 5-5
5.11

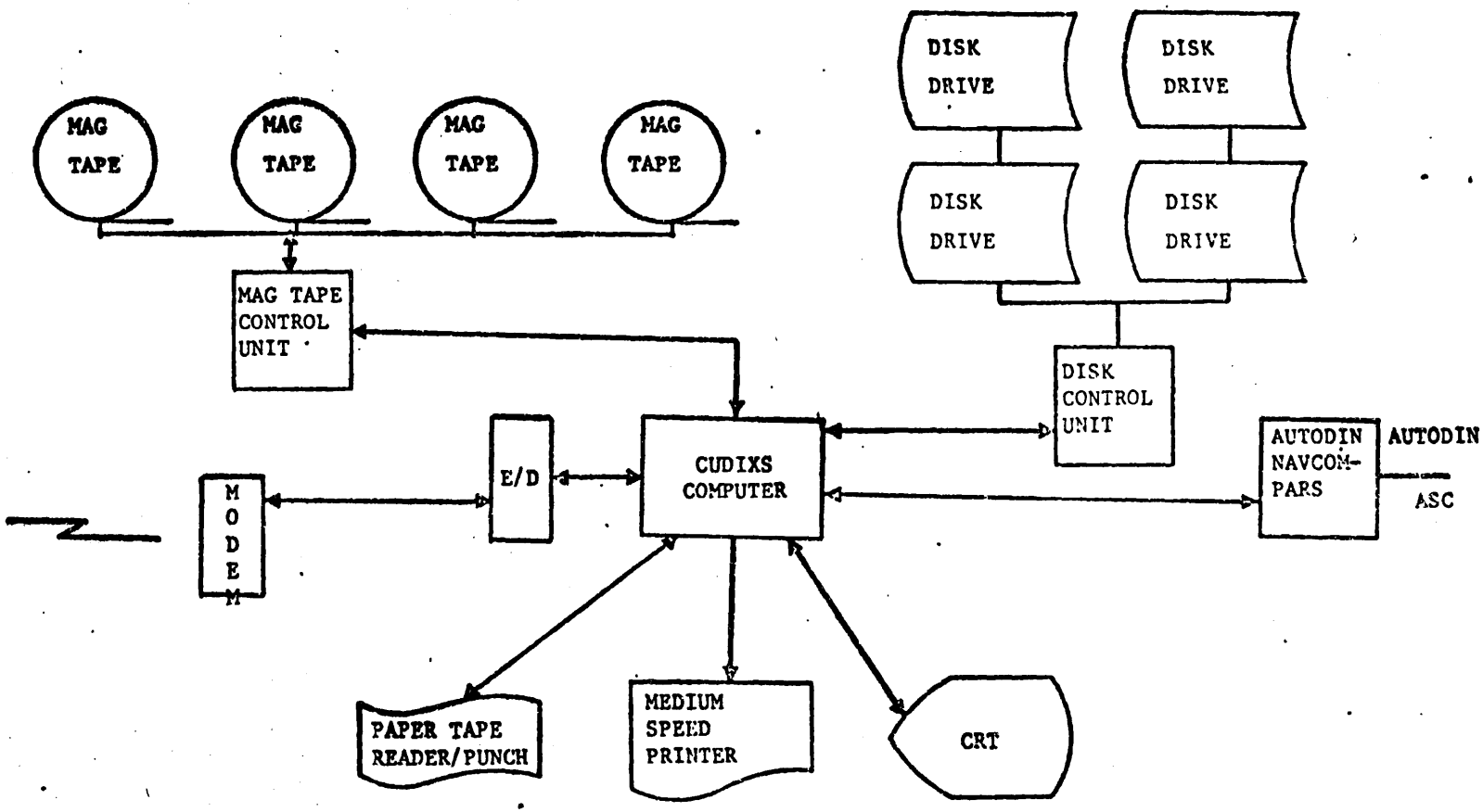
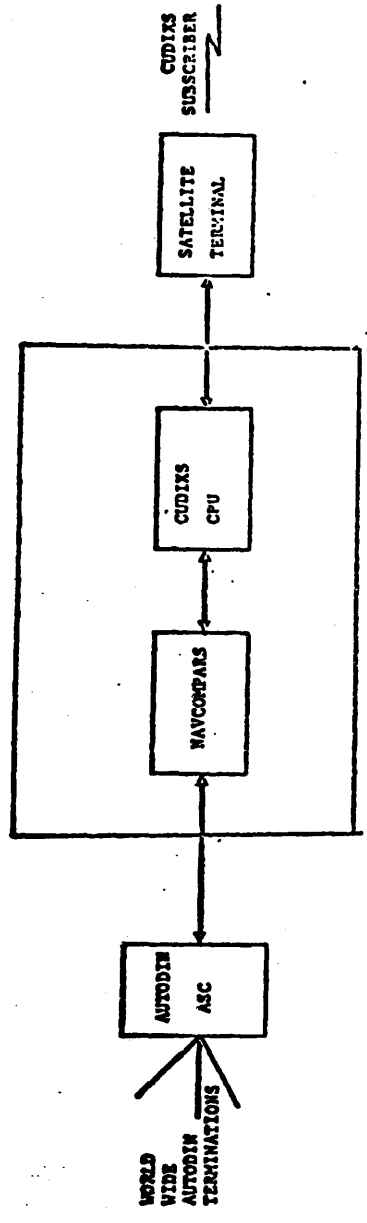


FIGURE 5.6
3.12

CUDIXS SHORE SUBSYSTEM INTERFACE BLOCK DIAGRAM

5.3.2 NAVCOMPARS/CUDIXS Interface

NAVCOMPARS, the Naval Communications Processing and Routing System, is the interface between CUDIXS and AUTODIN, the shore-to-shore message handler. It has the cspability to place AUTODIN-switched messages within CUDIXS for processing, and, conversely, to place CUDIXS incoming messages into the AUTODIN network for further routing.



CUDIXS SHORE EXTERNAL INTERFACES

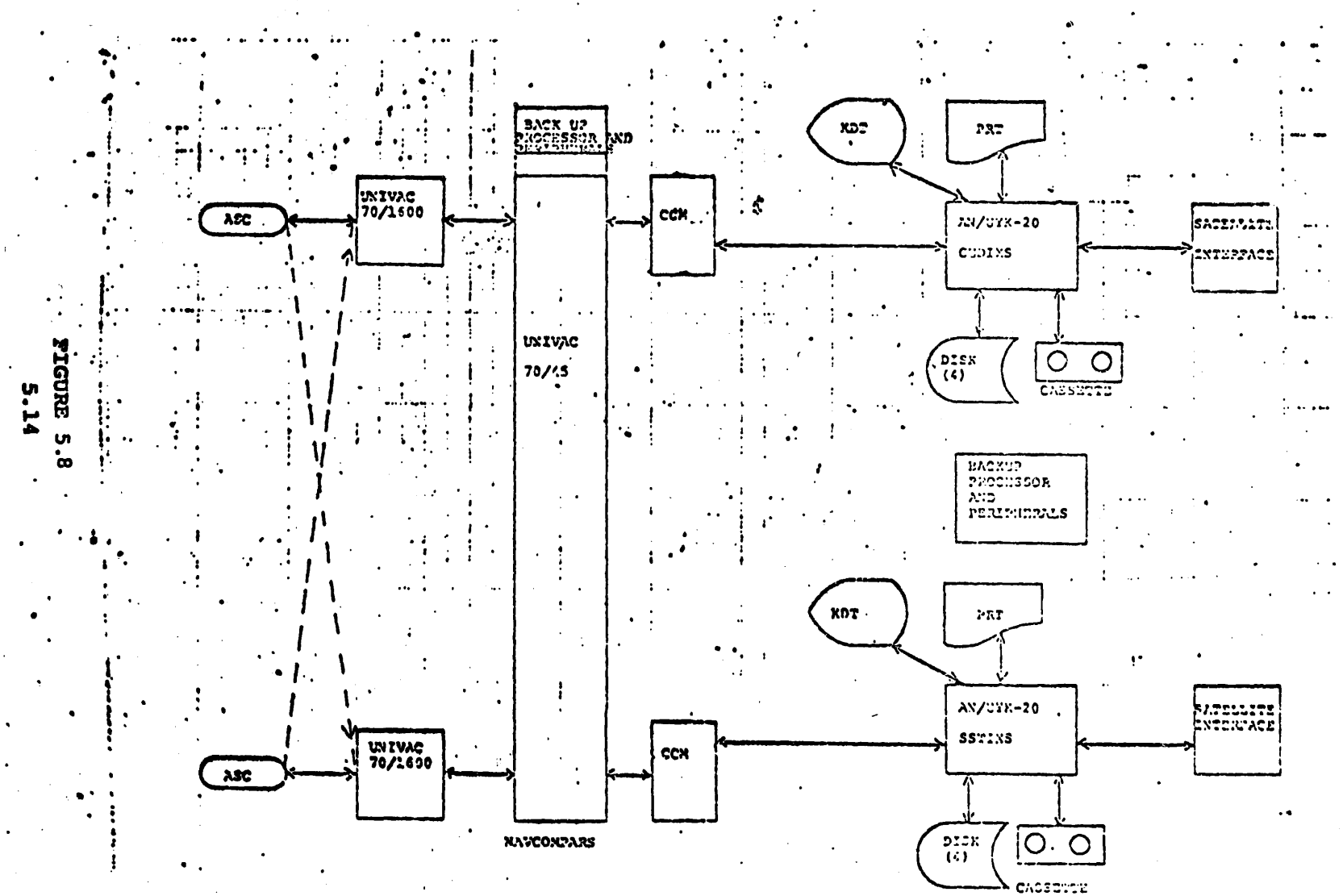


FIGURE 5.8

5.14

FIGURE 2-05. Proposed NAVCOMPARS-CUDIXS/SSTIXS Configuration

5.4 NAVMACS

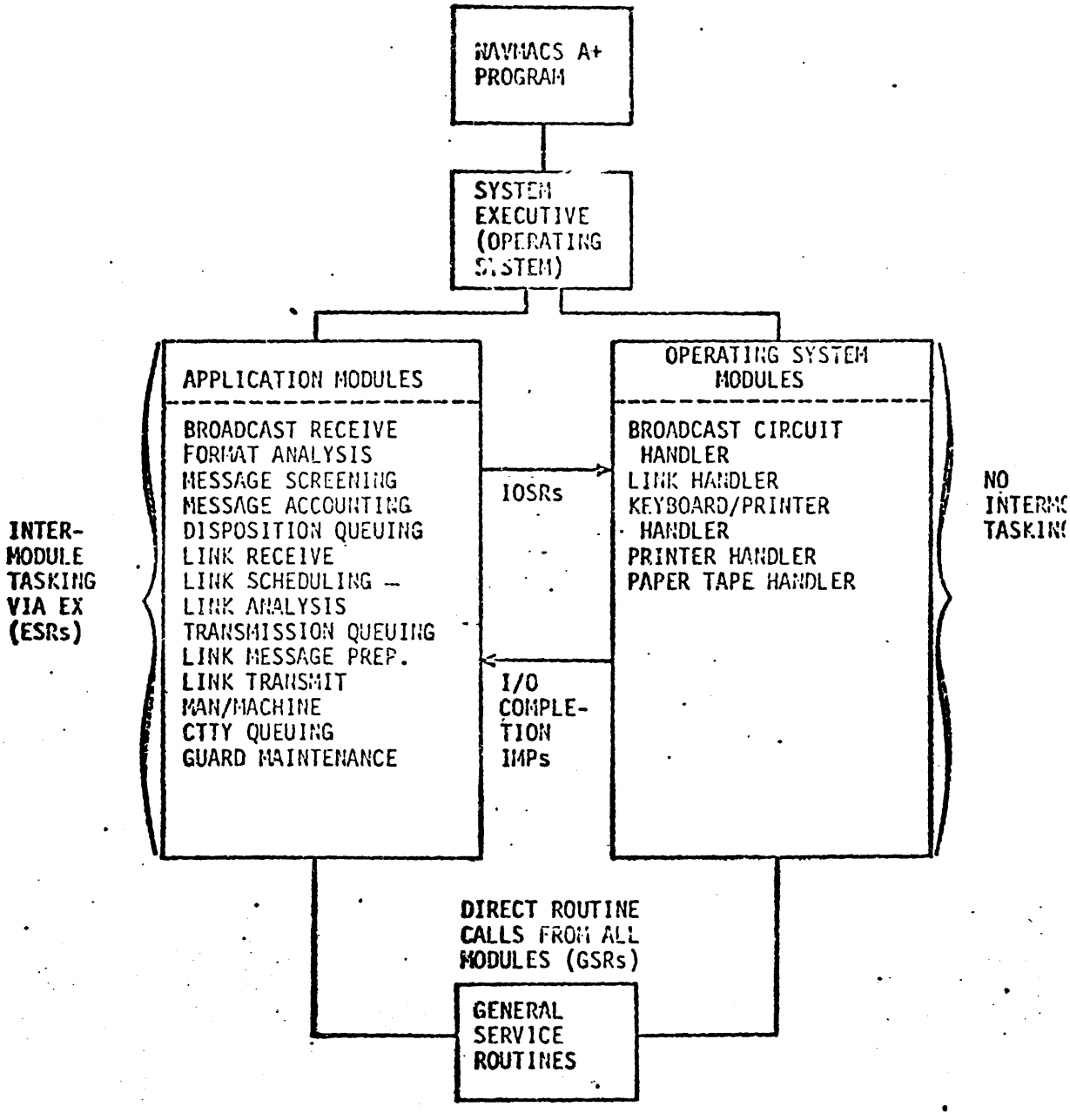
NAVMACS is a shipboard family of communication systems (A, A+, B ...) designed to automate the shipboard functions of message handling -- teletype processing of ship to shore, ship to ship and shore to ship messages. The various NAVMACS families are generally upward compatible, dependent on specific shipboard needs.

5.4.1 NAVMACS System A+

NAVMACS A+ is a core-resident system that is interrupt driven. System A+ is primarily transaction oriented -- an individual transaction is processed by a series of subprograms before transmittal. The actual sequence of subprograms used is dependent on the message type.

The executive schedules all application subprograms and controls allocation of system resources. Within memory, the system itself is fixed in place but input/output buffers and working storage is dynamically allocated as required.

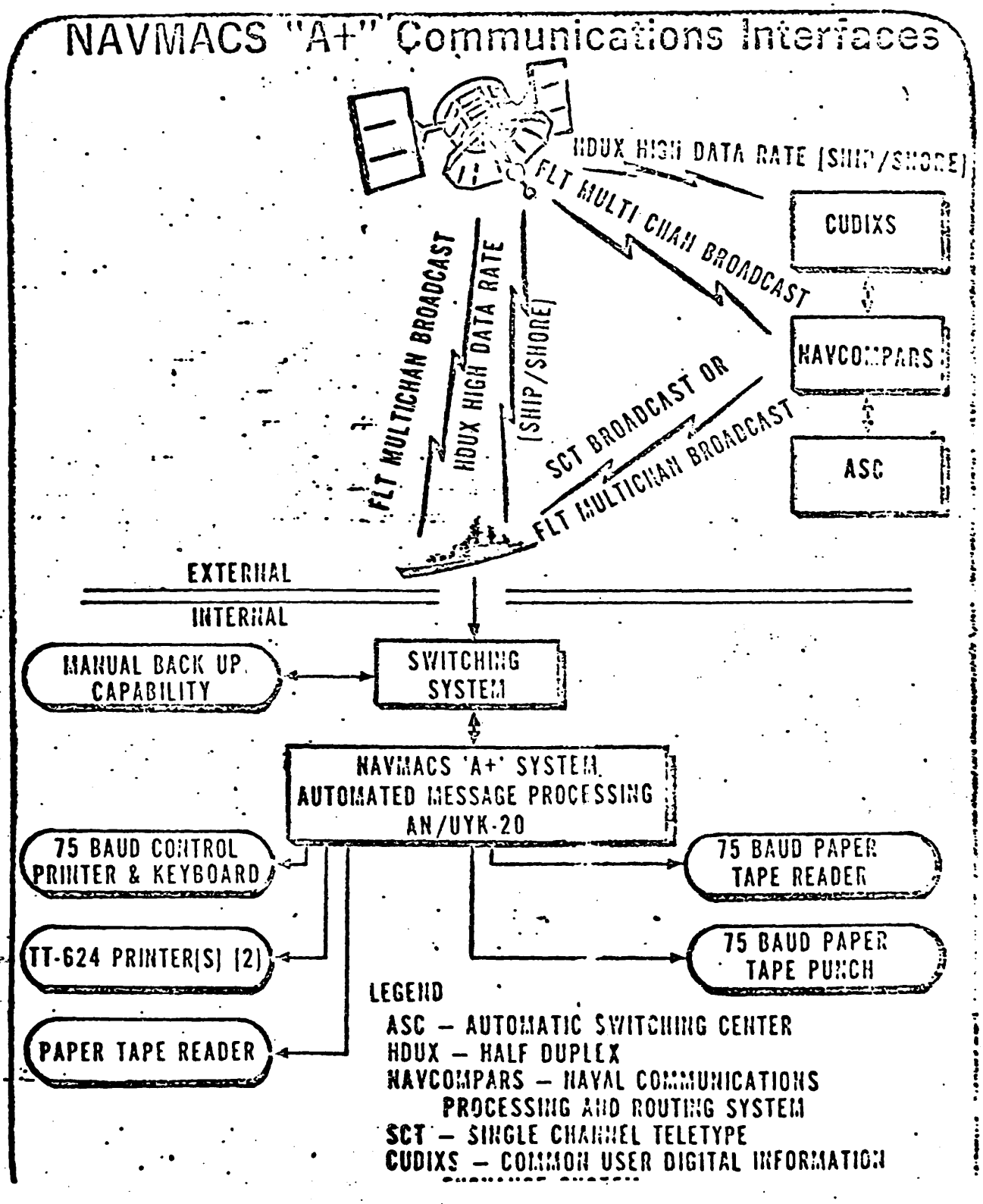
5.4.1.1 NAVMACS System A+ Control Diagram



NAVMACS A+ Program Structure

FIGURE 5.9

FIGURE 5.10


NAVMACS "A+" Communications Interfaces
HDUX HIGH DATA RATE (SHIP/SHORE)
FLT MULTI CHAN BROADCAST
CUDIXS
NAVCOMPARS
ASC
FLT MULTICHAN BROADCAST
HDUX HIGH DATA RATE
(SHIP/SHORE)
SCT BROADCAST OR
FLT MULTICHAN BROADCAST
EXTERNAL
INTERNAL
MANUAL BACK UP CAPABILITY
SWITCHING SYSTEM
NAVMACS 'A+' SYSTEM AUTOMATED MESSAGE PROCESSING AN/UYK-20
75 BAUD CONTROL PRINTER & KEYBOARD
75 BAUD PAPER TAPE READER
TT-624 PRINTER(S) (2)
75 BAUD PAPER TAPE PUNCH
PAPER TAPE READER
LEGEND
ASC – AUTOMATIC SWITCHING CENTER
HDUX – HALF DUPLEX
NAVCOMPARS – NAVAL COMMUNICATIONS PROCESSING AND ROUTING SYSTEM
SCT – SINGLE CHANNEL TELETYPE
CUDIXS – COMMON USER DIGITAL INFORMATION

5.4.2 NAVMACS System B

NAVMACS System B is a real-time automatic communications system. It is designed to automate certain manual functions that would otherwise be required during message receipt, processing, and delivery. The system is designed to support the receipt and transmittal of messages via conventional or satellite links in a real-time direct interface. The system also enhances security safeguards, provides selective message output through address screening, and assists in the recording and accounting of message handling. The system also makes provision for a fall back on a totally manual system is a major system problem occurs (e.g., power failures, or serious malfunctions).

5.4.2.1 NAVMACS B Control DIAGRAM

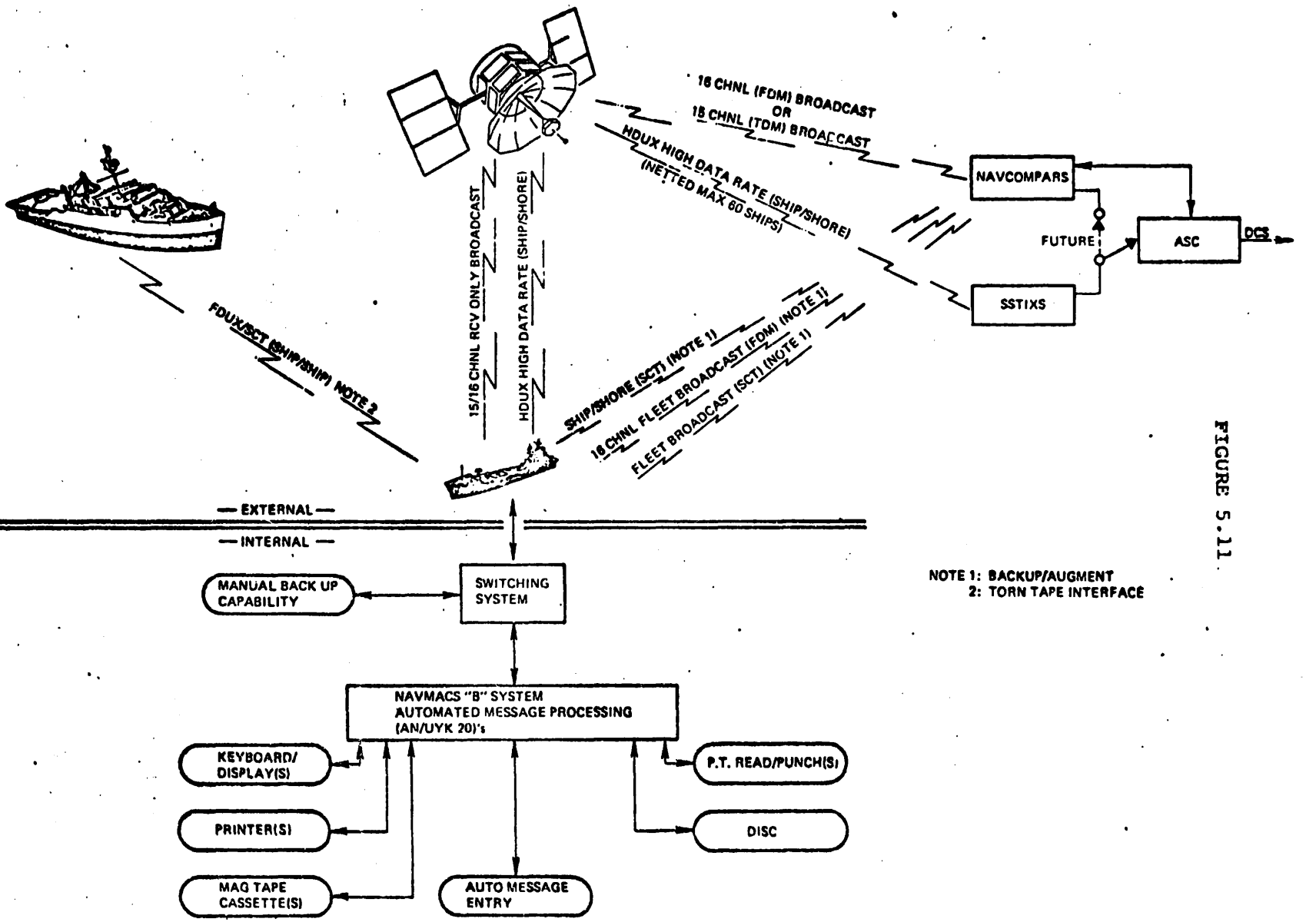


Figure 1. NAVMACS (B) Internal and External Interface

FIGURE 5.11

5.5 COS/UYK-20

5.5.1 Introduction

COS is configured for and implemented on an AN/UYK-20 computer for simple or multiprocessor configurations. The AN/UYK-20 is a 16 bit word, 32 register general purpose mini-computer, with direct and indirect addressing of 64K of memory and a 750 nano second access time. The AN/UYK-20 has channels to interface with peripherals and an intercomputer channel to communicate with the other processors. COS is the communications oriented operating system for the UYK/20 which provides I/O and processor support to program tasks operating under its control. Multiprogramming capability is provided to allow several tasks to co-reside in main memory and share processing facilities of the system.

COS provides peripheral and communications I/O handling to support the communications applications environment. A particular module can be selected at system generation for any requirement. All I/O control resides in a centralized peripheral and communications I/O functional area.

COS provides real-time scheduling support for communications applications tasks. This is in the form of multi-priority scheduling algorithm.

COS allows programs to operate in a multi-computer environment. This includes assigning tasks to computers at load time, capability to pass control information and data between tasks resident in different computers, and the capability to do I/O from peripherals not connected to that computer. These capabilities are provided so that tasks need not know in which computer it resides or which peripherals are connected.

COS provides an interactive capability for communication with the programs.

5.5.2 Devices

Typical devices interfacing with the AN/UYK-20 are:

1) operator console
2) interactive console
3) high speed printer
4) magnetic tape
5) disk
6) card reader/punch
7) paper tape reader/punch.

5.5.3 Resource Management

5.5.3.1 Interrupt Processing

There are three interrupt classes. Each class provides unique interrupt types, arranged by priority. Interrupts may not suspend a task with a higher priority. If a lower class interrupt occurs, it is scheduled and the control returned to the task. On interrupts where the task can be suspended, the interrupt is given control.

5.5.3.2 Memory Management

Allocation of main memory is performed within memory partitions, defined at system generation or initialization. Request queues are maintained so that as additional memory becomes available, it is automatically assigned on a task priority basis.

Assigned resources remain allocated until explicitly released by the user, or until program termination.

5.5.3.3 Scheduling

COS has a swapping mechanism for temporary roll out of low priority programs to make memory available to higher priority tasks. Swapped out programs are rescheduled to allow resumption of the program at the point of execution when memory becomes available.

COS has an open-ended, multi-priority scheduling and dispatching mechanism. Priorities are dynamic, changeable at run time (as opposed to SDEX-20 which has static priorities). COS supports a variable number of task queues, but initially there will be three queues

1) Event oriented tasks - tasks to be executed because of a particular event.

2) Time oriented tasks - execution begins at a particular time.

3) Priority oriented tasks - scheduled by other tasks, number of priority levels set at compile time.

Queues are searched by the dispatcher in the order: event, time (if time has come due), and then priority queue. The highest priority task within the queue selected is dispatched.

If an interrupted task is not suspendable, then it continues. A suspended task within any queue has higher priority than any other task in that queue.

The scheduler assigns an absolute system priority to a task, based upon requested priority; it is then entered into the appropriate queue.

5.5.3.4 File Management

The File Control Function of the COS/UYK-20 provides the interface to access all files in the system. It is capable of handling user or system file reference requests.

Since the COS/UYK-20 operates under multiprocessing, all file references are passed to the computer connected to the storage device (disk or drum) and are routed through a centralized I/O routine to determine appropriate peripheral device handlers.

CHAPTER 6

COMPUTER SYSTEMS FOR THE MARINE CORPS

6.0 Introduction

Studies regarding the applicability of automation to tactical command and control requirements sponsored by Headquarters, U.S. Marine Corps in the mid 1960's gave rise to the MTACCS concept, which envisions the development of automated, integrated air/ground tactical command and control systems designed to service the Marine Air-Ground Task Force. When fielded, the component systems will provide a full spectrum of automated support to aid the commander in the exercise of command and control. The primary and supporting systems of MTACCS are as follows:

6.1 MTACCS

Primary Systems

MIFASS	Marine Integrated Fire and Air Support System. It deals with coordination, command and control of fire and air support. Expected to be operational in the 1983-84 time frame.
TCO	Tactical Combat Operations--TCO supports the functions of ground and air units.
MACCS-85	Marine Air Command and Control System for 1985 and is the follow-on system to replace MACCS-70. (similar to NTDS, but for the Marines)
MIPS	Marine Integrated Personnel System. An MIS type system.
MILOGS	Marine Integrated Logistics System--MILOGS supports the combat service support element of the air-ground task force.
MAGIS	Marine Air-Ground Intelligence System.

Note: MIFASS and MACS-85 are very real-time systems, while MILOGS and MIPS are Management Information Systems (MIS) type systems requiring data management capabilities. The specific DBMS has not been selected.

Supporting Systems

COMM	Communications--COMM is the communications system existing at the time of MTACCS implementation, which is designed to serve all command and control systems.
TWAES	Tactical Warfare Analysis and Evaluation System--TWAES aids in the conduct and analysis of field exercises.
TESE	Tactical Exercise Simulator and Evaluator--TESE is used to simulate combat to provide realistic training, short of field exercises.
PLRS	Position Location Reporting System--PLRS is an air and ground navigation system which provides real-time air and ground unit locations.

6.2 MTACCS Test Bed

MTACCS (Marine Tactical Command and Control System) has a "test bed" inevironment for Command and Control systems for the Marine Corp. In the past, contractors have been asked to deliver systems to the Marine Corp with insufficient specifications. This often results in the delivery of a system that does not satisfy the military's needs precisely. MTACCS task is to evaluate the Marines' system requirements beforehand, simulate the perceived needs and then produce a set of detailed specifications for a system, usually in the form of a detailed requirements document.

6.2.1 MTACCS Environment

Although much of the actual simulation work is done on an IBM 360, the applications programs and the actual evaluation is run on a CDC 3500 at Camp Pendleton. The system has 4 CDC 604 tape drivers, 4 CDC 844 disks (similar to IBM 3330's), and a card punch and line printer. It has 256K of core of which 192K is left for applications programs and 64K for the executive.

The executive is MASTER which comes with the CDC machine. A data base management system called MARS (made by CDC) is also available.

The CDC 3500 is connected to a PDP-8 which is used as an interface to 9 Delta CRT displays and CDC "GRID" displays. The CRT's are used to display textual information while the GRIDs use a minicomputer for displaying graphic tactical information in, for example, war games, simulating the battlefield, and firing guns. A graf pen can also be used for plotting a battlefield by drawing the outline over a map. Hard copy output is available, and a CALCOMP plotter can reproduce the output on the GRID screen.

Programming is done in COMPASS (the CDC assembly language) and in FORTRAN.

6.3 MIFASS

The Marine Integrated Fire and Air Support System, MIFASS, deals with the coordination and command and control of fire and air support. It is expected to be in operational readiness by 1984, and is currently under development. It is designed to enable a ground commander to have more effective use of supporting weapons.

MIFASS equipment and design exploits modern technology in such areas as real-time tactical information and advanced digital communications. Equipment is designed to be small, lightweight, rugged and reliable. The system is designed to be a significant improvement over currently available systems in all applicable areas.

Personnel using MIFASS will not be data processors; they will be such operational personnel as infantrymen, artillerymen, pilots, etc. Utilizing the small and lightweight input/output devices currently under development, (and presumably operational by the 1980 time frame) MIFASS's information management capabilities will be made available to the average user. MIFASS will perform calculations and store, process, and give out any information requested on the current tactical situation.

The basic idea of MIFASS is to provide each operator with an automated graphical display he considers important to the current tactical situation. It allows him to recommend decisions, and then implement these decisions. The system is designed to be flexible and can be employed in several ways.

6.3.1 MIFASS Requirements

MIFASS cannot operate in isolation and assumes that several other systems currently under development are available in one form or another. MIFASS must be supported by a reliable and accurate digital communications network within the landing force. It is expected that planned 1980's capabilities will satisfy MIFASS requirements.

Also required by MIFASS is an automatic position location system for friendly ground and air units. MIFASS will also be able to display this information.

6.3.2 MIFASS Advantages

MIFASS has many advantages over the current manual system. These would include:

- More responsive to needs of infantry commands.
- More coordination between ground fires, air missions, and ground manuevers.
- More safety through more efficient and effective mission clearance procedures.
- Faster and more flexible fire support planning and execution.
- More reliable support during heavy activity.

6.4 TESE

TESE, the Tactical Exercise Simulator and Evaluator, is an on-line, interactive war game simulation. The computer simulates the entire battlefield--air forces, anti-aircraft defences, ship movements, logistics, ground forces, communications, etc. The simulated battle can be up to 3-4 days in length and the data base is updated every 30 seconds.

The TESE controller knows all of the facts about the simulation and can influence the mock battle. He can pass information from one side to another and can move aggressive units from place to place.

6.4.1 TESE Environment

TESE runs on a 360/65. It was written in PL/1 and was developed in six months. It is also being developed for the AYK-7 with graphics display capability.

The TESE command and control system is hardware independent. Most previous command and control systems had been very hardware dependent, and therefore not easily exportable to other systems.

6.4.2 TESE Executive

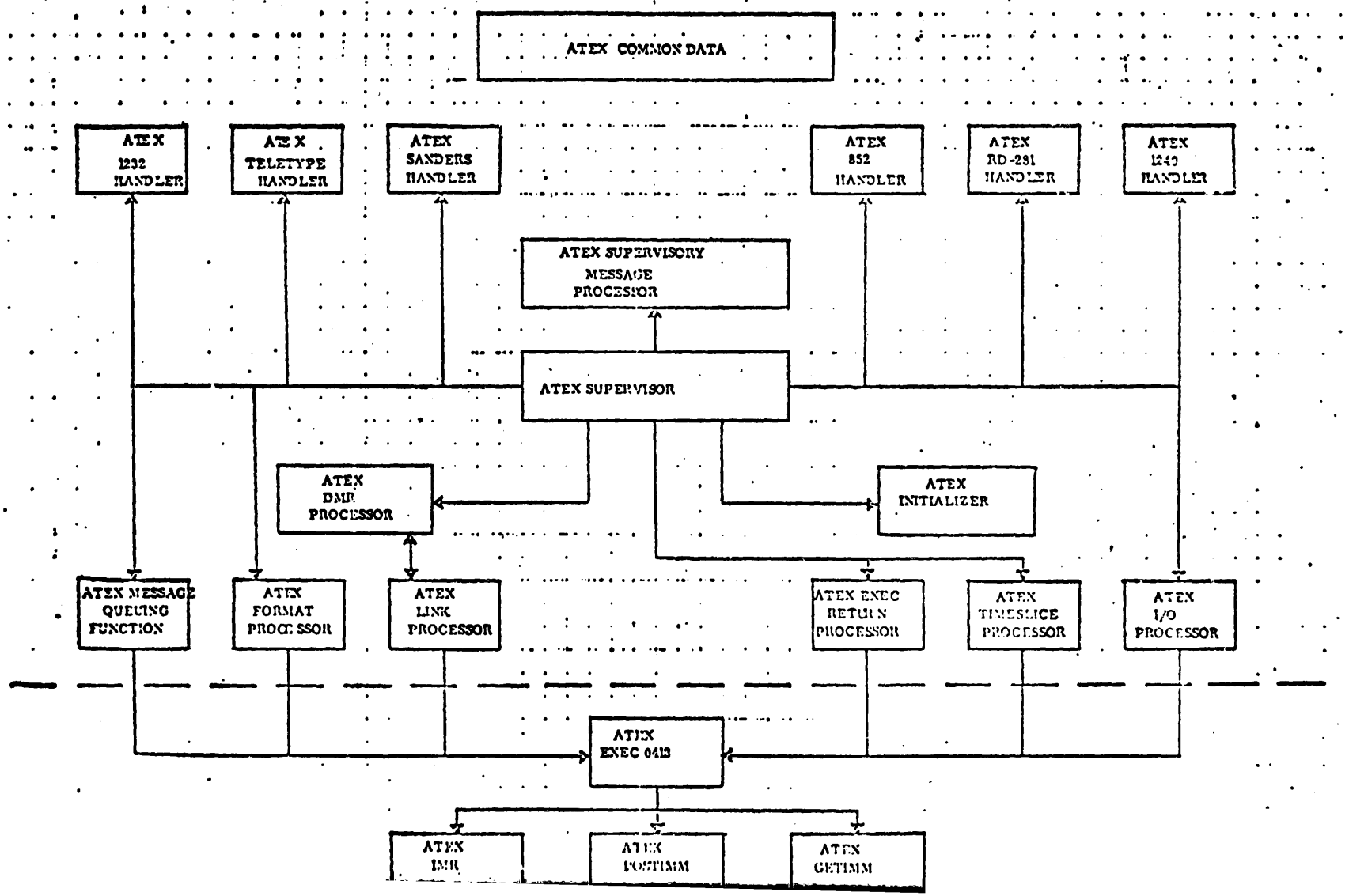
ATEX, a very highly modified version of the TWAES Executive, is used as the Executive for TESE. The TWAES Executive was unsuitable in several respects--for example, it has no distinction between task and executive states and there is no logical file handling capability.

TESE added a logical file system while retaining the multiprocessing and dynamic module replacement of the TWAES Executive. The TWAES scheduler was also retained.

This is yet another example of a system where an existing O.S. was not used but was modified before usage.

6.4.3 ATEX Control Diagram

6.4.3 - ATEX Executive Program Control Structure



6.5 TWAES

TWAES, the Tactical Warfare Analysis Evaluation System is a field-based computer controlled war-game system. Two teams compete, with umpires feeding in status information. It is designed to provide a method for effectively evaluating operations from data received in 'near real time' environment. Exercise data is analyzed by TWAES and as a result an exercise can be analyzed and evaluated as the operation progresses.

TWAES can also 'replay' segments of an exercise, even while the exercise is in progress. History data files are generated automatically during a battle, and from these a complete exercise can be reconstructed.

6.5.1 Pre-Exercise Mode

In pre-exercise mode, TWAES performs certain initializations and system startup. This mode must be performed before any exercise utilizing analysis or evaluation.

6.5.2 TWAES Exercise Mode

In this mode TWAES functions as a 'near real time' system. Exercise data is entered through message reports from field umpires and observers via the Digital Message Entry Device (DMED). Validation is performed and the data is processed for CRT display. At this time input data is recorded for use in poast-exercise operations.

6.5.3 Post-Exercise Mode

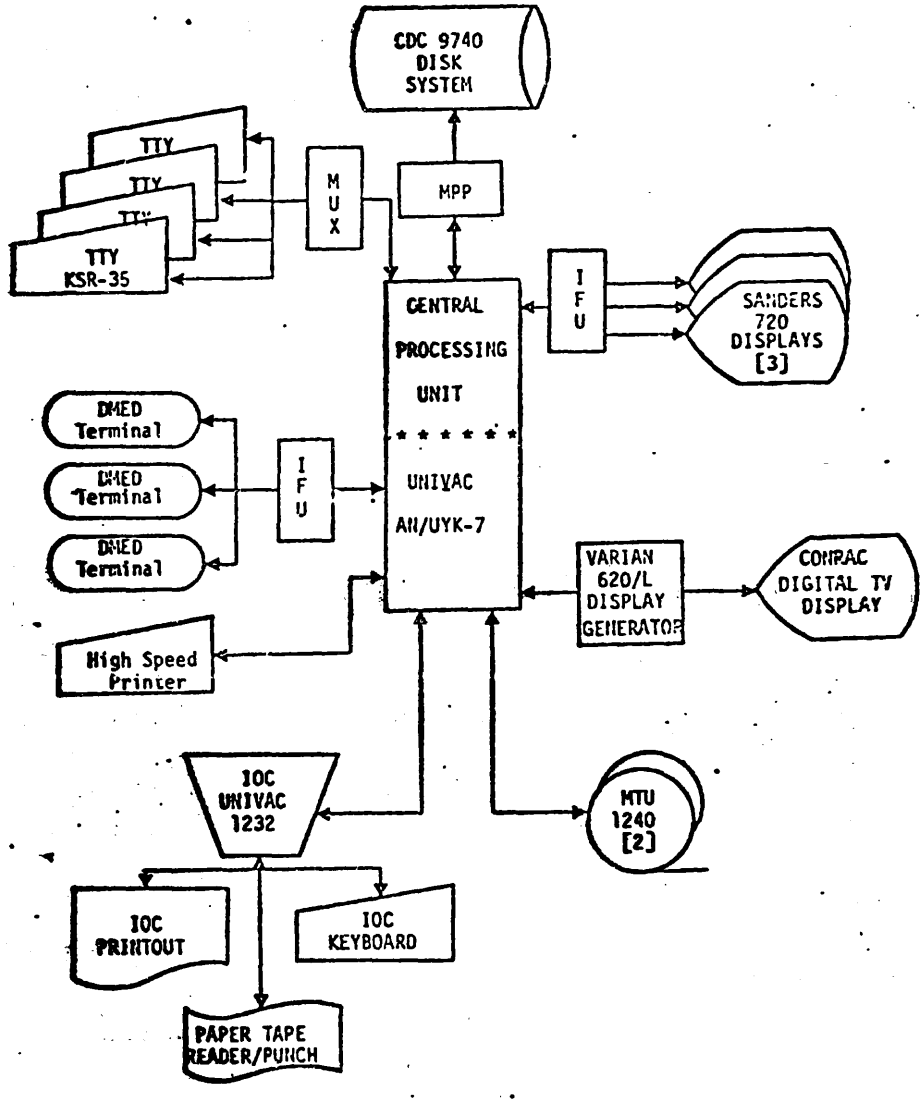
In this mode exercise data is obtained from history file(s), and reconstructed for analysis and evaluation. The overall exercise may be observed, and flaws and faults in an operation may be detected.

6.5.4 TWAES Operating System

The TWAES Operating System, named TOS, is a modular Executive designed exclusively for TWAES use on the UYK-7. TOS modules may be dynamically configured to meet widely varying

requirements. TOS includes certain common data pools, common routines, peripheral handlers, the Executive, and interrupt and error processors.

6.5.5 TWAES I Hardware Configuration



6.6 Other Marine Systems

Other supporting Marine Systems are as follows:

6.6.1 Tactical Combat Operations

The Tactical Combat Operations system (TCO) supports the functions of both ground and air units during combat.

6.6.2 MACCS-85

The Marine Air Command and Control System for 1985 is the follow up system for MACCS-70. It is a very real-time C&C system.

6.6.3 MIPS

The Marine Integrated Personnel System (MIPS) is a management information system (MIS) requiring database management facilities. The particular DBMS has not yet been selected.

6.6.4 MILOGS

The Marine Integrated Logistics System is another MIS system designed to support the supplies needed for an air-ground task force. The particular DBMS has not yet been selected.

6.6.5 PLRS

The Position Location Reporting System, PLRS, is designed for use in an air and ground navigation system providing real-time air and ground locations for tactical use.

6.7 System Requirements

It must be noted that many of the proposed systems that the Marine Corps will be developing require a data base management facility. Many of the Marines' applications that the authors have encountered have the following characteristics:

- The applications require a facility for storing large quantities of various types of data. This includes numerical data as well as non-numeric data, such as character values for storing names.

- Data must be selected and accessed according to varying criteria. It must be easy to input and update data.

- Validation mechanisms for the data are important. Security mechanisms are especially important. Moreover, the user should have implicit assurance in the data integrity mechanisms and the security mechanisms While he should not have to be burdened with the details of these systems, he must have confidence that adequate mechanisms exist.

In addition, military applications impose the following considerations:

- Several classes of users, each of which has a different degree of sophistication.

- Complex and changing security requirements.

- Data that exhibits complex and changing interrelationships.

- Changing needs to be met by the information system.

- Need for quick and inexpensive implementation. This is especially true in the military where it is often difficult to specify exactly what the requirements of the application will be. It is more advisable to bring up prototypes on a flexible data base management system. The application can then be evaluated and modifications made as necessary.

An example of a data base management application that created difficulties because of changing perceptions of users needs was the Integrated Flagship Data System or IFDS, developed in the late 60's. Although it is a potentially useful system, it could not change to conform to a change in its perceived use and hence, the effort was scrapped. However, the application is representative of military DBMS applications and is described as follows:

6.7.1 IFDS

The objective of MPIFDS effort (started in the mid 60's) was to design and then develop a fleet flagship data system to serve fleet commanders during the 1970-1980 time frame. It was to be a command and control system as well as an intelligence system.

The command and control functions include logistics, planning, communications control and administrative tasks. The intelligence functions include processing general area intelligence, determining the location and status of enemy forces, communicating orders of battle and recording weapons, equipment and platform characteristics. Information was gathered from many sources such as from ships, sattelites, casualty reports and on more facilities but the ship does most of the updates on the data base.

In many ways IFDS had to support standard MIS type functions. For example, a typical query would be determining the last 24 hour casualty information. There are some requirements that make it different from a conventional MIS. The necessary response time is very quick (less than 15 seconds) especially in a wartime tactical situation. Moreover, "bad" or inaccurate information can have a very negative effect.

The IFDS data base management system was written in a language(NELIAC) similar to ALGOL 60. Most of the development phase of the system (three years) was spent in designing and developing this data base management system. A separate language was used for validating data (it was not incorporated in the DDL). The data base system was interfaced with graphics and alphanumeric displays. The DBMS had a good english like DML. For example, a query for finding the names of ships with SQS-23 sonar would be:

```
IF SHIPTYPE   = DD   (Destroyer)
And EQUIPNAME = SQS-23
SAVE VALUE A  = UNITNAME
```

The system was run on a Univac 642 computer with RD281 disk drives. The executive used was the Master Control System or MCS.

The major functions of the MCS were scheduling and control functions, and file and storage management. MCS was intended to: be responsive to priorities and job structures defined by the user, maximize resource utilization, promote system continuity, facilitate reconfiguring and partitioning

of system resources, assure file integrity and security, and provide for file growth.

MCS monitored system performance by using table printouts. Job priorities were assigned by automatic implementation of priority assignments. File storage allocation was controlled by automatic dynamic storage allocation based on input of file use factors. MCS schedules jobs, files, peripheral equipment, memory and storage, and I/O requests.

Conclusion

Currently, the military's data base management needs are being met by bringing up limited DBMS capabilities on a variety of machines including minicomputers such as the UYK-20. It is recommended that a coherent strategy be undertaken in providing military systems personnel with a general and flexible capability that will enhance ease of implementation and reduce the costs of bringing up the initial versions of the system.

PART III

FEATURE-BY-FEATURE ANALYSIS

OF THE PRINCIPAL COMPONENTS OF

TACTICAL EXECUTIVES

CHAPTER 7

INTERRUPT HANDLING

7.1 Introduction

Upon occurrence of a processor interrupt, the executive automatically saves the current CPU status, and then either performs the processing directly or reflects the interrupt to an entry point of an appropriate module. The portion of the system which does this is termed the interrupt processing function.

Typically, the classes of interrupts recognized depends on the machine architecture. The following breakdown is typical:

For hardware faults and power failure, typically a transfer is made directly to some fixed routine, often the appropriate entry of a selected module can be scheduled.

For software faults, typically a message is constructed and set to a system error-processing module (thus scheduling that module). For some software faults (which are not treated as errors), other processing is performed.

For input/output interrupts, the I/O entry of the module which initiated the request leading to the interrupt is scheduled.

For Executive Service Requests, which are called via a hardware instruction that generates an ESR interrupt, the processing is performed by the Executive.

The analysis of interrupt handling in the executives attempts where possible to adhere to the following outline:

I. Analysis of interrupt processing function
II. Discussion of interrupt types recognized
III. Interrupt scheduling
IV. Special requirements

7.2 Comparison

This section presents a system-by-system comparison of Navy Executives' interrupt processing features. It must be noted that since ATEP, ATEP/MAX, ATEP/MMS, and SDEX/7 all run on the UYK-7, it is appropriate to first discuss the interrupt handling features of the UYK-7 computer.

7.2.1 UYK-7 Interrupt Processing

Executives that run on the UYK-7 respond to interrupts generated by computer programs and the AN/UYK-7 computer CPU equipment. Interrupts cause the AN/UYK-7 computer to suspend the executing program and to transfer to an interrupt entrance address associated with the class of interrupt. An interrupt status code and the address of the next sequential instruction of the suspended program and the CPU active status register are stored in the control memory of the AN/UYK-7 computer CPU. The interrupt status code identifies the source of the interrupt, and the interrupt processing function interprets the interrupt status code to determine the appropriate executive processing function to receive control. There are four (4) classes of interrupts: (These classes of interrupts are the same for all UYK-7 computers).

7.2.1.1 Class I Interrupts

A Class I interrupt indicates a power failure or an equipment or equipment fault. All lower class interrupts are locked out until this particular interrupt has been processed. In all cases except power tolerance, control is passed to a read-only hardwired program. The ROM program then passes control to the executive. For power fail interrupt, 250 microseconds are available in which to save the contents of control memory, then the computer is shut down. The intercomputer timeout interrupt can be handled either by the executive or by an entry point into a module specified channel program. If this alternative method is not used, an error condition is raised.

7.2.1.2 Class II Interrupts

Class II interrupts are all software related. These program faults signify errors that are identified and passed to error management routines for handling. This class includes floating point errors, break point interrupts which are used in the test mode of operation, and clock interrupts (which are used to limit the amount of CP time spent within a user module and for time slicing specified user module tasks).

7.2.1.3 Class III Interrupts

Class III interrupts are all I/O related and signify status of a requested I/O operation. Typical returns might be successful completion, device busy, or error condition. The IOC monitor clock interrupt indicates that a periodic task is due to be scheduled.

7.2.1.4 Class IV Interrupts

These are Executive Service Requests (ESR), and are generated by the execution of the "enter executive state" instruction. This switches the CPU operating mode from task state to interrupt state. The executive must then activate and process the ESR.

7.2.1.5 Interrupt Scheduling

In the AN/UYK-7 computer, interrupt scheduling is handled by the hardware on a priority basis. Class I interrupts are handled before Class II, II before III and so on. If the CPU is in interrupt state, only Class I and Class II interrupts are handled immediately.

7.2.2 ATEP

The ATEP interrupt processing function responds to Class I interrupts in the standard way described above. It responds to Class II interrupts (program faults), by passing control to the system error module which sends a message to the operator, builds an error packet describing the condition, and schedules the designated error module.

IOC monitor interrupts (Class III) are requested by the initiating task so the module can be informed of certain I/O conditions, e.g., buffer full, buffer empty, or I/O complete. Interrupt information is provided to the ATEP interrupt processor which in turn schedules the requesting module at either the buffer complete entry point or the I/O channel program complete entry point (depending on the interrupt) with the I/O packet set to identify the operation as input. For an equipment error interrupt an attempt is made to retry the operation. The system error module is called if a definite error has occurred. I/O interrupts are queued--status information is placed in a queue for processing.

In the event of a Class IV interrupt, all legal ESRs are requests for some ATEP service, and the interrupt processing program transfers program control to the ATEP/MAX processor that provides the desired service.

When a faulty parameter is discovered, the system error handler builds an appropriate error packet as an input to the error processor.

7.2.3 ATEP/MAX

The interrupt processing function is identical to ATEP except for its special multiprocessing requirement.

The hardware instruction that enables or disables IOC interrupts on a particular channel only affects the interrupts

on the particular CPU that issued the instruction. In a multiprocessor configuration in which more than one CPU accepts interrupts from the IOC it is necessary for the other CPUs to also give the enable/disable instruction. The CPU that receives the ESR command signals the other CPUs via an Interprocessor Interrupt. The I/O processing function in the receiving CPUs then repeats the enable/disable instruction.

7.2.4 ATEP/MMS

The interrupt processing function is identical to ATEP except for the multiprocessing requirement (identical to ATEP/MAX), and the ATEP/MMS shared memory requirement.

In a shared memory configuration, there may be limits in the connection between CPUs and memory units, between IOCs and memory units, and between CPUs and IOCs. The following requirements are satisfied by the ATEP/MMS Kernel:

a) If a module running in one CPU gives an I/O ESR that relates to a channel in an IOC that is not connected to the local CPU, the ATEP/MMS Kernel passes on the request to a CPU that is connected to that IOC via the Inter-CPU Request Table. The Kernel in that CPU carries out the ESR. A requirement on the AN/UYK-7 connectivity is that the IOC must be connected to the memory unit that holds the private data of the requesting module.

b) If a CPU picks up an IOC interrupt that requires the scheduling of a module that runs in another CPU, the local ATEP/MMS Kernel passes the scheduling request to the Kernel in the module's CPU via the Inter-CPU Request Table.

c) As in multiprocessing, if more than one CPU is connected to an IOC, any enable or disable interrupt command to the IOC must be sent from all connecting CPUs. These requests are passed from CPU to CPU via the Inter-CPU Request Table.

7.2.5 SDEX/7 Standard Executive

The SDEX/7 handles the standard four classes of interrupts that are generated by the UYK-7.

When a Class I interrupt occurs (hardware failure), the error is indicated by passing the associated interrupt status code to the error management function for handling.

Class II interrupts (software errors) are identified and passed to error management routines for handling. The following Class II interrupts are handled as special cases:

a) Floating point error - status is saved and CP control is returned to the point of interrupt. If the executing module subsequently executes the Return Floating Point Error ESR, a Class II interrupt status code and the address of the instruction causing the error are passed to the module.

b) Breakpoint interrupts - when this occurs, the task state environment and the Class II Designator Storage Words are passed to the module registered for processing breakpoint interrupts.

c) CP Monitor Clock Interrupt - this interrupt is used to limit the amount of CP time spent within a user module and for time-slicing specified user module tasks. It signifies the module task currently being execured has exceeded its allotted run time. If the task is suspendable successor or time-dependent task or background task, the task state environment is saved, the task is rescheduled for another time-slice and CP control is passed to the scheduling function. Otherwise, a module overrun error is indicated and passed to the error management function.

Class II (I/O) interrupts are handled in the standard manner. If the requesting module registers not to accept the interrupt, the SDEX error management function is called. If the module does elect to process its interrupt, a lockout of all other Class III interrupts occurs for one millisecond.

Class IV interrupts are caused by modules executing the "Enter Executive State" instruction. When a Class IV interrupt is received, the interrupt management function validates the interrupt status code as a proper ESR request. If it is not a proper ESR request, it is passed to the error management function as an illegal ESR. Otherwise, CP control is passed to the appropriate executive service routine.

In the processing of an ESR, certain validity checks are performed. If a validity check shows a condition where an ESR cannot be completed, a status indication (negative flag in a task accumulator) is set and CP control is returned to the requesting task. It is the task's responsibility to monitor this ESR status. In other instances certain executive conditions such as scheduling list overflows prevent the ESR from being completed. These are considered error conditions and passed to the error management function.

When a requested ESR process has been completed, CP control is returned to the requesting task at the point where the interrupt occurred. In this case, the status indicator is cleared indicating completion of the ESR. The ESRs requesting exits are exceptions and result in CP control being passed to the scheduling function.

7.2.6 AN/BQS-13

The Computer Executive Program (CEP) for the AN/BQS-13 operates as an interrupt-driven program on the AN/UYK-7. That is, the only entries to the CEP are interrupts. The CEP recognizes and handles the standard four classes of interrupts on the AN/UYK-7.

7.2.7 P-3C Update

The P-3C Update runs on the CP-901 computer. The executives' interrupt processing function must handle four types of interrupts.

Class I interrupts are hardware related failures, including power faults and memory protection. They are handled in the executive by error processing routines that function in degraded mode.

Class II interrupts are software-related program failures and are handled by error processing routines in the executive.

Class III interrupts are I/O related interrupts and are processed by the I/O interrupt handler. The handler saves the environment of the interrupted task, associates the interrupt with the Task or Executive module which is waiting for the I/O event and informs it of completion, determines if any additional task or Executive Requests for this device are queued, and interfaces with the I/O request module if a request has been queued.

Class IV interrupts are generated by Executive Service Requests from User tasks and application modules when using an Executive call. The executive performs the operation called for.

I/O interrupts can be generated by several sources, including the following:

a. I/O Monitor Interrupts are associated with all I/O data transfers except to or from those devices which only require a few microseconds to transmit their data. The executive is not further interruptable at this point due to a CP-901 architectural limitation which will not allow I/O data channels and their interrupts to be selectively enabled or disabled.

b. External Interrupts are used by various devices to signal the status of both current and previous data transfers. Unsuccessful completions interface with the error processing module of the executive.

c. The Count Down Clock interrupt is generated by a continuously decremented clock and will interrupt all lower priority interrupts. It can be used to schedule periodic tasks and to check possible loop conditions in Application tasks. The Count Down Clock Interrupt always preempts the task in control of the CPU and forces the examination of the Dispatching queue. This technique assures that lengthy background tasks do not monopolize the CPU.

7.2.8 COS/UYK-20

The Interrupt Processing Function is responsible for the initial receipt and subsequent processing of all hardware interrupts. Only three (3) interrupt types are recognized:

Class I Interrupts are hardware-related faults including power failures, and are sent to an appropriate error routine within the executive.

Class II interrupts are software program failures and are sent to appropriate error routines within the executive.

Class III interrupts are Executive Service Requests which can be generated by the user and applications tasks. It includes all I/O related interrupts.

Each class provides unique interrupt types arranged by priority. When control is automatically transferred to the hardware-assigned location associated with the class and level of interrupt, an indicator is checked to determine if the interrupt is scheduled along with the parameters defining the interrupt, and control is returned to the interrupted task.

If the flag indicated that the interrupted task is preemptab the priority of that task is compared to the priority assigned to the interrupt. Depending on which has the higher priority, one task will be suspended (i.e., its operational environment saved and sent to the appropriate dispatching queue) and the other task will be given (or resume) control.

7.2.9 SDEX/20

Three types of interrupts are handled by SDEX/20, with ESR calls being treated as a special case. SDEX/20 will only process major hardware errors; all others, if not registered for by a user module, are treated as errors.

Class I interrupts are of the highest possible priority and indicate major hardware problems. Such problems would include memory failure or loss of power tolerance.

On the occurrence of a power tolerance interrupt, SDEX/20 will save the general registers and then loop testing the power tolerance indicator. If power falls below a certain level an automatic master memory clear results. If power returns, SDEX/20 transfers to the error management function. Processing then resumes. If power returns after the master clear has taken place, SDEX/20 will automatically restart the system.

Upon the occurrence of a memory resume interrupt, control is transferred to the error management routine and normal processing is continued.

Class II interrupts include real-time clock and monitor clock interrupts. Software errors, such as "invalid op-codes" are also Class II, and are sent to the error management function.

Monitor clock interrupts occur when a timer has run out. These interrupts enable a job to perform the following tasks: Monitor clock interrupts occur when a time has "timed out."

- Acquire immediate control at its time-critical entrance; this provides the system with a precise method of scheduling time-critical jobs.
- Initiate a specific I/O chain; this is also useful for periodic I/O.
- Specify a successor task.

Class III interrupts are generally I/O related. Upon occurrence of a Class III interrupt, both classes II and III are temporarily locked out. If a module has registered for the particular interrupt, control is given to the module's interrupt handler. Otherwise, control goes to the error routine; SDEX will not automatically process I/O related interrupts. (ESR requests are not considered to be interrupts on this system; on most other Naval systems they compose Class IV interrupts).

7.2.10 Proteus General Purpose Executive

The first stage the interrupt processor on the CPIO effects the saving of all program status information and registers for the interrupted task in a storage area associated with the task. During interrupt processing interrupts of a higher priority remain enabled. Thus user tasks may be interrupted by any interrupt source while an executive task may be interrupted only to process interrupts requiring a higher priority executive task to be executed.

When interrupt processing (possibly involving a pass through the scheduling and dispatching algorithms) is complete, the executive is required to either return control to some interrupted task or initiate a new user task. In any case, exit from the executive is preceded by restoration of the program status and registers to the saved state associated with the task.

Three (3) types of interrupts supported are:

Class I - hardware failure
Class II - software failure
Class III - Executive Service Request

7.3 Discussion

Interrupt processing is substantially the same for each of these executives: a function is provided that recognizes the type of interrupt and sends it to the appropriate handler. (Comments referring to ATEP include ATEP/MAX and ATEP/MMS.)

Power failure, hardware failure, and software faults, (Classes I and II) are handled in the same manner by all the executives with the following exceptions:

* Only ATEP, SDEX-7, and COS provide a reloading scheme for graceful degradation and load-leveling in the case of a hardware failure in a multiprocessor configuration.

* Only ATEP and SDEX-7 specifically provide breakpoint handling for test mode of operation.

For I/O-related interrupts ATEP, P3-C, SDEX-7, and EQS-13 specifically designate a particular interrupt type: Class III. COS and Proteus group them with Executive Service Requests. The lack of a distinct interrupt type is seen as a disadvantage in speed and flexibility of scheduling. Only ATEP and SDEX-7 provide the user with the fullest degree of flexibility in handling completion of I/O. SDEX/20 also supplies such flexibility but at the cost of never having the capability to let the system process interrupts automatically. P3-C does not allow the user to temporarily suspend I/O interrupts. All the Executives do provide a scheduled interrupt for time-critical tasks (monitor clock).

Under Executive Service Requests, only ATEP does not require the user to handle an ESR which cannot be completed. ATEP builds an error packet and sends it to the Error Management Function, while the other executives give the user the alternative of handling the error condition himself. It is anticipated that for the design of OS/AADC, flexibility in interrupt processing will be among the most important considerations. Additionally, it will be necessary to design for very low overhead within interrupt processing. It may be decided feasible to place certain functions directly in the hardware in order to achieve the speed necessary, although this has the potential of affecting overall flexibility.

CHAPTER 8

SCHEDULERS

8.1 Introduction

One of the most important jobs of any executive is the scheduling of tasks. An appropriate scheduling algorithm can enhance system response, support time-critical tasks, and give priority to important tasks. Scheduling algorithms are especially critical in Navy systems because of their real-time nature.

8.1.1

For the purposes of this report, the following view of schedulers is used (this may vary, at least in terminology, with that used in the CPPS of those systems).

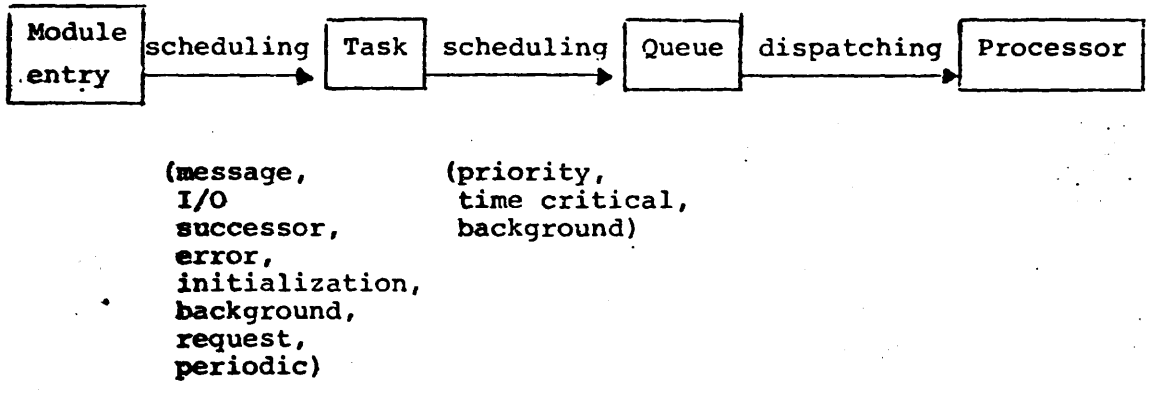
The machine code for a program is held in a module. A module can have some number of entries which are to called for various reasons (e.g., because a message has been sent, or because an I/O)interrupt has come in;.

When an event occurs that makes a module entry ready to run, that entry is scheduled, creating a task which is entered in a queue (or queues*) of tasks that are to be run. Part of the operation of scheduling involves the order in which the queue is kept, and the algorithmic basis for this order. For example, each module entry might have associated with it the priority that its task should have, and the queue would then be kept in priority order. Finally, whenever it is possible to run a task, the next one is chosen from the queue (or queues) and dispatched (started

*Multiple queues are often used to make parts of this processing quicker. The internal format of the database is not usually important.

to run on a processor). In some systems, a task waiting to run may pre-empt a task that has not yet finished running, if the task running is pre-emptible (and if, in some systems, the new task is at a higher pre-emption level). In this case, the status of the pre-empted task is saved, and it is resumed as soon as the new task completes. Some systems do not allow pre-emption.

Summarizing, the following diagram shows the differences between scheduling and dispatching:



8.1.2 Module Entry Scheduling

The message entry of a module is scheduled when a task sends a message to it. This is called message scheduling.

The I/O entry of a module is scheduled when an interrupt associated with an I/O request initiated by a task running in that module occurs. This is called I/O scheduling.

Every task can have associated with it some number of successor modules. When the task terminates, the successor entries of these modules are scheduled. This is known as successor scheduling

The error entry of a module is scheduled when an error occurs in a task running in that module. The exact types of errors differ from system to system. This is called error scheduling.

During system initialization and system restart, the initialization entries of all modules are scheduled. This is called initialization scheduling.

The background entry of a module allows a background task to be created, to run when there are no non-background demands on a processor. This is called background scheduling.

In some systems, an entry of a module can be explicitly scheduled via a supervisor call. This is termed request scheduling.

The above types of module entry scheduling are event dependent. There is also a class of module entry scheduling known as time dependent scheduling.

Primary among these is periodic scheduling. In this case, a module has associated with it a periodicity, which indicates how often the periodic entry of the module should be scheduled.

8.1.3 Task Scheduling

Task scheduling algorithms in the Navy fall into the following categories (or into some combination of them):

1. Many Navy schedulers are priority driven. When called, they search the dispatching queue (or queues) for the task with the highest priority. This allows urgent processes to be executed quickly. Priorities of tasks are usually fixed, but often can be changed at run time.

2. Time critical scheduling associates with every task a time by which it must be completed. The amount of time that the task will take is also known.

3. Background scheduling involves the scheduling of background tasks, which is done when no other tasks remain to be run. These tasks are not real-time, and are typically maintenance oriented. Background tasks are almost always pre-emptible.

8.1.4 Task Dispatching

When the dispatcher is called, it checks to see whether the next task to be run can be started. In a non-preemptive system, this implies checking whether a processor is free. In a

preemptive system, it must check whether there is a processor which is either free or is running a task of lower preemptive priority than the task waiting to be run.

If a new task can be run, the status of the preempted task, if any, is saved, and the new task started to run on the machine. Preempted tasks are typically rescheduled as soon as possible after the completion of the new task.

The dispatcher is always called upon task termination. In many systems it is also called upon scheduling of module entries, of whenever the exective gains control, or in some subset of these cases. As noted above, calling the dispatcher does not always start a new task running.

When a new task is started running, it either runs to completion, or runs until it is pre-empted. The amount of time that a given task takes is typically very small. On some systems, in order to support longer tasks, time-slicing is also incorporated, and treated as a form of pre-emption.

8.2 Analysis

The chart on the following page compares the schedulers of the major operating systems we have studied.

SCHEDULING FEATURES

Executive	Multi-processing	No. of pre-emption levels	No. of priorities per preemption level
ATEP	No	4	16
ATEP/MAX	Yes	compile-time variable	compile-time variable
ATEP/MMS*	Yes	compile-time variable	compile-time variable
P3-C	No	5	16
SDEX/7	Yes	4	1
COS/UYK-20	Yes	compile-time variable	compile-time variable
BQS-13	No	variable	variable
PROTEUS	No	not specified	not specified
SDEX/20	No	4	up to 15

*Multiple CPU's each with its own executive, may share common memory.

8.3 System-by-System Comparison

The following is a system-by-system comparison of Navy Executives scheduling functions.

8.3.1 ATEP

8.3.1.1 Introduction

ATEP's scheduler maintains one priority-ordered queue of tasks awaiting execution. ATEP dispatches for execution the highest priority tasks in the queue, and provides the required data linkages for the task.

In ATEP, the default priority for a task is the priority associated with its module in the master module list.

8.3.1.2 Priority

ATEP maintains four (4) preemptive levels, with sixteen (16) non-preemptive priority levels within each level. Two or more tasks within the same level are served on a FIFO basis. A task in one preemptive level can preempt tasks in lower preemptive levels, but not tasks in the same preemptive level. A task, however, can designate all or part of its execution as non-preemptible. A preempted task will be dispatched before any other task belonging to the same preemptive level.

8.3.1.3 Successor Scheduling

An operating task has the capability to specify up to four (4) successors to be scheduled for execution. It also has the capability to specify the priority and preemptiability status with which a successor is to be scheduled. If the predecessor does not specify a priority, then a default for the module is used.

8.3.1.4 Message Scheduling

An operating task can specify up to four (4) tasks to receive any one message. The priority with which the receivers of the message shall be scheduled is specified by the sender or a default for the module is used.

8.3.1.5 Periodic Scheduling

ATEP schedules periodic modules when they are due, and then according to its module's priority. A module can request that ATEP change its frequency for running by issuing the appropriate request.

8.3.1.6 I/O Interrupt Scheduling

When a I/O interrupt occurs, ATEP automatically schedules the module associated with the interrupt, at it's modules priority.

8.3.2 ATEP/MAX and ATEP/MMS

ATEP/MAX has the same basic scheduling algorithm as ATEP, execpt that it also supports multiprocessing. The following writeup on ATEP/MMS also applies largely to ATEP/MAX.

8.3.2.1 Multiprocessing

ATEP/MMS can support a maximum of 3 CPU's and 2 IOC's or 2 CPU's and 4 IOC's. In this multiprocessing environment, the ATEP/MMS Kernel provides load leveling and graceful degradation. Load leveling allows any processor any task available except for those dedicated to a particular CPU. Graceful degradation allows, after equipment failure, for surviving equipment to take over the tasks of any casualty unit so that the entire operation still continues at the cost of less performance.

8.3.2.2 Priority

There is one scheduling queue per copy of the ATEP/MMS Kernel (each CPU has its own copy). There are four (4) priority levels with sixteen (16) sublevels apiece. If, during multiprocessing, a module is preempted in one CPU, it can only be restored in the same CPU.

8.3.2.3 Successor Scheduling

A predecessor can specify the successor's priority. If not specified, the successor module's priority is used.

8.3.2.4 Message Scheduling

One (1) to four (4) modules can be scheduled to receive a single message. The priority of the recipients can be specified along with the send request. If not specified, the receiving module's priority is used.

8.3.2.5 Error Scheduling

The ATEP/MMS fault processor requests scheduling of a task module associated with the error, or alternatively, the ADEP central error module.

8.3.3 P3-C UPDATE

8.3.3.1 Priority

The dispatcher queue is ordered by task priority. In P3-C update, there are five (5) preemption levels, each with sixteen (16) non-preemptive priorities. Within a priority, the queue order is FIFO.

In addition to preemption levels and priorities within levels, two (2) task attributes are also considered in the scheduling algorithm:

a. A task which can complete all processing requirements within 10 milliseconds can only be preempted by Preemption Level 1 tasks, thus minimizing task switching overhead.

b. A task which has issued a "Lock Data Base" request and has not yet issued a "unlock Data Base" request. These tasks too, are susceptible to preemption by Preemption Level 1 tasks only. This provision is intended to assist data base integrity in a preemptive environment, and to minimize wait times for the data base.

Background tasks are also supported, at the lowest possible priority.

8.3.3.2 Multiply Scheduled Module Entries

Certain entries can be multiply scheduled, i.e., they can be scheduled even while they are in an active or ready-to-run state. The following rules would then apply:

a. The queue will contain a task for each request.

b. All tasks at an entry must at the same state/priority level.

c. The order of tasks for a particular entry shall be FIFO.

8.3.4 UYK-7 Standard Executive (SDEX/7)

8.3.4.1 Successor Scheduling

At compile-time the user can specify whether successor tasks for a given module are to be time-sliced.

8.3.4.2 Message Scheduling

SDEX keeps a buffered list of messages awaiting processing. A FIFO flow of messages to receiving user modules is maintained.

8.3.4.3 Task Scheduling

Periodic and time-critical scheduling are provided. Time-dependent tasks are tested for execution strictly on a round-robin basis and have no associated priority.

Background tasks are time-sliced using the CP monitor clock to periodically return CP control to the dispatching function.

Once a background task has been started, it proceeds on a time-sliced basis until the tsk is complete. The background task is then rescheduled.

There is no priority system. Each task is tested for execution strictly on round-robin basis.

8.3.5 COS/UYK-20

COS uses an open-ended, multi-priority scheduling mechanism. Priorities are dynamic and changeable at run time. COS is suitable for a multi-computer multiprocessing environment.

8.3.5.1 Task Queues

COS supports a variable number of task queues. Event-oriented tasks and time-oriented tasks are kept in separate queues.

8.3.5.2 Task Suspension

The capability for task suspension is provided. If an interrupted task is not suspendable, the interrupting task shall be queued and the interrupted task allowed to continue execution.

8.3.6 AN/BQS-13

The BQS-13 Sonar Executive uses a priority scheduler. The dispatching function is entered either by the interrupt handler or by executive call upon completion of an applications task.

8.3.6.1 Dispatching Algorithm

Preemption is supported. If the dispacher is called by the interrupt handler, the priority of the previously running program, if it can proceed, is compared with the highest priority new task waiting to run; the highest priority ready task is dispatched.

If no other programs are ready for execution, the task scheduler directs the computer to executive Performance Monitoring and Fault Location (PM/FL) programs. The execution of the PM/FL programs shall continue until an interrupt occurs and a program of higher priority is ready for execution.

8.3.7 PROTEUS

Proteus controls three (3) scheduling types.

8.3.7.1 Request Scheduling

Request scheduling is done by explicit request. The priority of the task scheduled can be explicitly given in the request.

8.3.7.2 Event Scheduling

Event scheduling is done when a predetermined event occurs. These events are specified at sysgen time. Generally, the events are external ones to be detected by the executive.

8.3.7.3 Message Scheduling

A message task for a module is scheduled when the executive or a user task sens a message to the module.

A sending task has the option of specifying the priority of its receiving task.

A message can be sent to more than one message task.

8.3.7.4 Time Critical Scheduling

Time-critical task are implemented as periodic tasks which, once scheduled, must be completed before their next scheduling request occurs. A time-critical module entry has associated with it its required compute time.

Preempted time-critical tasks are re-scheduled on the basis of a "least-time-to-go" algorithm.

8.3.7.5 Background Scheduling

Background tasks are scheduled at a predefined periodic rate. The desired periodic rate of background task is associated with the module entry.

If the time for a background entry to be scheduled again should arrive before the completion of a task currently running in that entry, scheduling of the entry is skipped for the period.

Pre-empted background tasks are scheduled on the basis of a "shortest period" algorithm.

8.3.8 SDEX/20

SDEX/20 can schedule four types of tasks: successor, message, periodic, and background. The relative priorities are:

successor message time-dependent background

A task is given CPU time whenever there are no higher priority tasks waiting and before any lower priority task. It is possible to selectively drop certain scheduling classes at compile time, with the exception of message scheduling, which must always be present.

8.3.8.1 Successor Scheduling

Successor tasks are scheduled by one of 15 levels of priority. Tasks with the same priority are scheduled on a FIFO basis. If there are no tasks waiting to be scheduled, the next lower class, normally the message scheduling class, is searched. Otherwise the highest priority task is executed. Upon task completion, control returns to the scheduling function.

8.3.8.2 Message Scheduling

All message processing is scheduled on a FIFO basis. When all message processing is completed, the next scheduling level, normally time-dependent tasks, is searched. When a message is located for processing, a pointer to the message packet is sent and the module is started at its message entrance. Upon completion of message processing, control is returned to the scheduling function.

8.3.8.3 Time-Dependent Scheduling

When higher class tasks have been honored, SDEX will search the list of time-dependent tasks for a job whose time-to-initiate-execution (TE) is less than or equal to the current timer. There is no priority among time-dependent tasks; each task is checked on a round-robin basis, and the first suitable task is scheduled.

When a suitable time-dependent task is located, the next TE of the task is computed and CP control is given to the task at its time-dependent entrance. Upon completion, control returns to SDEX/20. A facility to stop a time-dependent task and to delete its entry in the time-dependent tables is provided through the appropriate ESR requests.

It should be noted that this scheduling facility provides no support for very-time-critical jobs which MUST run at certain intervals; in this scheduling class there is a possibility that tasks may wait indefinitely for CPU time. There is another facility which exists for very-time-critical tasks; such tasks may set a system timer and will be immediately started whenever the interrupt from the timer rounout is received (the task previously executing will be pre-empted). This support is needed throughout Navy systems and SDEX/20 would have been inadequate without it.

8.4 Discussion

Most of the scheduling methods used are appropriate for their respective environments since they were largely written for these environments. The schedulers used on FOS will need to be able to support all the sets of functions previously used. Additionally, it will be necessary to carefully balance, within any particular version of FOS the tradeoff between flexibility at runtime and high speed. The scheduler and dispatcher form the most crucial part of the system.

CHAPTER 9

MEMORY MANAGEMENT

9.1 Introduction

Memory management consists of:

1. Managing memory permanently allocated for modules and their data areas, including linking and associated functions.
2. Allocating memory for newly stored non-core-resident modules.
3. Allocating memory as a temporary "scratchpad."
4. Allocating memory for buffer space.
5. Managing memory so that as much as possible is available for the user at all times.

Memory management is often machine-dependent, as many techniques (paging, segmentation) are at least partially hardware-supported.

9.2 Memory Management Features

Executive	Supports Dynamic Storage?	Supports Temporary Storage?	Supports Non-core Resident Jobs?	Supports Dynamic Files?
ATEP	Yes	Yes	Yes	No
ATEP/MAX	Yes	Yes	Yes	No
ATEP/MMS	Yes	Yes	Yes	No
P-3C	Yes	Yes	Yes	Yes
UYK-7 SDEX	Yes	Yes	No	No
COS/UYK-20	Yes	Yes	Yes	No
BQS-13	No	No	No	No
PROTEUS	No	No	No	No
SDEX/20	No Information Available			

9.3 General Description

The degree to which Navy Executives manage memory varies enormously. Some do not even have to manage memory -- all modules are permanently core resident. Other Navy Executives keep certain jobs permanently in memory and have an area reserved for non-core-resident jobs and temporary storage, on a first-come-first-served basis. Other Navy Executives run a true multiprogramming environment.

9.4 System Descriptions

The following is a system-by-system description of current Navy Executives memory management facilities.

9.4.1 ATEP

ATEP uses a sophisticated system for memory management. Many programs are considered to be core-resident and as such are always available and active. Other programs are non-core resident and are available when needed from a disk unit.

ATEP maintains a fixed pool for "temporary storage" - used for either buffer space for currently running programs to or to hold a newly activated non-core-resident job. Temporary storage can be requested by any running job in blocks of 2^4 - 2^8 words. If storage is not available the Executive will roll out any non-running non-core-resident jobs or unused buffer space. The Executive will not remove a finished job until such space is needed.

The amount of temporary storage available depends on system configuration and is communicated to the Executive upon system generation. Memory is managed through a Temporary Storage Directory, containing a list of all blocks, their identity, and size, and through a Temporary Storage Status Table containing the status of all data blocks with an indicator of available blocks for each size.

ATEP also performs base register setup and linking to common databases and subroutines. If the memory pool reserved for non-core-resident jobs and buffer space is exhausted, further requests will be queued and acted on by priority level. Non-core-resident jobs will be rolled out when inactive and replaced by waiting jobs. On no occasion will core-resident jobs be removed.

9.4.2 ATEP/MAX

ATEP/MAX uses a very similar scheme to that of ATEP. Certain programs are core-resident; a portion of memory is reserved for temporary storage and non-core-resident jobs, similar to ATEP. Temporary storage can be requested in blocks of 2^4 to 2^8 words; maximum memory available for non-core-resident programs is 8,192 words (8K).

Memory is managed through the same set of tables as ATEP, namely a Temporary Storage Directory containing a list of all blocks, their identifiers, and size. The Temporary Storage Status Table contains the status of all data blocks with an indicator of available blocks for each block size.

ATEP/MAX also performs base register setup and linkage to common databases and subroutines. Non-core-resident jobs that have terminated remain in core until more temporary space is needed, when they will be copied back onto the disk. If more temporary storage is needed than available, ATEP/MAX will queue requests and run them on a priority basis. Non-core-resident jobs will be rolled out when inactive and replaced by waiting jobs. On no occasion will core-resident jobs be removed.

9.4.3 ATEP/MMS

ATEP/MMS presently uses the same memory management scheme as ATEP or ATEP/MAX. It is, however, designed in a modular fashion and is intended to be augmented in power in the future. The following describes the Kernel of ATEP/MMS's memory management scheme, which is identified to the previous schemes used for ATEP and ATEP/MAX.

ATEP/MMS again maintains a memory pool for non-core-resident jobs and buffer/scratchpad space. Modules can acquire or release temporary storage on demand. Modules can even catalog temporary storage for later retrieval - a feature not available on ATEP or ATEP/MAX.

Dynamic storage is maintained for non-core-resident modules and buffer space; 256 words maximum is allowed for buffer space and 8,192 words (8K) for non-core-resident programs. A finished non-core-resident module will remain in core until memory is again needed - roll out is not provided. Two tables are again used: a Temporary Storage Directory containing a list of all blocks, then identifies, and size, and a Temporary Storage Status Table containing the status of all datablocks with an indicator of available blocks for each size.

ATEP/MMS again performs base register setup and linking to common databases and subprograms. If the temporary storage pool is exhausted requests will be queued and run on a priority basis. Non-core-resident jobs will be removed when inactive to allow jobs in the queue to run. On no occasion will core-resident jobs be altered.

9.4.4 P-3C

The P-3C Executive, running on the CP-901 computer, uses a complicated scheme for memory management. All programs and data are either core- or non-core-resident. The scheme is split into two sections.

The first section is used for core-resident jobs and is allocated at system initialization by the System Generator. Core-resident jobs are permanently in memory and are always available.

The second section is used for allocation of non-core-resident jobs and for "scratchpad" memory. When "scratchpad" or buffer space is requested, the request is passed to the Core Allocator module. This module accepts requests from:

1) Scheduler request to allocate or load a non-core-resident program or file.

2) Application task request to read a dynamic file.

3) Application task request to scratch/work area.

The Core Allocator must also know the task file identification, except on a request for scratch storage. There are four (4) types of temporary memory: transient task (non-core-resident job), transient task and file, file only (database), and scratch area. If core is available the request will be satisfied immediately; if not, the request is put on a "wait for core" list by priority type (either preemptive or nonpreemptive). When memory becomes available the Core

Allocator checks the preemptive list before checking the nonpreemptive list. Allocation within each list is done on a First In-First Out (FIFO) basis. Then a capability exists to go to a 16 level priority system using non-FIFO selection, but this is not currently implemented.

Memory is allocated by segments of varying sizes. The Executive contains a map of all segments stating if available or not available; if not available, the segment is either assigned to permanent database or an active task or file or is defective.

9.4.5 UYK-7 Standard Executive (SDEX-7)

SDEX's algorithm for memory management takes full advantage of the UYK-7's hardware segmentation. The executive uses a variable length table containing the base address and storage protection attributes of all user programs to manage memory. The executive also maintains a fixed amount of memory for user I/O buffers. No facility for non-core-resident programs is provided.

9.4.6 COS/UYK-20

The COS uses a memory management system. Memory is divided into a partitioned basis, the partitions being allocated at system generation. COS maintains a memory assignment table reflecting current tasks and memory assignments.

The task must specify the number of words requested, the absolute base address (relative to partition start) for the segment, and an identifier. A best fit algorithm is used when searching through partitions. If the request cannot presently be satisfied, a lower priority job is swapped to the disk. If no such job exists, the request is queued until memory is freed. When memory becomes available

the queue is searched for the highest priority job. The system operator is notified when the Executive feels memory requests are becoming excessive for capacity. An individual task may request to be swapped out if appropriate.

9.4.7 AN/BQS-13

The BQS Executive is designed for a small, dedicated application sonar system. It contains no real memory management. A Task Control Block is used to contain the status of all jobs. Job scheduling is done on a priority basis. There is no facility for running Non-Core-Resident jobs.

9.4.8 PROTEUS

Proteus uses no dynamic memory management at all. All programs can concurrently fit in core, and as such, there is no need for dynamic management. Proteus is being provided with facilities needed to support swapping.

9.5 Discussion

The techniques of memory management are highly hardware dependent. As has been shown on some of the more advanced systems above, higher order memory management techniques - segmentation and paging - are suitable for Navy systems.

Considering the features of current Navy tactical executives, the FOS should be able to:

- Manage memory dynamically, by segmentation and/or paging (depending on hardware available).
- Make dynamic buffer space available to running programs.
- Run non-core-resident programs.
- Have facilities for files to be stored in memory when requested, greatly speeding up file access time.

CHAPTER 10

I/O PROCESSING AND DEVICES MANAGEMENT

10.1 Introduction

The Navy Executives covered in this study typically provide an I/O initiator/scheduler function, along with a scheme for reflecting I/O interrupts back to the requesting module. This leaves the job of device management either to the user level or to standard system supplied handlers.

Typically, the I/O initiator/scheduler function is invoked by an ESR. It checks the requested operation for validity. After checks for device status, the request is either initiated or queued to be initiated later.

Upon receipt of an I/O interrupt, the interrupt processing function reflects the interrupt to the module registered to process I/O interrupts for the device.

10.2 System Descriptions

The following is a system-by-system analysis of I/O Processing and Device control.

10.2.1 ATEP I/O Processing

10.2.1.1 I/O Initiation

The calling sequence for initiation of all I/O operations is via an initiate I/O Executive Service Request (ESR) made by a user module. ATEP passes the interrupt to the ATEP I/O processor (XIOP).

Each module performing I/O operations on specialized devices may prepare and contain its own I/O channel program; for standard devices, ATEP provides a standard device handler. Along with each I/O ESR, the module must provide an I/O packet delineating the parameters of the operation. As a result of I/O initiation, access to the module's channel program is gained by the I/O Controller (IOC) and the program is executed.

Depending on the type of interrupt being monitored, either the "buffer complete" entry point or the "channel program complete" entry point of the module is scheduled.

10.2.1.2 Control Function

The I/O Control Function is responsible for controlling segment module I/O channel programs and their interfaces with the I/O channels. The I/O control function initiates all I/O operations and responds to requests from modules to perform I/O channel services and enabling/disabling of a channel. The I/O control function operates in conjunction with the interrupt processing function. The I/O control function is also responsible for processing computer program queues of Class III interrupts to prevent their loss.

10.2.1.3 I/O Controller

AN/UYK-7 hardware I/O services are supplied by the I/O Controller. The IOC controls I/O data flow and associated device control, and executes channel associated programs. A centralized I/O package is used by ATEP to construct channel programs for standard devices. For other devices, the module performing the input or output constructs the actual channel program. In either case, the channel program, running in the IOC asynchronously from CPU operation, can signal the occurrence of events significant to the associated module.

10.2.1.4 Data Buffer Control

Data buffers for I/O may be common or private data areas, or may be temporary storage areas obtained through storage request ESR's. To provide flexibility for a channel program to access more than one fixed buffer of data, variable buffer control words can be related to a channel program. These buffer facilities provide for double buffering, cyclic use of chained buffers and concurrent I/O, and processing of essentially continuous streams of data.

10.2.1.5 Peripheral Device Support

Peripheral device support falls into two major categories: general purpose devices supported completely by ATEP (disks, tapes, and printers) and specialized applications devices (the AN/SPY-1 radar, the MK 26 guided missile launching system, and Weapons Direction), which are in part supported by the appropriate user module and in part by ATEP.

The user communicates with all devices through specialized ESRs. Existing modules for general purpose devices provide channel programs and controls. For the specialized tactical devices, the user module must supply channel programs in addition, ATEP fields all I/O interrupts to assure that the responsible module receives notification of the interrupt. Typically the user has the option of specifying immediate return to the module or return only after the I/O operation is complete.

10.2.1.6 System-Supplied Device Handlers

The system-supplied device handlers for the AN/UYK-7 system are ATEP supplied task state modules. These contain channel programs which translate device-specialized ESR's from user modules into the basic I/O ESR's used by the ATEP interrupt and I/O processor.

10.2.1.7 Specialized Applications I/O Devices

These devices are not completely supported by ATEP but are also directly controlled by the running user program. Examples would be the AN/SPY-1 radar, the Command and Decision Control, and the Weapons Direction System. Special options peculiar to each device are available.

10.2.1.8 Error Recovery Processing

In the event an error is detected during I/O operations either the module requesting the I/O may be scheduled at its error entry point or the ATEP-provided system error handler module may be scheduled. In addition, provision is made for the device handler to display an error message on the operator's console.

10.2.2 ATEP/MAX

I/O processing and device management is the same as that of ATEP.

10.2.3 ATEP/MMS

I/O processing and device management is the same as that of ATEP.

10.2.4 AN/BQS-13

10.2.4.1 I/O Control Function

The I/O Control Function handles all I/O requests from operational tasks. When such a request is received, the function invokes the appropriate routines which examine the status of the required channel, and either initiate the request if not in use or place the request on a queue if it is in use.

A similar function handles I/O interrupts generated by devices, channels, and controllers. This function also invokes appropriate routines for handling initiation of IOC chains, data transfer, monitor interrupts, etc.

10.2.4.2 I/O Facilities

Sixteen channels are available to AN/BQS-13. They connect with both standard and applications-type equipment, including:

- Magnetic Tape Transport
- Typewriter Console
- Card Reader/Punch
- Disk
- DNA Sonar System
- Fire Control System
- Integrated Multifunction Console

10.2.4.3 Error Recovery

Recovery processing and error handling are accomplished via the error management function.

10.2.5 P3-C Update

10.2.5.1 I/O Initiation

Access to all P3-C Update I/O devices is controlled by the Input/Output Control Module (IOCM). It accepts requests for I/O initiation and also provides the necessary channel scheduling and queue manipulation logic required for those devices which have multiple users and/or share I/O channels for data or control purposes.

The IOCM allows users to specify wait or immediate return on their requests, interfaces with the scheduler and dispatcher as appropriate and only allows application hardware-test tasks to communicate with a device which has a "down" status. The IOCM maintaining control tables in conjunction with the I/O interrupt processing routines identifying:

a. Status of all I/O devices (up, down, busy, test,etc.)
b. Current user of the device
c. User error and normal return pointers
d. Maximum response time

10.2.5.2 Peripheral Devices

All I/O devices are categorized as either executive- or application-controlled. Executive-controlled devices are those which the executive program needs to meet its own data processing requirements (magnetic tape, hi-speed printer, drum, etc.) A part of the executive IOCM contains a full "handler" capability for each device based on the requirements of executive routines and application tasks.

10.2.5.3 Error Recovery

P3-C can detect malfunctioning I/O devices, disable the offending unit, and return an "error" indicator to the user.

10.2.6 SDEX/7

10.2.6.1 I/O Initiation

Executive supervision of I/O channel utilization is done by the input/output management function. Inputs to the function are ESRs from user modules requesting the initiation of I/O operations or the enabling or disabling of I/O channel interrupts.

Immediate initiation of an IOC command chain is provided for. A user module specifies the IOC number and the location of the IOC command chain to be initiated. CP control is then returned to the requesting user modules to continue processing.

All user modules responsible for interrupts on a channel are required to register their desired response for interrupts. A maximum of four (4) modules are allowed to respond to interrupts on one channel, one for each interrupt type. Response options provided to the user modules are:

a. Immediate user module entry through the I/O interrupt entrance; or

b. Queuing of the interrupt and scheduling of the user module's successor entrance.

10.2.6.2 I/O Interrupt Enabling/Disabling

User modules registered for I/O channels are allowed to selectively enable and disable interrupts on those channels through the use of ESRs. Interrupt enable or disable instructions are passed on to all CPs except when specifically prohibited by the user. The enable/disable interrupt instructions are executed by the receiving CPs when entering the scheduling function.

10.2.7 COS/UYK-20

10.2.7.1 I/O Initiation

Unlike the previous systems, control of all peripheral device I/O functions are centralized with the COS. All I/O requests are issued utilizing the Executive Return (ER) instruction and supply and I/O packet address. The I/O packet contains a function code, the channel number, the data location and length, and any data format information. The packet is analyzed by centralized I/O and then forwarded to the identified device handler.

10.2.7.2 Centralized I/O

Centralized I/O is the interface between user requested I/O and the hardware I/O handlers. When the I/O request is received, control is passed to centralized I/O. For all requests, the requesting task has the option of receiving control following I/O initiation or not. The request is checked for validity, associated with a physical device, and, if necessary, data conversion performed. In a multi-computer configuration, I/O calls for devices which are not connected to the processor making the request are transmitted to the appropriate processor via an inter-computer channel. If the device is available, the I/O is initiated immediately, and control returned to the user (if requested). If the device is busy, the request is queued and control returned as above.

10.2.7.3 Peripheral Device Handlers

A separate peripheral device handler is required for each type of peripheral device in the system. The device handler receives control from centralized I/O in response to an I/O request. The packet address and the channel device table are also inputs to the handler. The handler analyzes the function requested, incorporates user-requested options and initiates the I/O chain. When I/O termination has occurred the interrupt is processed by the handler, and control returned to Centralized I/O with the device status.

10.2.7.4 Error Recovery

All I/O device types are serviced by the peripheral error logging function. The error logging service receives control and validates that the device type exists and the the error is valid for the device type. If an invalid request is found, the task is aborted. If the request is valid, a control indicator is examined to determine if the user should be scheduled to receive control on the interrupt. An appropriate message is then constructed and output to the system operator.

10.2.8 The Proteus General Purpose Executive

10.2.8.1 I/O Initiation

Input/output operations can be initiated either by the user via an Executive Service Request (ESR) or upon occurrence of a stimulus for a Time-Critical Task or Selected Event Task. The I/O operations are described to the executive by an "I/O Request" which is a packet of information sufficient to allow the executive to initiate and monitor a specific I/O channel transfer and to schedule and communicate with the requesting task.

Whenever requested I/O requires activity on a busy channel, the request is queued until the channel is free. The request may select immediate return after the I/O request is honored (as soon as the dispatching algorithm will allow) or he may select to be suspended until the occurrence of the I/O completion interrupt. A time-out facility allows the executive to protect the user from indefinte suspension due to a looping or hung I/O transfer.

10.2.8.2 Status Returns

Status information is returned in the I/O Status Word specified in the I/O Request. This status is updated on each interrupt resulting from the I/O Request and whenever the request is queued or un-queued. This status information includes information on irrecoverable error conditions.

10.2.8.3 Error Recovery

For errors which are related to an I/O operation, such as channel fault or an equipment reject, the executive attempts to reco from the error condition and reset the channel or equipment to a con dition where the device is usable. The executive then reports to th task which initiated the I/O operation the error condition and the current status of the device.

10.2.9 SDEX/20

10.2.9.1 I/O Initiation

The executive manages I/O through the I/O management function, through which modules may register for I/O channel responsibility and initiate I/O. All I/O operations are conducted through ESRs. Actual I/O initiation is always done by the executive in accordance with an ESR request passed by the user module.

10.2.9.2 I/O Registering

User modules must register to receive the particular interrupt types on a specific channel; the executive will not automatically process I/O interrupts. Up to four modules may register for each channel -- one for every interrupt type. Two options for response are available when an interrupt comes through; the user may specify immediate interrupt processing through the I/O interrupt entrance or queueing of the interrupt and scheduling of the receiver as a successor task. The first form is useful fo real-time and time-critical rapid-response systems.

User modules registered for an I/O channel may selectively enable or disable their associated interrupts.

10.3 Discussion

10.3.1 I/O Initiation

For all Navy executives, the actual initiation of an I/O operation is handled by an executive supplied module. In all cases, this is done primarily in response to user requests (ESRs).

In addition to permitting I/O requests via ESR, the Proteus system allows I/O to be initiated upon occurrence of a stimulus for a Time-Critical or Selected Event Task. Thus, Proteus allows I/O to be periodically initiated along with a periodically scheduled module. The other systems can achieve the same effect in a more roundabout way by scheduling a periodic module to initiate I/O. The Proteus scheme may be seen to be both more efficient and simpler.

Under those executives which supply handlers for general purpose devices, the device handlers themselves utilize the centralized I/O initiation function. In ATEP the user might execute a tape request ESR but the tape handler is itself a task state module containing tape drive channel programs. The tape handler operates on the tape request ESR and then executes its own Initiate I/O ESR to the centralized I/O initiation function. This scheme has the advantages of keeping the executive state module as small as possible and of permitting the modular inclusion of only those device handlers required in a particular system configuration.

It is planned for the design of FOS to utilize the standard-device-handler organization, but to expand it to all I/O devices. For non-standard devices, a new device handler will first be written, and then it will be used in the standard way by the application program.

10.3.2 Validation

Only ATEP and COS make specific mention of the validation of requests. Presumably, however, the other executives perform some form of testing procedure.

Mention of the consequences of an invalid request is made in the Recovery Processing section.

Under ATEP, validation of I/O requests is done at system initialization time. Each module containing channel programs (including device handlers) executes a "Setup I/O" ESR, at which time that module is matched against ATEP supported tables detailing which modules are to be allowed access to which channels and what their access capabilities will be.

COS, on the other hand, makes a validity check on each I/O request as it comes in, because of the logical I/O structure maintained by COS. At the time of the check a match is made between the request and a physical I/O device.

10.3.3 Scheduling

All of the executives will immediately initiate an I/O operation if the required path is free. If the path is not free, responses vary:

In SDEX/7, the centralized module does not even check for path busy. Channel status checking and device handling is entirely under the control of the task modules which call the executive only for execution of the privileged "Start I/O" instruction.

In BQS-13 if a channel is found to be busy the new request is simply added to the existing channel program via chaining. No mention is made of how interrupts are generated signalling completion of requests buried in the chain or of how completed requests are removed from the chain.

Proteus and P3-C queue requests for busy channels within the centralized initiation modules and execute them on a first in--first out basis.

COS is similar to Proteus and P3-C in that it is the central initiation module which handles queued requests; however, in addition to this, COS offers a logical I/O structure. I/O requests are made to a logical device number and the request is associated with a physical device by the executive as part of the validation process. The requesting module has the option to allow substitution (i.e., if the disk is busy, output may be directed to a tape in the interest of speed). Substitution is only possible for output devices and if the option is not used, output requests will be queued first in-first out.

ATEP is in a sense similar to SDEX/7 in that the kernel I/O function does not queue requests. This is a function of the task state device handler, or of the requesting module in the case of a specialized device. In this way a separate queue is maintained per device. For ATEP supplied device handlers queing is on a first in-first out basis.

10.3.4 Channels

Under all of these systems, the actual hardware I/O is done by an I/O channel under the control of a channel program running asyncronously with respect to the CPU. A channel can indicate completion by generating an interrupt and invoking the centralized interrupt handling module, or, in the case of the AN/UYK-7 computer, a channel has a limited ability to communicate directly with a user module through a capability to set and test bits.

10.3.5 Buffer Control

There is a lack of information in this area. In P3-C I/O buffers are permanently allocated in specific locations in memory. In ATEP, the module requesting service allocates the buffers. In ATEP buffers can be chained and used cyclicly and may be dedicated areas or may be dynamically requested by the module.

10.3.6 Data Translation

ATEP and COS provide code translation facilities. In ATEP, code translation can be specified by a user requesting alphanumeric console I/O or can be requested in a general way through a translate ESR.

10.3.7 Peripheral Device Support

Device handler modules are supplied by the Executive for system supported devices. In ATEP, if a user module wishes to make an I/O request to a general purpose device a specialized ESR for that device is issued. The actual channel programs are contained within the device handlers for supported devices and within the user module for specialized devices.

System-supported devices are characterized by the presence of system-supplied device handlers which act as intermediaries between user modules and the device itself. The use of a common device handler allows concentration of all the idiosyncracies of a device into one spot.

10.3.7.1 Applications Devices

In all of the systems specialized devices that are dedicated to particular modules have the device handlers incorporated directly into the modules. This technique allows reduction of the overhead that would be necessary to call a device handler that is used by only one module.

Executive services for support of applications devices are the basic level ESR's for initiation and termination of I/O, and reflection of interrupts from the device.

CHAPTER 11

PROCESS SYNCHRONIZATION AND MESSAGE COMMUNICATION

11.1 Introduction

In many Navy applications, it is necessary to prevent simultaneous modification of shared data-bases by parallel tasks (also termed processes). These and similar problems are prevented by the use of process synchronization tools. Processes often need to pass information between themselves. Although this may often be done through shared segments, it is often simpler and more efficient to use a (synchronized) form of message communication, particularly if the messages are directly associated with the occurrence of events.

The process synchronization and message communications facilities are necessary to support multiprogramming and/or multiprocessing environments. These facilities provide the means for different tasks to coordinate their activities. This is desirable for many reasons, among them: protection of shared databases; data sharing between different tasks; and timing synchronization between different tasks to accomplish a larger activity. As this report demonstrates in earlier sections, current Naval tactical systems rely heavily on a multiple task approach (some further complicate the matter by utilizing multiple task approach (some further complicate the matter by utilizing mutliple processors); thus, facilities for task synchronization and message communications between these mutliple tasks are not only desirable, but required.

11.2 System-by-System Survey

The following is a system-by-system comparison of the Navy Executives synchronization and message handling functions.

11.2.1 ATEP Inter-Computer Communication

Communication between computers in the same system is via inter-computer channels. The number of inter-computer channels linking computers is specified in advance by AEGIS segments.

11.2.2 ATEP/MAX

ATEP/MAX supports multiprocessing and must ensure data base protection over several computational elements.

11.2.2.1 Data Base Protection

Before a task is allowed access to a common table, the appropriate lock is tested. If set, it is not allowed access. Otherwise, it sets the lock by issuing a Set Lock and Non-preemptive State ESR. This ESR also makes the task non-preemptive, thus minimizing the conflict with other CPU's by allowing the locking module to complete itscritical code as quickly as possible. As a precaution, all locks associated with a task will be automatically released when that task exits. A task may have a maximum of three (3) currently set locks.

11.2.2.2 Module Communication

Communication between tasks is achieved through task-to-task message transmission controlled by ATEP/MAX. Messages are in a standard format, and ATEP/MAX controls the routing, transmittal and reception of messages.

Modules communicating with each other may be within one computer, or they may reside in different computers of the same segment.

When a task needs to send a message, it first requests temporary storage before building its message and sending it. Messages are of variable length, the maximum being 256 words.

Messages sent to other computers are queued for the inter-computer channel program.

11.2.3 ATEP/MMS

11.2.3.1 Introduction

ATEP/MMS supports a multiprocessing multi-CPU environment. Each CPU contains the ATEP/MMS Kernal. The CPUs communicate with each other through shared memory.

11.2.3.2 Database Locking

To ensure that no race conditions occur in the accessing of Executive Common Data by the multiple CPU's, ATEP/MMS uses a test-and-set lock technique, similar to the one described for ATEP/MAX.

11.2.3.3 Message Communication

When ATEP/MMS operates in a non-memory sharing mode, message communication between modules is supported in almost exactly the same way as that in ATEP/MAX. The only difference is in inter-computer message processing.

11.2.3.4 Inter-Computer Messages

There are two (2) ways to send a computer-to-computer message. One is that used in ATEP/MAX, in which system tables indicate whether or not a message to a particular module is to be passed on to another computer. The sending task must do its own lookup. In the second method, the ESR packet itself indicates to which computer the message goes.

11.2.4 P-3C Update

11.2.4.1 Database Locking

The P-3C supports approximately 300 tasks, all of which have access to a common data base. To ensure that no task accesses this common data base when another task is updating data, the P-3C Update supports ESR's which the "writing" task can use, the "Lock Data Base" ESR and the "Unlock Data Base" ESR. When a task has the dat base locked, no other task can access it. Also, to ensure system efficiency, a task which has issued a "Lock Data Base" ESR is normally allowed to complete its accesses to the data base and not be a candidate for preemption until it has issued a "Unlock Data Base" ESR.

11.2.4.2 Message Communication

Explicit message communication between tasks is not supported by P-3C Update.

Files, however, are used to allow tasks to share and pass data. Since the data is shared or passed only among tasks of the same subprogram, it is more efficient to use files than the tactical common data base.

11.2.5 UYK-7 Standard Executive

11.2.5.1 Database Protection

In the UYK-7, simultaneous access by multiple CP's to any scheduling lists and other critical code is prohibited. This is implemented using the test-and-set technique described in ATEP/MAX.

11.2.5.2 Message Processing

A fixed amount of memory is available for use by user modules as message packing areas. The total amount of memory reserved for this purpose is a SDEX/7 compile-time parameter. The system messages packing area is divided into segments consisting of a number of fixed length packets. The number of segments, segment size and packet lengths for each segment are SDEX/7 compile-time parameters.

11.2.5.3 Immediate Messages

Any module may initiate an immediate message to the user module designated as the Common System Module.

The Common System Module will be immediately executed, preempting the requesting module. Upon completion, the module initiating the immediate message shall be restored and run.

11.2.5.4 System Messages

During program loading and initialization, an area of memory is set aside as a system message packing area. Through SDEX/7 compile-time parameters, the user may determine the length of the area, the number and size of segments within the area and the size of system packets within each segment.

Storage in the system message packing area may be obtained via an ESR. Once the packet is assigned, the user module may use it to pack a message. The sending ESR is then used to initiate the message to up to four (4) receiving user modules.

11.2.5.5 Local Messages

User modules may utilize local data storage areas for the purpose of sending messages. These local messages have the same format and are handled in the same manner as system messages with the following exceptions:

1) message packing areas are provided by user modules and not obtained via ESRs.

2) user modules are responsible for ensuring that the packet is not reused before the last receiving module has processed the message.

11.2.6 COS/UYK-20

11.2.6.1 Inter-task Communications

An inter-task communications capability is provided by the scheduling/priority control functions which validates the parameters and determines the type of scheduling requested. It then stores the parameters to be forwarded in executive memory, and creates a pointer to the parameters.

11.2.6.2 Synchronization

Although supporting a multiprocessing and multi-computer environment, COS has no synchronization facilities or database locks.

11.2.7 AN/BQS-13

There is no mention of synchronization or message handling in the documentation referenced.

11.2.8 Proteus

11.2.8.1 Inter-Module Communication

Inter-module communication is provided. A message task can receive either a system or a local message.

A system message is information in a fixed format placed in the Task Communication Work (TCW) of a message task by the executive. There is no way for a sending task to determine if a receiving message task has received a system message.

11.2.8.2 Local Messages

A local message is information in a user defined format which is intended for task-to-task communication. The address of the local message is placed in the TCW of a message task by the executive. The task which sends the local message is able to determine if the message has been received.

11.2.9 SDEX/20

11.2.9.1 Inter-Task Communication

SDEX/20 user modules may send two types of messages -- local messages and system messages. Local messages can be sent only to one receiver, while as many modules as desired can receive a system message. In either case, the receiving module(s) are entered in SDEX/20's list of modules awaiting message scheduling, and control is returned to the initiating job.

11.2.9.2 Synchronization

There is no mention of synchronization or database locking mechanisms in the documentation referenced.

11.3 Discussion

All documentation referenced covered only a few aspects of synchronization.

Most Navy systems solve synchronization problems by having tasks declared non-preemptive. The advantage to this is that critical code, locking a database, is executed immediately. Conversely, system response may suffer.

As is evidenced by the system discussions above, most of the existing Naval executives support some type of message communication scheme. Further, many support features to protect shared databases from simultaneous modification by parallel tasks. Both of these facilities are very necessary in any configuration with multiple tasks. The integrity of common databases must be protected against simultaneous modification or else the necessary requirements for system consistency will be lost (the system would no longer be deterministic). Additionally, there must be a means for parallel tasks to communicate data between each other allowing the multiple task approach to be utilized for complex tactical systems. Without such facilities, such coordination would be difficult indeed.

A major fault of the discussed executives is their lack of true synchronization primitives. Generally, the method for synchronizing tasks consists of sending a message to the message entrance of the desired task. The message entrance code then must accomplish the synchronization necessary. Of course, the sending task could issue a request to schedule another task but this forces task execution from the beginning again. Task synchroni-

zation primitives such as "BLOCK", "WAKE", are not directly implemented and must be explicitly programmed. Not only does this fault cause increased programming overhead, but increased system overhead probably results as scheduling calls must be used to "fake" such primitives. Evidently, the original task idea assumed that a task would complete one job without preemption by a task of equivalent priority. (This idea is mainly supported by the fact that few executives have a time-slicing function). Thus it was not necessary to plan for synchronization of parallel tasks as this situation should not arise. However, newer programming techniques such as process driven I/O (i.e., having processes awoken upon reception of the appropriate interrupt) and parallel computation (not just totally different tasks sharing a processor) make these type of synchronization facilities necessary. Any standard Navy executive should supply such facilities as system primitives.

A step further than just supplying message communication and process synchronization facilities is their combination. Primitives such as "BLOCK" and "WAIT FOR MESSAGE" are far easier to program than having separate message task entrances. All the capabilities of the current message communications schemes (e.g., datasharing, database protection) can easily be incorporated into such primitives. Concepts such as "events"and "semaphores" are not new and offer a better approach to this problem. Such mechanisms should be investigated in an effort to supply facilities to synchronize parallel tasks in a more efficient and flexible manner.

It is recommended that FOS be capable of incorporating far more sophisticated forms of process synchronization than are currently used, as these will be necessary for applications in the future.

It is also recommended that the message communication features provided by FOS be both more transparent to the user, so that the hardware configuration of the machine is unimportant at that level, and more general, to handle the problems of large data-base systems.

CHAPTER 12

FILE MANAGEMENT

12.1 Introduction

This chapter discusses the file management schemes used by the Navy's Executive systems.

Of the Executives analyzed, three (3) do not contain any file management system at all. Three (3) Executives do not have file management per se but do have relatively sophisticated memory management features which resemble, in part, those of file management. The remaining Executives contain standard file management systems with provisions for file manipulation, protection and storage on secondary storage devices.

12.2 Feature Chart

	File Management System	Core Resident Files	Files Used For System Use
ATEP	Yes	Yes	Yes
ATEP/MAX	Yes	Yes	Yes
ATEP/MMS	Yes	Yes	Yes
P-3C	Yes	Yes	No
UYK-7 (SDEX)-7	No	No	No
COS/UYK-20	Yes	No	Yes
BQS-13	No	No	No
PROTEUS	No	No	No

12.3 System Comparison

The following is a system-by-system comparison of file management features for the various Executives.

12.3.1 ATEP

ATEP does not have a file management system. Instead, it uses segments located in main memory as data bases for modules who wish to use such areas for information keeping, message transmission and scratch pad work. Modules have access to four (4) types of database organizations:

12.3.1.1 Types of Data

Private data is organized such that each module may have a single private data set (specified at the module's compile time) for unrestricted use. Protection against access by other modules is provided.

Common data allows users to define multi-user data sets. During execution, application modules can request access to any prespecified common data area. Common data provides the principal means for data interchange among program modules.

Temporary data are data storage areas dynamically acquired by requesting modules. Most temporary data areas are required for inter-module messages and working storage. The Temporary Storage Processor allocates space from a storage pool, and re-allocates that space (for messages) when designated receivers have processed the message or (for working storage) upon module completion. If a module wishes a data area to remain for longer, it can request that the area be cataloged. This area may then be retrieved by other modules. Releasing a cataloged area is done by a module request.

Scratch pad data is a common, prespecified area fixed within memory which is used as a workspace by all modules. The data content of a module's scratch pad area is never preserved after termination of the module.

12.3.1.2 Protection

For protection, ATEP will permit each module to specify accessing limitations on the data.

12.3.2 ATEP/MAX

ATEP/MAX has a file management system identical to that of ATEP.

12.3.3 ATEP/MMS

ATEP/MMS has a file management system identical to that of ATEP.

12.3.4 P-3C UPDATE

The P-3C file management module supervises and controls the use of system files by executive routines and application tasks and is responsible for the accounting of all files, whether in main memory or on the drum.

Files are generally utilized for:

a) management of data

b) to share and pass data among tasks of the same subprogram (so that data need not reside in the tactical common data base).

12.3.4.1 Protection

Each file can have four (4) protection attributes:

a) read protect - file cannot be read from

b) write protect - file cannot be written into

c) delete protect - file cannot be deleted

d) attribute protect - permanent system database

Attribute protected files cannot have their attributes changed without rerunning the system generation process, where attributes are initially defined. Non-attribute protected files can have their attributes changed at any time.

12.3.4.2 File Maintenance

There are maintenance routines available that provide the facility for file protection and manipulation. These include:

1. Create file - allows creation of non-physically existing scratch files (i.e., exist by name only).

2. Delete file - allows deletion during system operation of physically existing files that are not delete protected.

3. Change file attribute - change the attributes of any file not attribute protected.

4. Lock file - allows exclusive use of a file. A locked file can be accessed only by the task that locked it. It can only be unlocked by the task that locked it.

5. Unlock file - allows a task to unlock or give up exclusive use of a file previously locked.

6. Open file - indicates an intention to read/write from a file. This call also informs the file management module where to store the file (e.g., in main memory for heavy usage). Files may be opened at any time. This call has no effect on file protection attributes.

7. Close file - indicates that the task is finished reading/writing from the file it had previously opened. Files may be closed at any time. This routine has no effect on usage of that file by other routines.

8. Read file - allows reading of an open file that is not read protected or locked by another task.

9. Write file - allows writing into an open file that is not write protected or locked by another task.

12.3.5 UYK-7 Standard Executive

This Executive contains no file management system.

12.3.6 COS/UYK-20

The File Control Function of the COS/UYK-20 provides the interface to access all files in the system. It is capable of handling user or system file requests.

Since the COS/UYK-20 operates under a loosely linked form of multiprocessing, all file references are passed to the computer connected to the storage device (disk or drum) and are routed through a centralized I/O routine to determine the appropriate peripheral device handlers.

12.3.6.1 File Management

There are four (4) functional areas of file management:

File Definition provides the user with the capability of defining the characteristics of a file, all parameters being placed in the File Control Block.

File Assignment creates the link between the user program and file control information. It requests the assignment of facilities not currently assigned to the program.

File Release terminates use of a file and releases any facilities that were assigned to the file.

File Manipulation provides the user with the means to access files through a set of file and record manipulation functions. Before these functions are executed, they are compared to the file definition data in the File Control Block to see if the function is valid for that file. Functions provided are:

1. Open File
2. Close File
3. Read File
4. Write File
5. Read (with lock)
6. Position

File Deletion causes the specified file to be deleted from the file catalogue and its area made available for allocation.

12.3.7 BQS-13

There is no file management system for this Executive.

12.3.8 Proteus

There is no file management system for this Executive.

12.4 Discussion

There are two (2) basically different types of file management systems.

The ATEP, ATEP/MAX, and ATEP/MMS's file management system manages data bases in main memory allowing modules areas for scratch work, temporary storage and messages. No provisions are made for permanent files residing in secondary storage,e.g., having a program take a collection of data from memory and file it for later analysis.

In comparing the COS/UYK-20 and the P-3C UPDATE, the P-3C provides more features considered essential in a file system (esp. protection) than the COS/UYK-20. This is at first surprising, because the COS/UYK-20 is a multiprogramming, multiprocessing communications oriented system which requires much file manipulation and access.

The other Executives do not have explicit file management systems.

FOS will be designed so as to provide any necessary level of file system support. This will range from no file system, through sophisticated main memory management, through a full file system including secondary memory use, up to a Multics-like file system for those applications which need it.

CHAPTER 13

ERROR MANAGEMENT

13.1 Introduction

All of the executives must have facilities for detecting errors, and for recovering on error, because in a critical Navy operation, error recovery must be attempted at all costs.

13.2 System-by-System Comparison

The following is a system-by-system discussion of the error management functions of the Navy Executives.

13.2.1. ATEP

13.2.1.1 Introduction

The ATEP Executive handles errors on two (2) possible levels. At initialization, a definition table specifies a central error module to schedule when an error condition arises. If a central error module is not designated, then when an error occurs, the error entry point associated with that module is scheduled.

13.2.1.2 Error Handling

There are a few basic actions that can be taken upon the occurrence of an error. For errors occurring during an interrupt state operation and during user error processing, the user is notified of the error and computer operation is halted. The more common and important action is encountered when an error occurs during the normal operation of a program. In this case an error packet is built that describes the environment at the time of the error and is passed to the appropriate module (chosen as described above). For device errors which are not solved when retried, an appropriate module is again chosen and scheduling requested.

13.2.1.3 Types of Error

There are four (4) classes of errors possible. Class I is power failure. Under this condition, control memory is stored and computer operation halted. Class II errors occur as a result of program operation. Class III errors are those associated with I/O operations and Class IV are program related (i.e., illegal instructions, etc.).

13.2.2 ATEP/MAX

ATEP/MAX's error handling function is identical to that of ATEP.

13.2.3 ATEP/MMS

ATEP/MMS's error handling function is identical to that of ATEP.

13.2.4 P-3C Update

13.2.4.1 Error Handling

When a program or memory protect error occurs, it is possible that the cause of the fault was either hardware and/or software. When an abnormal interrupt occurs, in the P-3C executive special actions are taken to isolate the cause. Initially, the task that has caused the error is re-allocated and rescheduled in a different section of memory. If it again fails, the executive marks the task unavailable in the Task Directory. If, however, the task does complete successfully the second time, then the portion of memory in which it was first executed is labeled defective until a memory check is performed. If the memory check fails, that portion of core is labeled unusable.

If, however, the stack passes the memory check, then it is once again made available with the limitation that if another error occurs in that area, it will immediately be labeled unusable. If there is suspect of an error in the executive software, then TACCO is notified. After the loss of five (5) sections of memory, a new minimized configuration is implemented.

13.2.5 UYK-7 Standard Executive

The error management function identifies hardware and software errors upon their occurrence and takes action as directed by user modules.

13.2.5.1. Error Handling

The executive allows user modules to selectively register responsibility for processing any or all error conditions. The user module can then receive immediate CP control at the entrance of its own error module, should the error occur. The errors encountered may be of two (2) types. There are interrupts signalled by the AN/UYK-7 hardware, and those isolated by the Standard Executive functions (i.e., caused by invalid data or improper sequencing of operations). The user modules may register responsibility for individual errors or groups of errors such as hardware, software, IOC and executive software errors.

If there is no user module for a specific error, the CP stops, indicating the type of error encountered on the maintenance panel, and waits for the computer operator to decide on the action to be next taken by the executive. The options are described below.

If there is a module, then error information shall be communicated to the registered module through an "error packet" built by the error management function and passed to the module. When the user module has completed its processing of the error, CP control will be returned to to the executive.

The executive has two (2) major options upon the processing of any error (either by module or operator):

1) Ignore the error and continue processing from the point of the error.

2) Return CP control to the scheduling function.

13.2.6 COS/UYK-20 Executive

There is no apparent error management scheme for this system. It is most likely accounted for in the other sections of the executive as the need for an error/interrupt handler arose.

13.2.7 BQS-13

There is no specifically defined error management scheme for this executive. The closest approximation to an error management technique is the interrupt processor. The interrupt processor handles each of four (4) classes of interrupts. These classes are as follows:

1) Fault and Hardware interrupts;

2) Program error and clock interrupts;

3) I/O interrupts;

4) Executive calls.

When an interrupt occurs for types 1) and 3) listed above, the interrupt handler interprets the condition and passes control to a program called the Performance Monitoring/Fault Location Program.

There appears to be no user recovery options available under this system.

13.2.8 Proteus

13.2.8.1 Error Handling

There are four (4) types of general error types recognized by the executive:

1) arithmetic

2) logic

3) I/O

4) mainframe (i.e., hardware).

Only on the occurrence of an arithmetic error (e.g., overflow, divide check) will the executive return control to the user program and allow it to process the problem. On an I/O error, the executive attempts to recover but if impossible, it merely returns a code to the user. On a mainframe error, (e.g., power, parity) the executive immediately halts the machine. On a logic error, (addressing, protection violations) a message is sent to the module's message task and the task which caused the error is aborted. One final feature is that any of these errors may be masked off either on a task basis or on system basis.

13.2.9 SDEX/20

13.2.9.1 Error Management Function

SDEX/20's error management function handles all error conditions detected by other portions of the executive. Through this function a user module may register responsibility for processing its own or system error conditions; should an error occur, the registered module receives immediate processor control at its message entrance. If no module processes the error (none is registered) processing is stopped. SDEX/20 will not retry or otherwise manage errors; all processing is halted. Errors sent to the error management function include

those indicated by the UYK-20 interrupts (including hardware errors) and those due to incorrect processing or an improper ESR request.

13.2.9.2 Error Types and Registration

The following error types are recognized:

- Class I hardware errors
- Class II software errors
- Class III IOC software errors
- Executive software errors

User modules may register for specific errors or whole classes of errors. A power tolerance error will always be processed immediately by the error management function.

13.2.9.3 User Error Processing

When an error occurs, an error packet is built and CP control is given to the registered user module at its message entrance. Upon return to the error management function, the user can specify to SDEX/20 that it should:

- Ignore the error and return
- Return CP control to the scheduling function
- Return CP control to the initiation function

If a module has not registered for the error, processing stops and status information is displayed on the maintenance panel.

12.3 Discussion

Error management in FOS will primarily need to be flexible. At the same time, very high reliability will also be needed.

PART IV

SUMMARY

CHAPTER 14

SUMMARY AND CONCLUSIONS

14.0 Introduction

This survey was prepared with the objective of providing researchers involved in designing a standard operating system for the Navy with an appreciation of the Naval software environment. An attempt has been made to present an overview of the Navy operational environment to study the effectiveness of present day Navy executives, and to determine the environment that a future standard operating system will be required to support.

To attain these ends we have (a) provided a framework for analyzing and studying the Navy operational environment, (b) analyzed the major executives used in the Navy together with their operational environments, and (c) made a critical feature-by-feature analysis of these executives. A discussion of the strengths and weaknesses of these executives has also been made, wherever possible. (A companion volume to this Facilities Orientation report has also been prepared, consisting of twelve Trip Reports, each describing the details of a specific visit to a Naval installation or facility, and discusses the information gained, listing observations, problems, recommendations, etc).

Three major points emerge from the results of this study. These are:

(a) The reasons why the executives currently in use by the Navy would be inadequate to serve as a standard for future tactical applications.

(b) The pressing need for a standard family of operating systems in the Navy.

(c) Recommended features that should be incorporated into this family of operating systems as a result of this study.

Points (a) and (b) are further discussed in the remainder of this chapter. Point (c) discussion can be found through the chapters dealing with a feature-by-feature analysis and therefore has not been repeated here.

14.1 Current Executives

The diversity of the Navy software environment has led to the devlopment of a variety of Navy executives. Some of these operating systems were designed with specific applications in mind while others were designed for application independence. In any case, the results of this multi-dimensioned effort show no single operating system that is flexible enough to fulfill the variety of requirements of a standard Navy executive. Further, none of the existing executives that were explored in this report offered ease in extension--that is, modification of the existing executives to extend existing facilities. The various existing executives are not suitable as a flexible base for future Navy executives for one or more of the following reasons:

(1) Failure of the executive to support all the facilities necessary for Naval applications (e.g., peculiar forms of scheduling).

(2) Failure of the executive to be tailorable to a specific application so that particular facilities and only those facilities are supported in the application's version of the system (i.e., no more overhead than necessary).

(3) Failure of the executive to be easily modified so that changes required to convert it into a more flexible (in the sense of an application dependent configuration) executive can easily be implemented.

14.1.1 Facilities Available

Although several of the existing Navy executives support a sizeable set of facilities, no single executive discussed in this report supports all the facilities that would be required in the full spectrum of Navy tactical executives. (Examples of missing features can be seen in earlier sections of this report). Generally, the executives designed for a specific application lack features required by user applications in other applications (e.g., ATEP does not have a file management system that would be required for an M.I.S. application). Executives meant to be application independent support a very basic set of facilities; instead, they rely on the application programs to supply anything of a special nature (including I/O drivers). Furthermore, the basic set of facilities themselves are limited; memory management, I/O device management, and/or multi-processor management are often supported in a limited fashion (often not supported at all).

14.1.2 Tailorability

Most of the existing Navy executives offer limited control over their final facilities at systems generation time. Generally, such options involve scheduling selection, table-sizes, and hardware configuration support. This approach does not offer enough flexibility to tailor existing executives to the variety of environments in which a standard Navy executive would operate. The reasons for this are summarized as follows:

(1) Even with options, there is still an extremely restricted set of features (i.e., a scheduler that is required but not available as an option).

(2) Originators of the various systems may have missed future requirements for a Navy executive; consequently, no options were provided.

(3) Current Navy executives maintain much of their flexibility by forcing application dependent responsibility onto the user. This approach suggests leaving all the programming to the user for maximal flexibility. This may unnecessarily increase the work load on the user and, furthermore, encourage a proliferation of unnecessarily incompatible extensions - - The system should provide flexibility in a manner that offers maximal user support in all executive configurations.

(4) Finally, none of the Navy executives explored in this study offered true configuration control--that is, although options were provided, there were frills on top of an extensive and non-flexible executive kernal. Except for scheduling, no options were provided to allow real user control of particular system functions.

Such control over which system functions, and what particular versions of those system functions are to be in a particular application-tailored executive is not available. This is a serious flaw, which alone makes these executives unsuited for serving as a basis for a family of compatible operating systems.

14.1.3 Modifiability

None of the material referenced in this study suggested an existing Navy executive that was structured so that modification could reasonably be accomplished. Thus, applications that required additional different executive features would require major software investment to use an existing executive (the very reason that there are so many executives in existence). Because of this essential modification flexibility, the executive to be used as a Navy standard must be modularly structured so that changes can be effected. Further, such modules should be interdependent in a structured manner so that module changes can be accomplished with a limited effect on other system modules. No executive in this study offered a structured modularity to enhance executive tailorability. This lack of structured modularity restricts existing systems to these

applications for which the options have been pre-planned, and makes extension of such executives to other applications extremely difficult at best.

14.2 The Need For A Standard F.O.S.

A reduction of the large amount of software proliferation is the main reason why a standard operating system is needed in the Navy. Currently, the requirements of almost every new application mandates the design and implementation of a new executive, because either no existing military executive can provide the necessary facilities or it is too cumbersome to modify an existing executive.

The primary argument for standardization is the tremendous costs associated with software proliferation. This cost stems not only from the amount of time and money expended in the design and implementation of independent operating systems for each computer system and mission, but from the costs associated with the maintenance, updating and lack of transferability of such produced software.

The problem of recurring software costs is especially severe for Navy systems. Because many systems are intended to be used for many years, the costs of keeping programmers trained for maintaining and changing programs for a wide variety of systems is considerable. Moreover, as applications and requirements change over the years, retraining programmers and rewriting programs becomes a major cost. Moreover, as Navy applications become increasingly complex, the complexity of the operating systems increases considerably and unless the executive is sufficiently flexible, it will often have to be rewritten.

Ideally, the Navy would want a single standard operating system that can be used in any hardware configuration and application. The past software experiences in the Navy, as well as the computer industry in general, indicate, however, that a single operating system cannot meet the anticipated future needs for both functionality and performance. Navy executives must support a very wide variety of

applications, ranging from shipboard logistics and MIS type systems to very real-time systems; such as, missile guidance, which require a very quick response time and a large amount of computational power.

The Navy tactical environment thus necessitates the development of a compatible family of operating systems, which comprise a compatible family in the sense that they would have the ability to invoke a feature in the same manner in all family members in which the feature is present. Additionally, particular feature absense does not result in a change of behavior for other features. This must be accomplished in a manner such that a small version of the operating system does not pay the penalty of generality required only of a larger,more elaborate version. Further, the larger version will not pay the penalty of the specificity required in a smaller version.

Such a family of operating systems will also provide support for the flexibility of equipment configuration that is necessary for tactical systems. Specialized I/O handlers for the wide variety of military peripherals would be interchangeable from system to system, and there would be standardized interfaces between dependent systems. For example, if a small system is used as the front end of a larger system, it would use a smaller member of the family, but would provide all necessary support for information transfer and inter-computer communication systems.

Many Navy installations experience difficulties in the development of a system under an executive that was designed for operational use. The operational executive usually lacks many development aids and utility programs (such as tracing facilities). Using a family of operating systems would enable development work to proceed under one subset that provides all the features necessary for development and the operational system would run under another compatible subset.

14.3 Necessary Features of an F.O.S.

From this study of Navy executives we can identify the main features of the wide range of characteristics that must be supported by such a family of operation systems. The primary function of any operating system is to control and coordinate all equipment resources such as processors, main storage, secondary storage, I/O devices, and files; to resolve conflicts, attempt to optimize performance, and simplify the effective use of the system; and to interface between the physical computer hardware and applications programs, user programs, and associated software.

If a family of operating systems is to be designed, a facility is needed to select the subset required for a particular environment. (For example, the lowest level of support for a specific functionality can be non-existent). The operating system's system generation facility must be capable of generating compatible configurations ranging from extremely small to the extremely large (e.g., multiprocessor and minicomputer). The main functional capabilities that the operating system should provide general design features that should be incorporated into the family, and specific functional features that should be provided to support the Navy environment. Some of these general design features are discussed below.

14.3.1 General Design Features

14.3.1.1 Modular and Structural Independence

A high degree of modular and structural independence is needed so that changes and modifications to elements of the system are not necessarily propogated to other, unrelated elements of the system. Clean functional separation of components will, for example, permit functionally equivalent modules optimized on different parameters to be interchanged.

This clean functional independence will eliminate the problem of rewriting entire executives for the sake of small functional changes (for example the TWAES executive which was not modular a nd had to be rewritten to add a logical file handling capability

for the TESE system).

14.3.1.2 System Generation

System Generation of a specific operating system is of importance in having such a family of operating systems readily accepted in all the Navy installations, because problems in the past have made Navy programmers skeptical of modifying software that comes from other installations (the "not-invented-here" syndrome).

Moreover, one of the main reasons that the Navy has a proliferation of incompatible systems is that military operating systems have been developed by programmers knowledgeable in the tactical problem, but lacking expertise in operating systems. The use of a good system generation facility for such a family of operating systems would permit the programmer knowledgeable in the tactical problem to generate a specific operating system from a high level description of the mission, platform, hardware, real-timer response requirements, and other criteria as necessary. Such a facility should also contain tools for the evaluation of such factors as size estimation, free running time determination, testing and validation.

14.3.1.3 Machine Independence

The family of operating systems must be as independent as possible of the machine upon which they are implemented. Machine independence is important because of the uncertainty of the hardware that will be used for future tactical applications, and because it is important to be able to implement the FOS on much more advanced hardware in the future. To attain a high degree of machine independence, the design should use only a few, well-isolated and cleanly invoked machine-specific features, and the non-machine-specific portions should be defined in a high-level machine independent language.

14.3.1.4 Protection and Security

Protection and security aspects are very important in military systems to prevent data from being destroyed or accessed by unauthorized users, including dynamic revocation of access rights and access filtering. The system design must be able to completely protect programs and data from access by programs which have no need to use them. Note that this criterion does not demand a fixed protection policy throughout all members of the operating system; it only requires that the family have a mechanism by which any reasonable policy can be implemented. A proven way of providing for maximum security is through the design of hierarchically modular capabilities-based operating systems, and the FOS design team should make use of these advanced concepts.

14.3.1.5 Conversion Considerations

Conversion considerations are important in the design of a future standard family of operating systems to minimize both transition costs and any other undesirable impact to Navy operations and activities. This means that a facility must be provided to run under environments similar to those provided by existing Navy executives, e.g., COMMON and CMS-2Q. In summary, the emphasis throughout the design of FOS should be on a non-traumatic transition from the current operating environment. It is clear that at some point during the design phase a decision will have to be made whether to make subsets of the family compatible with current executives, or whether it is more convenient to provide facilities for emulating current systems (such as the facilities that a virtual machine environment provides).

14.3.1.6 Support of Data Base Systems

There is an increased need for data base facilities in the Navy, not only because of the greater use of MIS type applications (such as logistics and supply systems), but because of the large amounts of data that must be managed while performing tactical functions. To support these requirements, FOS must not only

have a well-defined file system (a facility that many Navy executives totally lack) but it must also be able to support two types of data base management systems, as follows:

(a) A flexible, easy to use, DBMS that provides for quick and inexpensive implementation will be needed. Since it is often difficult to specify in advance exactly what the requirements of the system will be, it is more advisable to bring up prototypes on a flexible DBMS, and then test the prototypes against the user's needs as many times as necessary. The prototype must have the ability to incorporate all the provisions for security, validation and changing interrelationships of data as necessary. Present day technology suggests that a relational data base system has all the necessary requirements for building these flexible information systems.

(b) There is a tradeoff between flexibility and efficiency of data base systems. The system that is used for building a prototype is often inefficient once the structure of the system has been "frozen." Another DBMS will be needed to "freeze" the prototypes built using the system described in (a) above.

This DBMS will not only process and answer user queries, but also provide information when needed in a given tactical application. The tactical application programs request information using cleanly invoked queries, leaving the data accessible by other applications programs or users.

Ideally these DBMS's would comprise a compatible "family" of data management systems, all supported by FOS. Thus, it would be a relatively easy matter to "freeze" a prototype once it has been developed, with a high degree of application independence.

14.3.1.7 Abnormal Condition Handling and Recovery

Abnormal condition handling and recovery is of utmost importance in tactical systems, both because Navy systems are very prone to accidents in a tactical environment, and because they are performing critical functions (e.g., flight navigation) that require backup systems. Military systems have typically solved this problem by dual-redundancy (i.e., using another complete, backup system), but FOS should provide more advanced facilities for recovery, including automatic retry where possible, graceful degradation (fail soft), reconfiguration around a failed module (whether hardware or software), and graceful restoration and smooth integration of restored modules. In addition, the system should be responsible for capturing information describing the environment (equipment, procedures, and data) of an abnormal condition. Self-test and diagnostic support for the hardware and software is also necessary, which includes protection of the software operating system so that a system error will not cause an operational failure.

14.3.1.8 User Interfaces

User interfaces must be provided in as high-level a manner as possible. This includes debugging facilities the interface between the operating system and the programmer, and the interface with the operator.

14.3.2 Summary

The standard family of operating systems must be designed and implemented in an easy-to-use, well-structured, hierarchical, modular fashion, taking advantage of the current state-of-the-art in system structuring. It must allow easy modification of and addition to or of any portions of the system. Procedures must exist for the flexible, application-oriented tailoring of individual versions of the system for particular uses.

The system must be capable of incorporating state-of-the art tools to be used in all areas of the system (e.g., security,

database management, etc). It must be possible for individual versions to be as compatible with each other as necessary for easy program portability (both between different applications, and between developmental and operational system versions). Compatibility with current Navy systems must also be provided to some extent.

APPENDIX

APPENDIX A -- Sources of Information

Meetings and Presentations Attended: Persons Contacted

1. Naval Underwater Systems Laboratory — 19 August 1974
   Newport, R. I.

   Trident Fire Control and Sonar System

   Mr. George Bain
   Mr. Phil Sedgwick
   Mr. Tom Conrad

2. Naval Underwater Systems Center — 19 August 1974
   New London, Connecticut

   Sonar Systems; BQS-13 Executive

   Mr. Jim Shores
   Mr. Bob Gordon

3. Grumman Aircraft Corporation — 27 August 1974
   Bethpage, New York

   E-2C program; F-14 Fighter System; A6B; TFCC

   Mr. Martin Lewis

4. Naval Air Development Center — 28 August 1974
   Warminster, Pennsylvania

   P-3C System; Proteus

   Mr. Hank Stuebing

5. FCDSSA — 29 August 1974
   Dam Neck, Virginia

   Software Support and Development Facilities

   Cdr. Jack Cooper

6. RCA — 5 September 1974
   Moorestown, New Jersey

   AEGIS; ATEP; ATEP/MMS

   Mr. Warren Mulle

7. Naval Air Development Center — 4 December 1974
   Warminster, Pennsylvania

   AADC Review Meeting

8. Naval Weapons Systems Center — 11 December 1974
   Dahlgren, Virginia

   MK86 Gunfire Control System; Fleet Ballistic Missile Systems

   Mr. Dave Brown

9.	Raytheon Company Wayland, Massachusetts TARTAR System Mr. Dave Higgenbotham	12 December 1974
10.	N.E.L.C. San Diego, California CUDIXS; TESE; TWAES; NAVMACS; NALCOMIS Mr. Warren Loper	15 May 1975
11.	Camp Pendleton California MTACCS Col. Chase	19 May 1975
12.	FCDSSA San Diego, California NTDS; SHARE/7; LHA; IFDS Cdr. Sharp	20 May 1975
13.	Raytheon Company Wayland, Massachusetts Further details on TARTAR System Mr. Dave Higgenbotham	7 July 1975

(In addition, numerous meetings were held with Intermetrics and persons involved with the design of the CS-4 Language).

Partial List of Documents Examined

1. Interim Report, Master Executive Control for the Advanced Avionic Digital Computer, Volume I: Summary, Honeywell Document Z9506-3018, for Naval Air Development Center, by K.J. Thurber, E.D. Jensen, L.L. Kinney, P.C. Patton, L.C. Anderson, L.A. Jack, and P.A. Houle, June 18, 1972.

2. Final Report, Operating System/AADC Preliminary Functional Specification, Honeywell Document Z9506-3018, for Naval Air Development Center, by K.J. Thurber, L.L. Kinney, P.A. Houle, P.C. Patton, L.A. Jack, E.D. Jensen, and L.C. Anderson. October 18, 1972.

3. Interim Report, Master Executive Control for the Advanced Avionic Digital Computer, Volume II: Technical Document, Honeywell Document Z9506-3018, for Naval Air Development Center, June 18, 1972.

4. Interim Report, Master Executive Control for the Advanced Avionic Digital Computer, Volume III, Honeywell Document Z9506-3918, for Naval Air Development Center, June 18, 1972.

5. Draft Report Language Benchmark Tests, Intermetrics, Inc., by J.T. Pepe and J.R. Nestor, August, 1973, for N.E.L.C.

6. A CS-4 Primer, Volume I: Basic Features, Intermetrics, Inc., for N.E.L.C., by R. Fourer, January, 1974.

7. Final Report on the CMS-2 Compiler System Part 2 Implementing the CMS-2 Compiler on the Advanced Avionics Digital Computer, Systems Consultants, Inc., Naval Air Development Center, Washington, D.C., October 13, 1970.

8. Advanced Development Model Specification, AADC Program Management Unit (PMU) Minimum Configuration, Raytheon Co., Missile Systems Division, for Naval Air Development Center, June, 1974.

9. Final Report for AADC Arithmetic and Control Logic Design Study, Part I, Raytheon Company, for Naval Air Development Center, November 1972.

10. Final Report for AADC Arithmetic and Control Logic Design Study, Part II, Raytheon Company, for Naval Air Development Center, November, 1972.

11. Final Report for AADC Arithmetic and Control Logic Design Study, Part III, Preliminary Programmers Reference Manual, Raytheon Company, Missile Systems Division, for Naval Air Development Center, December 1972.

12. All Application Digital Computer: Course Notes, United States Naval Postgraduate School, by G.H. Syms, March, 1973.

13. AADC DPE Programmer Reference Manual, Raytheon Company.

14. Future Naval Aircraft Subsystem/AADC Interface Definition for Operational and OBC Requirements (U), Final Report Volume I, Grumman Aerospace Corporation, for Naval Air Development Center, April, 1972.

15. ATEP/MAX Programmer's Reference Manual, Tactical Data Systems, Fleet Combat Direction Systems Support Activity, San Diego, CA, February, 1974.

16. Document Type: Draft Technical Report, Project: Analysis of Major Computer Operating Systems, August, 31, 1970.

17. CS-4 Language Reference Manual and Operating System Interface. Naval Electronics Laboratory Center, for Naval Air Systems Command, December, 1973.

18. Analysis of the CMS-2 Programming Language, for Naval Air Development Center, December, 1971.

19. Requirements for Digital Computer Program Documentation, Naval Ordnance Systems Command, Department of the Navy, November, 1971.

20. Automatic Data Processing Program Reporting System (ADPPRS), Department of the Navy, SECNAVINST, February, 1973.

21. Proceedings Volume I, DOD Joint ADP Conference of Central Data Systems Design and Programming Activities, for U.S. Army Computer Systems Command, October, 1972.

22. Management Information Systems, Army Catalog of Automated Data Systems, Department of the Army Pamphlet No. 18-1-1-1, December, 1971.

23. USN/USMC Future Data Systems Requirements and Their Impact On the All Application Digital Computer (AADC), Computer Sciences Corporation, San Diego, California.

24. AADC Software Test Plan, Revision 1, Final Report, Systems Software Group, Sperry Univac, January, 1974.

25. MTACCS Exploratory Development Studies Task F. Report Critique of Intermetrics AADC Language in Response to N00039-70-C-3552, Amendment A00044, Item 0006, Hughes Aircraft Company, June, 1974.

26. ATEP/MAX Executive Computer Program Performance Specification, Fleet Combat Direction Systems Support Activity, June, 1974.

27. Trip Reports Nos. 1-6, Facilities Orientation, M.I.T. Sloan School, OS/AADC Group, June, 1975.

28. Trip Reports Nos. 7-12, Facilities Orientation, M.I.T. Sloan School, OS/AADC Group, August, 1975.

29. Interim Standard Airborne Digital Computer, second draft, June 16, 1975, No. AS-4352(AV).

30. Evaluation of the AADC Program of the United States Navy, by Research and Consulting, Incorporated, Feburary, 1975.

31. Computer Program Design Specification for the ATEX Executive Program, by Computer Sciences Corporation, San Diego, CA.

32. Computer Program Performance Specification for AEGIS Tactical Executive Program/Multiprocessing and Memory Sharing (ATEP/MMS) Kernel. By AEGIS Program Office, RCA, October, 18. 1974.

33. MAX/CP, Comparative Analysis Summary.

34. The PROTEUS General Purpose Executive Program, by Naval Air Development Center, Warminster, PA.

35. AN/BQS-13 DNA Sonar System Program Specification, by IBM Electronics Systems Center, December 18, 1970, Revision 1.

36. Navy and Marine Corps Tactical Digital Equipment Catalog, by Tactical Digital Systems Office, Feburary 15, 1973.

37. Input/Output Interfaces, Standard Digital Data, Navy Systems, Military Standard, August 30, 1973.

38. TWAES System I Software Summary, July 15, 1975.

39. MTACCS Test Bed System Description, by Hughes Aircraft Co., June 15, 1973.

40. Computer Program Performance Specification for Aegis Tactical Executive Program, by RCA, March 24, 1971.

41. Computer Program Performance Specification for Standard AN/UYK-7 Executive, in accordance with TADTASK 2-73, Final Draft, January 18, 1974.

42. System Software Design Requirements P-3C Update Functional System, by Naval Air Development Center, January 25, 1974.

43. Preliminary Computer Program Performance Specification for a Communications Oriented Operating System

APPENDIX B -- CURRENT NAVY HARDWARE

Presented here are brief summaries of three of the more popular computers used in Navy tactical systems; the AN/UYK-7, the AN/UYK-20, and the PROTEUS CP/IO. Particularly, the following aspects are discussed:

a. CPU overview
b. instruction set
c. addressing
d. protection
e. interrupts
f. timing mechanisms
g. memory
h. I/O mechanism
g. reference literature

The following discussions are brief summaries of the hardware; for more complete detail consult the listed references.

AN/UYK-7

The AN/UYK-7 computer is a ruggedized multiple-processor system designed and manufactured by the Univac Division of Sperry Rand Corporation for military applications. The functional and physical modularity of the system affords a variety of processing and input/output capabilities for immediate and future applications.

Central Processor Unit

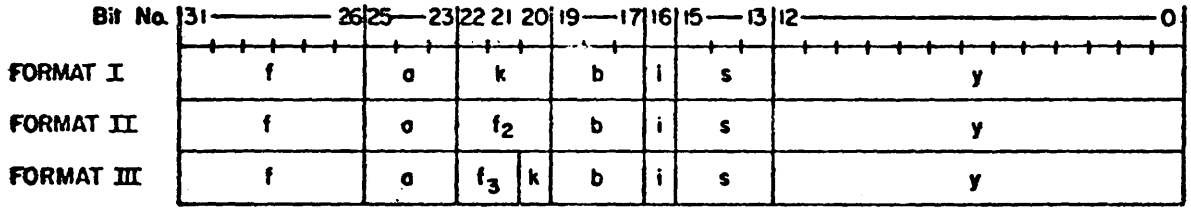
The central processor module contains all the control, arithmetic, and timing circuitry required for instruction execution. Central processors operate in two different modes; the interrupt state executes the executive type functions, and the task state processes the worker programs. A separate set of seven index, eight base, and eight arithmetic accumulator registers is available to the processor in each state.

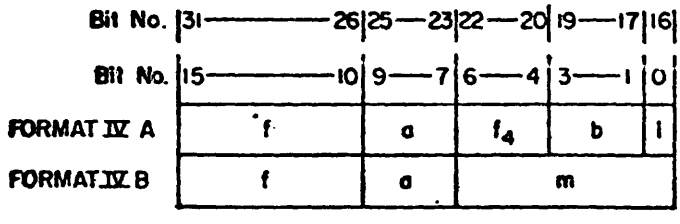
Instruction Set

The AN/UYK-7 is a 32-bit computer that offers 130 basic whole and halfword instructions that operate with 32-bit parallel, one's-complement, binary arithmetic. It contains both fixed and floating point hardware and can operate on 8-, 16-, 32-, or 64-bit operands. To complement its interrupt statemode, the AN/UYK-7 supports a set of privileged instructions which include input or output transfer initiation, read and control of the monitor clock, and control of the various processing activities in both the task and interrupt states.

FIGURE B1

AN/UYK-7 Instruction Formats





Elements of the Word are Interpreted as Follows:

field	basic definition
f	6-bit function code
f_2	3-bit subfunction code
f_3	2-bit subfunction code
f_4	3-bit subfunction code
a	3-bit accumulator register designator
k	operand interpretation designator
m	6-bit shift count designator
b	3-bit index register designator
i	indirect addressing designator
s	3-bit base designator

Instruction execution times vary from 1.0 microseconds to greater than 17.0 microseconds. The average instruction execution time is approximately 2.0 microseconds.

A novel feature of the UYK-7 processor is its ability to repeat an instruction. By utilization of the REPEAT instruction, a class of UYK-7 instructions can be repeated a specified number of times. At every repetition, the count in a specified index register is reduced until zero is reached; for example, multiple loads and stores can be accomplished in short form.

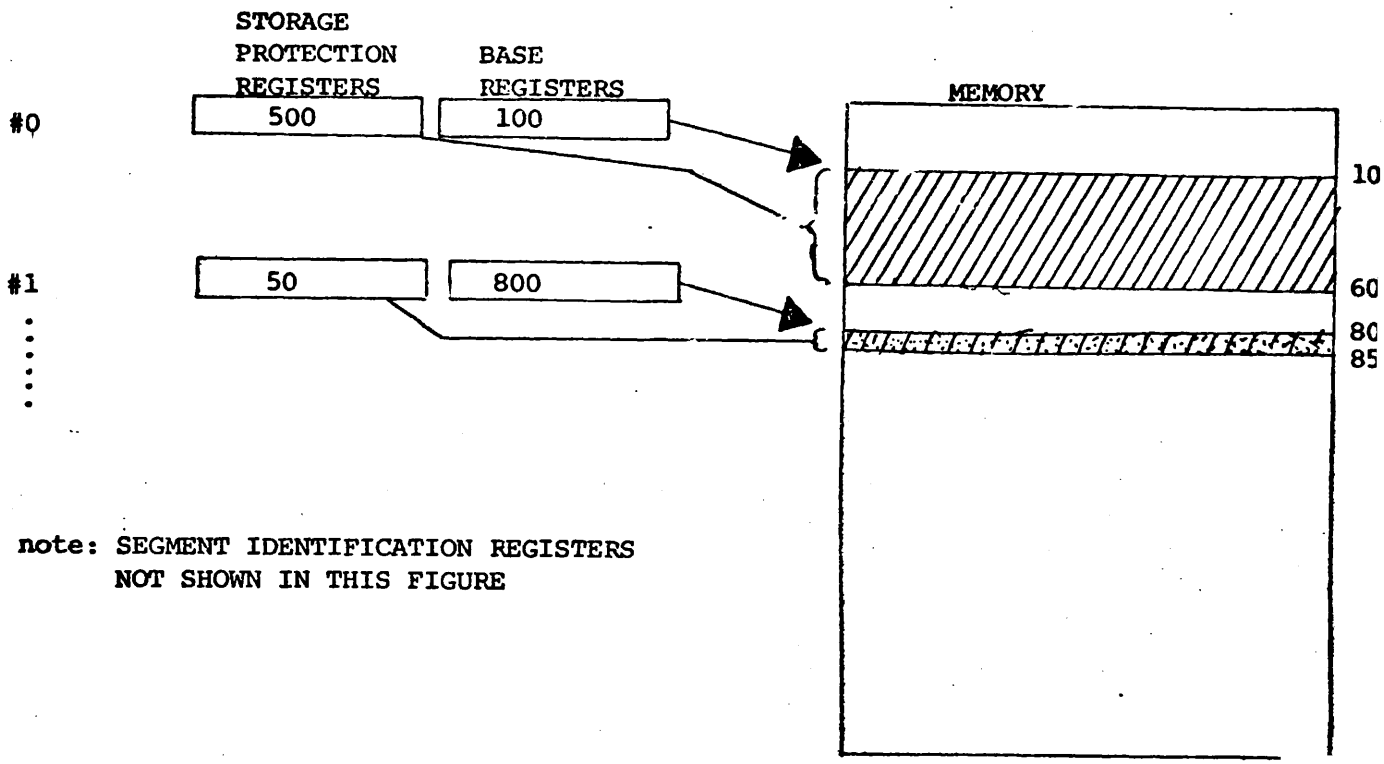
Addressing

The central processor can address up to 262, 144 memory locations (with internal address modification) via any specified base register. An index register may be used to modify the displacement address of an instruction, by a value up to 65, 536. Additionally, the AN/UYK-7 has the capability for both direct or indirect addressing and variable character length addressing.

Protection Hardware

Closely related to the addressing capabilities of the AN/UYK-7 is its form of protection. A processor, operating in task state, can be prevented from performing read and/or write operations in defined areas of any memory module. Three groups of Control Memory Registers govern memory lockout functions. For any block in memory (up to 65K words) a base register (one of eight) holds the beginning address, an associated storage protection register defines the lockout protection (read, write execute, or indirect address) and block size (number of words), and a segment identification register contains the relative address of the segment identifies (that address in main memory from which the lockout information is transferred). Since all memory references utilize these base registers (and thus their corresponding storage protection registers), proper initialization of worker base registers via the privileged instructions to access them) allows the executive to control worker task memory extent.

FIGURE B2



AN/UYK-7 HARDWARE PROTECTION EXAMPLE

Interrupts

The interrupt machanism of the AN/UYK-7 is fairly conventional. Interrupts are divided into four classes, these are the: fault and hardware interrupts, program faults and error interrupts, input and output program faults, and program-initiated entrance to the interrupt state. When honoring an interrupt the processor stores the current state of the machine and picks up appropriate new state information corresponding to the class of the interrupt. The new state information contains the starting location of the interrupt processing routine along with new status information such as lockout requirements for future interrupts. Interrupts that are currently locked out by the processor are held pending until such lockout is removed. Upon

completion of interrupt processing, the interrupt routine restores the state that existed at the moment of interrupt.

Timing Mechanisms

A 16-bit control memory register can be activated to decrease its count at the rate of 1024 counts per second. Additionally an external clock with a separate oscillator module can be incorporated into the system.

Memory

Main memory is composed of modules of randum access core storage with a read restore cycle of 1.5 microseconds. Each central processor or input/output controller can address up to 262K words (i.e., 16 modules of 16K each). Each processor also contains a 512 word NDRO memory containing various firmware routines(e.g., interrupt analysis, initial load).

Input/Output

The UYK-7 input-output controller contains the necessary control and timing circuitry to conduct orderly input and output transfers of data, external commands and external interrupts between accessible memory modules and the external devices on 4, 8, 12, or 16 full-duplex channels. IOC functions are governed by a chain of commands (set up by the central processor) initiated by one or more controlling central processors. Input/output programs define buffer areas, channel numbers, and any functions related to word or byte size, imposed monitors, and transfer types. A processor has the capability to support up to four I/O controllers; a particular controller can be controlled by 1, 2, or 3 central processors.

References

The preceding information extracted from the AN/UYK-7 TECHNICAL DESCRIPTION, Sperry Univac.

AN/UYK-20

The AN/UYK-20 computer is a ruggedized mini-computer designed and manufactured by the Univac Division of Sperry Rand Corporation for military applications. The computer consists of a central processor capable of executing instructions from a program stored in memory and an input/output controller that handles all I/O data transfers between the UYK-20 computer and peripheral devices.

Central Processor Unit

The central processor is a 16-bit word microprogrammed digital data processor capable of executing instructions from a program stored in memory. It contains one set of 16, 16-bit, general-purpose registers that the instruction set is tailored to manipulate. A second set of 16 general-purpose registers is available as a plug-in option; a particular bit in the processor status register controls which of the two register sets is currently active. The UYK-20 does not have multiple states nor does it have any form of privileged instructions.

Instruction Set

The UYK-20 is a 16-bit machine that has a repitoire of approximately 150 instructions which use both 16-bit and 32-bit instructions and performs operations using 8, 16, or 32-bit operands. Additional variations on the UYK-20 processor instruction set is obtained by the designation of instructions for operations between the processor registers, and operations between memory and the processor registers.

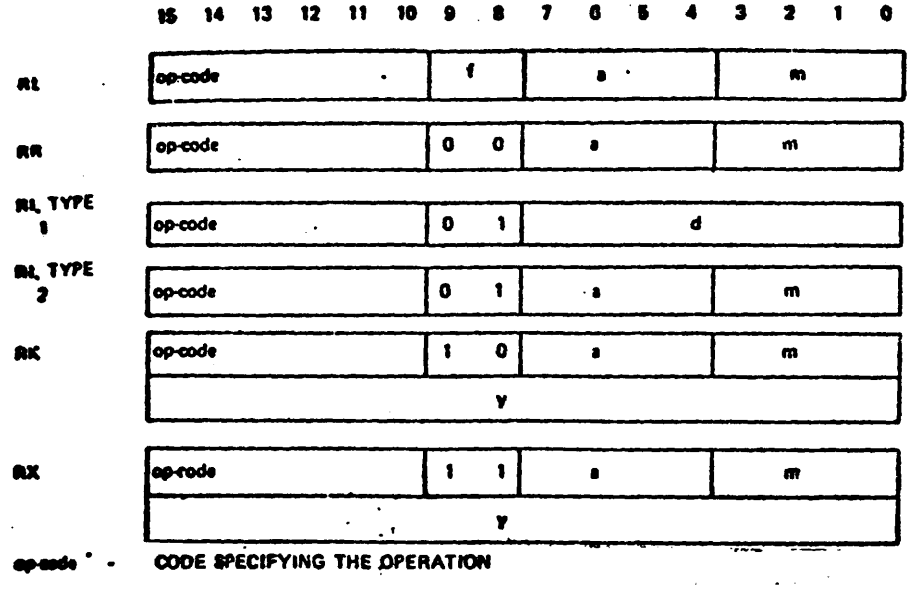
The UYK-20 performs integer/fixed-point arithmetic operations; an optional microprogrammed math package is available that supplies floating point operations and a variety of trigonometric and hyperbolic functions. Other instructions perform logical operations or I/O control functions.

Instruction execution times average 2.0 microseconds. Of course, different instruction lengths or increased operand lengths increase this time. Floating point operations greatly increase this; thus, although this processor has high speed register to register instructions, the instruction mix determined by the application will be a major factor in final speed.

It should be noted that although it is possible to have an AN/UYK-20 supplied with two sets of general registers, and there exists an Executive Request instruction (one that generates an interrupt), there does not exist an executive state. All instructions are executable at any time by any process.

FIGURE B3

AN/UYK-20 INSTRUCTION FORMATS



op-code - CODE SPECIFYING THE OPERATION

a - GENERAL REGISTER OR SUBFUNCTION DESIGNATOR

m - GENERAL REGISTER OR SUBFUNCTION DESIGNATOR (RR, RI, RK, RX) or 4-BIT LITERAL CONSTANT (RL)

d - DEVIATION VALUE (TWO'S COMPLEMENT)

f - SUBFUNCTION CODE

v - ADDRESS OR ARITHMETIC CONSTANT

Addressing

The central processor can address up to 65,536 word locations (with internal address modification) via an address translation scheme based on processor page registers. The main memory is divided into 64 pages each of which are 1K long. A relative address is formed from the current instruction by taking the specified displacement and adding the contents of an optionally specified index register. The six high order bits of the relative address specify one of the 64 page address registers which contains page numbers. These page numbers are used to select one of the 64 pages of main memory for absolute memory location. The lower ordered ten bits of the relative address specify the offset into the specified page. Any operation that stores a word in main memory also sets a bit in the specified page register.

The AN/UYK-20 has the capability for direct or cascaded indirect addressing. Further, such indirect indexing can be further indexed depending on the setting in a status word of the processor.

Protection Hardware

The AN/UYK-20 has no capability for hardware protection per se; however, by manipulation of the page registers and user cooperation a limited form of protection can be derived. Basically this involves restricting the user address space so that page registers point only to user areas. However, because the user has the ability to manipulate these page registers, a determined task can do anything it desires. In summary, the AN/UYK-20 supplies "enough" protection for a friendly environment.

Interrupts

Interrupts are generated by events within the processor (e.g., underflow) or the I/O controller. The system interrupts are classified in three priority levels; interrupts within a level are assigned a priority rank within that level. As each interrupt is honored, the current state of the processor is saved and new state is retrieved

from locations dependent upon the class of the interrupt. The new state information contains new masking information to determine what interrupts are now to be locked out. Interrupts that are locked out are held pending until the lockout is removed. Upon interrupt routine completion, the status of the machine at the moment of interrupt is reloaded and the interrupted task continues.

A unique feature of the AN/UYK-20 is the method it uses for dispatching the processor for interrupt processing. Although there are only three clasess of interrupts, each specific interrupt has a unique value within its class that is used to index into a table of dispatching address for that class. In this manner, interrupt decoding is hardware supported resulting in lower overhead for interrupt processing.

Timing Mechanisms

The AN/UYK-20 contains a real-time clock and monitor register. The real-time clock is a 32-bit register which is used as a count-up timing register. A one kHz real-time clock oscillator controls the counting speed of the register; an external jack is provided to allow an external clock oscillator. The monitor clock is a 16-bit register which is used as a countdown timer. The clock is loaded with a positive value and counts down at the frequency of the real-time clock. When the count reaches zero, an interrupt is generated.

Memory

Main memory consists of memory array boards each which contain 8,192 sixteen-bit words of magnetic core storage with a 750-nanosecond read-write cycle time. The maximum amount of memory is 65K (eight, 8,192-word boards). A block of 192 NDRO memory words is provided in the central processor; the programs contained in this memory are fixed at the time of manufacture.

Input/Output

The AN/UYK-20 contains an Input/Output Control section which performs I/O control operations and data transfers asynchronously with central processor operations. The Input/Output Controller contains a control memory which holds controlling data and instruction pointers for each channel as it performs its operations of control and data transfer between peripheral device and the computer memory. The Input/Output Controller utilizes a special set of instructions which are fetched from the computer memory. The central processor sets up an "I/O program" which consists of these instructions and initiates channel action on this program. The AN/UYK-20 supports 4, 8, 12, or 16 channels via this I/O Controller.

References

The preceding information was extracted from THE USER'S HANDBOOK FOR AN/UYK-20(V) COMPUTER SUPPORT SOFTWARE, Part 5 Hardware Description, Univac, Sperry Rand Corporation.

PROTEUS

The PROTEUS computer system is an advanced signal processor system designed and built by the International Business Machine Corp. It is a modular system which consists of a general purpose (CP/IO) processor that serves to control arithmetic processors, a storage transfer controller, an input signal conditioner, and digital I/O channels. The following hardware review is mainly concerned with the general processor module (CP/IO) of the PROTEUS system.

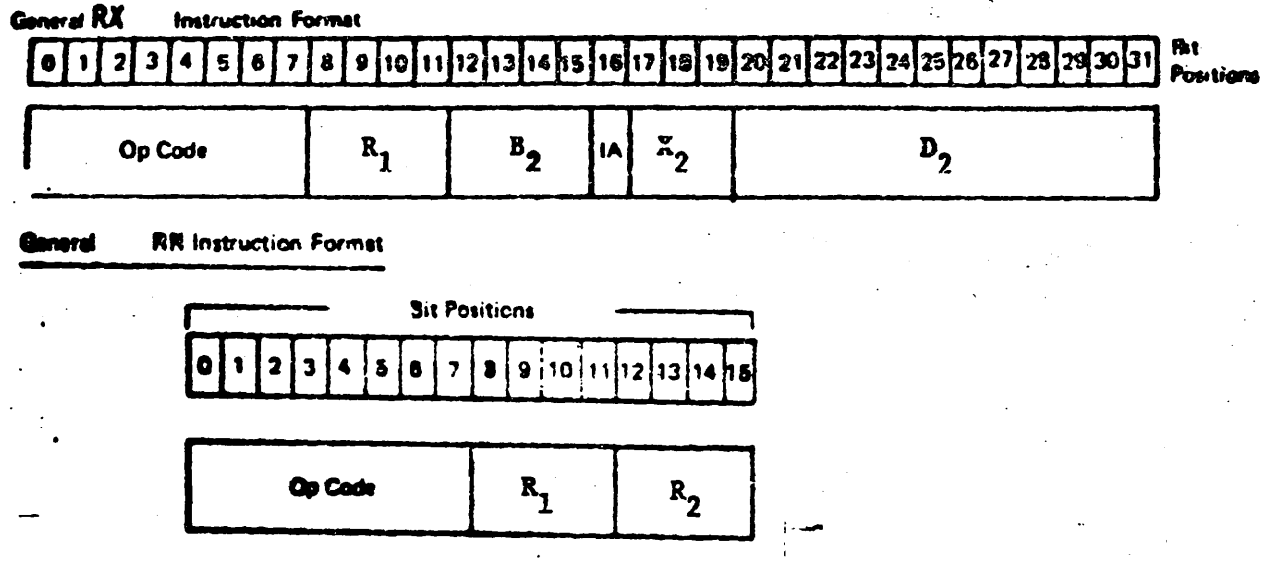
Central Processing Unit

The CP/IO module is an extensively micro-programmable general purpose processor that is responsible for controlling the rest of the PROTEUS system. It is architecturally very similar to the

IBM 360/370 series. It is a byte oriented machine with words consisting of four 8-bit bytes. Sixteen (16) general registers are provided; these registers can be addressed as 16 full-word registers or 32 halfword registers. Further, the CP/IO provides a register stack that allows 16 sets of these 16 general registers. Such sets are switched automatically by the processor upon the execution of particular instructions. Also, the CP/IO supports both a problem state and a privileged state.

Instruction Set

The instruction set of the CP/IO is very similar to that of the IBM 360/370 series. Additional halfword instructions are provided to allow mainpulation of the halfword registers of the computer. Further, due to the extensive microprogrammability of the CP/IO, different versions of the CP/IO (i.e., different applications) have additional instructions to facilitate better "application" efficiency. Also, instructions are furnished to allow the CP/IO to control the other parts of the PROTEUS system.



R - register
B - base register
IA - indirect bit
X - Index register
D - displacement

As in the 360/370 IBM computers, there is a set of privileged instructions that can be executed only when the machine is in the privileged state. Such instructions include protection modification, I/O initiation, and other control functions.

Addressing

Although the CP/IO can generate a 32-bit address, it is limited to 128K words. The address mechanism is the same as for the IBM 360/370 series; that is, a displacement in an instruction is modified by the contents of a specified base register (optional) and a specifie index (optional) register which results in an absolute address. Although microprogrammed, the CP/IO hardware automatically provides this address modification scheme to the microprogram.

The CP/IO has the ability for indirect addressing. Such addresses can be modified only before the indirection.

Protection Hardware

The CP/IO protection mechanism is block oriented. Main memory is divided into pages of 2K each. Each page has associated with it protection bits that determine whether the page can be used by the currently running process. The modification of these protection bits can be accomplished by a privileged instruction; thus, it is possible for the CP/IO to offer protection in other than friendly environments.

Interrupts

The CP/IO interrupt structure is identical (except for the addition of new classes of interrupts) to the IBM 360/370 series interrupt structure. Interrupts are grouped in several classes. Upon interruption, the processor stores the current state of the machine and retrieves a new state for the machine; both the current state storage and new state retrievel come from locationsdependent upon the class of the immediate interrupt.

Within the state specification, it is possible to lockout interrupts of various classes (some within particular classes); interrupts locked out will be held pending. The new state retrieved will specify the address of the routine to run. Upon completion, the interrupt routine restores the machine to the state at the moment of interruption.

Timing Mechanisms

The CP/IO contains a real-time clock that is a countdown timer. Upon reaching 0 it signals an interrupt.

Memory

Main memory consists of up to 64K 32-bit words that consist of 8K memory modules. There are also additional memories connected to the CP/IO: a control store memory which serves to hold the micro-programs for the CP/IO. A bulk memory accessible via a STORAGE TRANSFER CONTROLLER holds program data and other information not immediately necessary in the CP/IO. Other memories related to the use of the ARITHMETIC PROCESSORS and their functions are available to the CP/IO via special instructions.

Input/Output

The CP/IO has several mechanisms for inputting and outputting data. There is a set of 8 sink-source pairs of digital channels that are handled similarly to the I/O channels of the IBM 360/370; that is, the CP/IO initiates the particular I/O channel to execute a "channel" program that the CP/IO has previously set up in memory. There is a STORAGE TRANSFER CONTROLLER that enables the CP/IO to move data throughout the PROTEUS system. The INPUT SIGNAL CONDITIONER is the CP/IO's way of inputting and outputting real-time acoustic data into the bulk storage modules.

References

The proceding information was extracted from the IBM PROTEUS PRINCIPALS OF OPERATION, IBM, Federal Systems Division, Manassas, Virginia.

I N D E X